Kuliah 1 — Web, dari Ujung ke Ujung

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 1

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini tidak ada awan yang tetap menjadi awan: kita mengikuti satu halaman dari tekanan tombol sampai gambar di layar, lalu menyajikan halaman kita sendiri dengan lima cara. Buku: Bab 1, Subbab 1.1–1.20.

Cakupan Pertemuan Ini

1. **Pipa-pipanya** — internet dibandingkan dengan web; alamat IP dan port; nama DNS; TCP, UDP, dan QUIC.
2. **Percakapannya** — satu halaman ditelusuri dalam dua puluh langkah; URL diurai; permintaan, respons, metode, kode status, dan header HTTP, sampai ke tingkat byte.
3. **Kedua ujungnya** — apa yang dikerjakan peladen web untuk setiap permintaan; halaman statis dan dinamis; kode sisi peladen dan sisi klien; alur perenderan peramban.
4. **Gambaran besarnya** — tingkatan arsitektur, garis keturunan markup, generasi-generasi web.
5. **Praktik langsung** — perangkat kerja di VS Code, lalu satu halaman Profitina Laundry yang disajikan sebagai berkas dan dari PHP, Java, Python, serta C#.

Urutannya mengikuti urutan ketergantungan: pengalamatan, lalu penamaan, lalu percakapan, lalu apa yang dilakukan kedua program di ujung-ujungnya. Buku: Subbab 1.1 dan 1.3.

Tempat Kita Memulai: 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.

HTML, CSS, JavaScript, PHP, MySQL, XML, JSP, ASP.NET, ADO.NET: semuanya ada untuk menentukan apa isi kedua potong teks itu.

Studi kasusnya adalah **Profitina Laundry**: tiga gerai di Surabaya — Rungkut, Keputih, dan Wiyung — dan aplikasi jadi dengan sekitar 3.000 nota data contoh.

Program-programnya ada di `WebPro/Code_new/Chapter01/`, satu folder per teknologi: `html`, `php`, `java`, `python`, `aspnet`.

Biarkan terminal tetap terbuka dan ketikkan perintahnya, jangan menempelkannya; ketika sesuatu berhasil, rusakkan sekali dengan sengaja dan bacalah kegagalannya. Buku: Subbab 1.1.

Internet: Jaringan dari Jaringan-Jaringan

> **DEFINISI. Jaringan komputer** adalah dua komputer atau lebih yang dihubungkan sehingga yang satu dapat mengirim data ke yang lain. **Internet** adalah jaringan sedunia yang terbentuk dengan menggabungkan banyak jaringan yang lebih kecil, yang semuanya sepakat berbicara dengan keluarga protokol yang sama, **TCP/IP**.

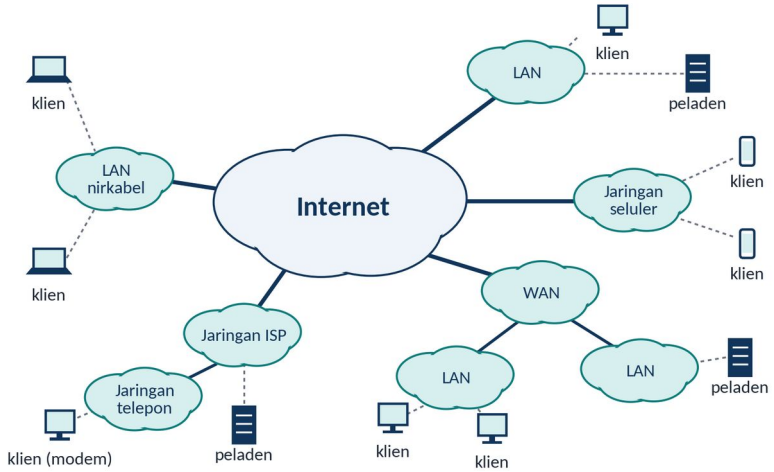
LAN — satu gedung atau kampus; **WAN** — satu kota atau negara; **LAN nirkabel** — yang oleh stiker di *router* disebut Wi-Fi.

Tidak ada yang membangun internet sebagai satu jaringan: internet adalah **kesepakatan** antara jaringan-jaringan yang dibangun orang lain.

Protokol adalah seperangkat aturan: apa yang dikirim, dalam urutan apa, dan apa yang harus dilakukan dengan balasannya.

Switch di dinding hanya meneruskan pesan dari soket tempat pesan itu tiba ke soket lain yang tepat. Buku: Subbab 1.2.1 dan 1.2.2.

Banyak Awan, Satu Kesepakatan



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.

Dalam Gambar 1.1 setiap awan kecil adalah jaringan milik seseorang; awan besar di tengah hanyalah kesepakatan di antara mereka. Buku: Subbab 1.2.2.

Web Adalah Satu Layanan di Atasnya

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

Internet juga membawa surel (SMTP), transfer berkas (FTP), gim, telepon, *streaming* — semuanya di internet, bukan di web.

Apa pun yang berbicara HTTP adalah bagian dari web, entah ada manusia yang melihat pertukarannya atau tidak.

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

Dalam aplikasi web kita tidak pernah menulis program yang mendengarkan jaringan: peladen web mengerjakan pekerjaan sulit itu dan menyerahkan kepada kode kita permintaan yang sudah diurai. Buku: Subbab 1.2.3–1.2.5.

IPv4: Tiga Puluh Dua Bit

DEFINISI. Alamat IP mengenali satu antarmuka jaringan. **IP** memasukkan data ke dalam amplop berlabel, yaitu **paket** (header ditambah muatan), lalu meneruskannya menuju alamat pada label itu.

IPv4 menulis 32 bit sebagai empat **oktet** desimal: `103.94.189.4`. Setiap oktet bernilai 0 sampai 255, karena $2^8 = 256$.

Persamaan (1.1) menghitung semua alamat IPv4 — sekitar 4,3 miliar, lebih sedikit daripada perangkat yang terhubung di bumi.

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

CONTOH TERURAI. $189 = 128+32+16+8+4+1$, jadi oktetnya 10111101. Seluruh 103.94.189.4 adalah 01100111 01011110 10111101 00000100 — titik-titiknya untuk manusia, dan tidak ada titik yang pernah dikirimkan.

Alamat yang memuat 256 *bukan sekadar salah, melainkan tidak dapat dinyatakan. Periksa setiap kelompok dengan menjumlahkan kembali nilai tempatnya:* $64+32+4+2+1 = 103$. *Buku: Subbab 1.3.1 dan 1.3.2.*

IPv6, dan Port

IPv6 memakai 128 bit: delapan kelompok empat digit heksadesimal, dengan satu deret nol boleh disingkat menjadi `::`, seperti `2606:4700:3031::6815:33af`.

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

Persamaan (1.2) adalah 2^{96} kali Persamaan (1.1). Pada 2024 sekitar 28,5 persen permintaan *dual-stack* yang diterima Cloudflare sudah datang melalui IPv6.

> **DEFINISI. Port** adalah bilangan enam belas bit (0 sampai 65535) yang mengenali satu program yang mendengarkan di sebuah mesin. Pasangan (alamat IP, port) adalah satu titik ujung percakapan.

80 HTTP, 443 HTTPS, 3306 MySQL; peladen dalam buku ini: 8000 statis, 8100 PHP, 8080 Java, 5000 Flask, 5100 ASP.NET Core, 5500 Live Server.

Peramban mengandaikan port 80 untuk `http` dan 443 untuk `https`; port lain harus diketikkan. Buku: Subbab 1.3.3 dan 1.3.4.

Apa yang Tidak Dijanjikan IP

Sebuah paket bisa **hilang**, dan tidak pernah tiba.

Dua paket bisa tiba dengan **urutan yang salah**, karena menempuh rute yang berbeda.

Sebuah paket bisa tiba **dua kali**, atau tiba dalam keadaan **rusak**.

> **PENGIRIMAN *BEST-EFFORT*.** IP saja tidak berguna untuk halaman web: halaman yang bagian tengahnya hilang atau paragrafnya teracak bukanlah halaman. Harus ada yang mengubahnya menjadi aliran yang andal dan berurutan — TCP.

IP berusaha sewajarnya untuk meneruskan paket, dan hanya itulah seluruh komitmennya. Buku: Subbab 1.3.5.

DNS: Nama Dibaca dari Kanan ke Kiri

> **DEFINISI. Domain Name System** (DNS) adalah direktori sedunia yang tersebar, yang memetakan nama yang dapat dibaca manusia ke alamat IP.

```
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
```

TLD generik (`.com`, `.org`, `.dev`) dan kode negara (`.id`, `.jp`); akar yang tak terlihat, sebuah titik, berada di sebelah kanan `id`.

`id` adalah domain tingkat teratas Indonesia, `ac` tingkat kedua untuk lembaga akademik, `its` tingkat ketiga, dan `www` satu mesin atau layanan di dalam ITS. `.ac.id` hanya untuk institusi akademik Indonesia yang terakreditasi: ccTLD biasanya membawa aturan tentang siapa yang boleh memilikinya. Buku: Subbab 1.4 dan 1.4.1.

Pencarian, Langkah demi Langkah

1. Komputer Anda bertanya kepada **resolver**-nya (penyedia internet, kampus, `8.8.8.8`, `1.1.1.1`): “berapa alamat `www.its.ac.id`?”

2. Resolver bertanya kepada **root server**, yang menjawab “tanyakan kepada peladen untuk `id`”.

3–4. Peladen `id` berkata “tanyakan `ac.id`”; peladen `ac.id` berkata “tanyakan `its.ac.id`”.

5–6. Peladen `its.ac.id` memegang catatan resminya dan menjawab `103.94.189.4`; resolver meneruskannya, lalu mengingatnya.

Setiap jawaban membawa **time to live**. Sebelum memindahkan situs, turunkan nilainya sehari sebelumnya, lalu naikkan lagi sesudahnya.

Empat pertanyaan kepada empat organisasi yang berbeda, biasanya dalam waktu jauh di bawah sepersepuluh detik; itulah buah dari sifat tersebar. Buku: Subbab 1.4.2 dan 1.4.3.

Pencarian Buatan Kita Sendiri

Kode 1.1 memanggil `socket.getaddrinfo`, fungsi pencarian milik sistem operasi — fungsi yang pada akhirnya dipanggil setiap peramban.

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})")
```

```
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: bentuk penyeimbangan beban yang paling sederhana, dan cara dunia berpindah dari versi 4 ke 6 tanpa satu hari peralihan serentak. Buku: Subbab 1.4.4.

TCP: Aliran yang Andal dan Berurutan

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

Nomor urut mengembalikan byte ke tempatnya; **ACK** (*acknowledgment*) berkata “saya sudah menerima semuanya sampai byte 5.400”.

Pengiriman ulang setelah batas waktu mengirim lagi apa yang belum di-ACK; **checksum** membuang paket yang rusak.

Kendali aliran melindungi penerima yang lambat; **kendali kemacetan** melindungi jalur yang dipakai bersama.

Dari sisi program tidak ada paket sama sekali: byte masuk di satu ujung dan keluar di ujung lain, seperti air di dalam selang. Buku: Subbab 1.5.1 dan 1.5.2.

Jabat Tangan Tiga Arah

SYN — klien mengusulkan nomor urut awalnya; **SYN-ACK** — peladen mengakuinya dan mengusulkan nomornya sendiri; **ACK** — klien mengakui nomor peladen.

Itu menghabiskan satu **perjalanan pulang-pergi** r sebelum satu byte berguna pun bergerak, sehingga pada koneksi baru yang tidak terenkripsi berlaku Persamaan (1.3).

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

> **CONTOH.** Dari Surabaya, r ke peladen terdekat bisa 10 ms dan ke Frankfurt 250 ms: batas bawah 20 ms berbanding setengah detik, secepat apa pun kedua peladennya.

Percakapan diakhiri dengan FIN, atau secara mendadak dengan RST ketika ada yang tidak beres. Buku: Subbab 1.5.3.

TCP, UDP, dan QUIC

Tabel 1.1 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

Tabel 1.1: sifat TCP "semua yang di belakang paket hilang ikut menunggu" disebut head-of-line blocking. QUIC membangun ulang janji-janji TCP per aliran di atas UDP, dengan enkripsi di dalamnya, dan HTTP/3 adalah HTTP di atas QUIC. Buku: Subbab 1.5.4.

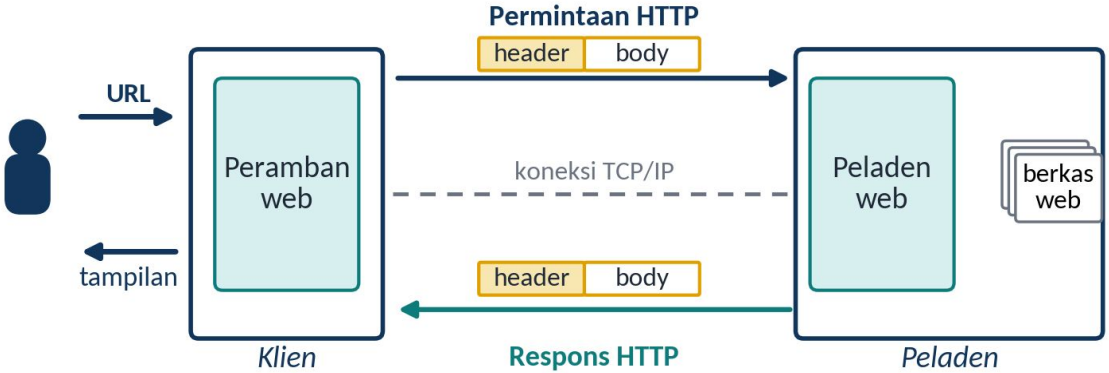
Empat Lapisan, Amplop Bersarang

Tabel 1.2 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 1.2 ke bawah untuk pengiriman dan ke atas untuk penerimaan. Karena HTTP tidak tahu tentang Wi-Fi dan Ethernet tidak tahu tentang halaman, keduanya dapat diganti tanpa mengganggu yang lain. Buku: Subbab 1.5.5.

Pertukarannya dalam Satu Gambar



Gambar 1.2 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*.

Gambar 1.2: URL masuk, permintaan HTTP keluar, TCP/IP di antaranya, respons HTTP kembali, lalu ditampilkan. Buku: Subbab 1.6.

Satu Halaman Ditelusuri: Langkah 1–10

Tabel 1.3 Satu halaman ditelusuri dari ketukan tombol sampai piksel, langkah 1 sampai 10: dari nama yang diketik sampai byte yang diserahkan kepada IP.

#	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

Tabel 1.3: langkah 4 dan 5 adalah DNS, 6 dan 7 jabat tangan TCP, 8 jabat tangan TLS untuk `https`. Buku: Subbab 1.6.

Satu Halaman Ditelusuri: Langkah 11–20

Tabel 1.4 Penelusuran yang sama, langkah 11 sampai 20: melintasi internet, melalui peladen, dan kembali ke layar.

#	Apa yang terjadi	Siapa yang mengerjakan
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

Tabel 1.4: langkah 19 adalah sebab halaman terasa lambat — halaman sederhana terdiri atas tiga puluh sampai empat puluh permintaan. Langkah 14 adalah persimpangan jalan: berkas yang ditemukan, atau halaman yang dihitung. Buku: Subbab 1.6.

URL Memiliki Tujuh Bagian

`https://sari:rahasia@www.its.ac.id:8443/course/webpro/ch01.html?lang=en&page=3#section-1-7` — port habis dipakai untuk koneksi, dan **fragment tidak pernah dikirim sama sekali**.

Tabel 1.5 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**

Tabel 1.5: tidak ada halaman PHP, servlet, view Flask, atau halaman ASP.NET yang dapat melihat fragment; apa yang harus diketahui peladen tempatnya di path atau query. Buku: Subbab 1.7 dan 1.7.1.

Mengurai URL dengan Python

Kode 1.2 `python/urlparts.py` — sebuah URL diurai menjadi bagian-bagiannya oleh pustaka standar Python.

```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)
```

```
query    : lang=en&page=3
parsed   : {'lang': ['en'], 'page': ['3']}
fragment : section-1-7
```

Kode 1.2: query menjadi kamus berisi list, karena `?tag=a&tag=b` itu sah. `parse_str` di PHP dan perulangan Java sederhana menyimpan satu nilai per nama dan diam-diam kehilangan `tag` yang pertama. Buku: Subbab 1.7.2.

Percent-Encoding

Spasi, `&`, `?`, `#`, `/`, atau `%` yang dimaksudkan sebagai data harus disamarkan: byte b menjadi tanda persen dan dua digit heksadesimal, Persamaan (1.4).

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

CONTOH. Spasi adalah $32 = 16 \cdot 2 + 0$, jadi `%20`; `&` adalah $38 = 16 \cdot 2 + 6$, jadi `%26`; `/` adalah $47 = 16 \cdot 2 + 15$, jadi `%2F`.

Formulir memakai dialek lama yang menulis spasi sebagai `+`: `%20` di mana pun dalam URL, `+` di dalam data yang di-*encode* sebagai formulir.

Jika dibiarkan mentah, `&` dalam "cuci kiloan & setrika" memulai parameter kedua dan memotong nilainya menjadi dua. Buku: Subbab 1.7.3.

Dua Encoder Milik PHP

Kode 1.3 `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
```

> **JEBAKAN.** Jangan pernah merangkai URL dari teks pengguna. Pakailah `quote` (Python), `rawurlencode` (PHP), `URLEncoder.encode` (Java), atau `Uri.EscapeDataString` (C#).

Kode 1.3: `rawurlencode` mengikuti RFC 3986, `urlencode` mengikuti aturan formulir. `URLEncoder` di Java hanya mengenal dialek formulir, sehingga versi Java mengganti `+` menjadi `%20` secara manual. Buku: Subbab 1.7.3.

Permintaan HTTP, Lengkap

Empat bagian: **baris permintaan** (metode, target, versi); **baris-baris header** `Nama: nilai`; **satu baris kosong**; dan **badan** yang boleh ada boleh tidak.

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

Dibaca keras-keras: *dengan GET, ambil `/health`, dalam HTTP 1.1; situsnya `localhost:8080`; saya rawhttp 1.0; tutup koneksi setelah selesai.*

HTTP adalah protokol permintaan–respons: satu pesan dari klien, satu pesan balasan, dan hanya itulah seluruh pertukarannya. Buku: Subbab 1.8 dan 1.8.1.

Metode: Aman dan Idempoten

Tabel 1.6 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

*Tabel 1.6: `GET` tidak boleh mengubah apa pun — peramban mengambil lebih dulu, cache menyimpan, crawler mengikuti setiap tautan. Aturan untuk bab-bab awal: **`GET` untuk membaca, `POST` untuk mengubah.** Buku: Subbab 1.8.2.*

Respons, dan Kode Statusnya

```
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
```

Tabel 1.7 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

Tabel 1.7: digit pertama sudah menceritakan semuanya; 4xx berarti Anda yang salah, 5xx berarti saya yang salah. Frasa alasannya untuk manusia; perangkat lunak membaca angkanya. Buku: Subbab 1.8.3 dan 1.8.4.

Kode yang Akan Anda Temui

Tabel 1.8 Kode status yang paling sering dijumpai di bab-bab awal, beserta situasi yang menghasilkan masing-masing.

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
`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
`500`	Internal Server Error	Program Anda mogok

401 BUKAN 403. `401`: saya tidak tahu siapa Anda. `403`: saya tahu persis siapa Anda, dan jawabannya tetap tidak.

Tabel 1.8 adalah pilihan; tabel di buku menambahkan `302`, `429`, `502`, dan `503`. Buku: Subbab 1.8.4.

Header: Kata Sifat Percakapan

Satu baris per header: nama, titik dua, spasi, nilai. Nama tidak peka huruf besar-kecil: `Content-Type` dan `Content-type` adalah header yang sama.

Permintaan: `Host` (wajib — satu IP, banyak situs), `User-Agent`, `Accept`, `Accept-Language`, `Cookie`, `If-Modified-Since`, `Content-Length`.

Respons: `Content-Type`, `Content-Length`, `Date`, `Server`, `Last-Modified`, `Location` (dengan `3xx`), `Set-Cookie`, `Cache-Control`, `Allow` (dengan `405`).

> **HEADER TERPENTING.** Badan respons tidak membawa label sendiri. Kirim HTML sebagai `text/plain` dan pengunjung melihat tag-tag Anda. Ketika keluaran tampak kacau, periksa `Content-Type` lebih dulu.

Buku mendaftar setiap header beserta contoh nilainya dalam dua tabel, satu untuk permintaan dan satu untuk respons. Buku: Subbab 1.8.5.

Berbicara HTTP dari Program

Kode 1.4 Dari `python/rawhttp.py`: satu koneksi TCP, satu permintaan HTTP yang ditulis tangan, satu respons.

```
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
```

Kode 1.4: `socket.create_connection` *menjalankan seluruh jabat tangan tiga arah. Program yang sama di PHP memakai* `fsockopen`*, di Java sebuah* `Socket`*; ketiganya menerima 148 byte. Buku: Subbab 1.8.6.*

Contoh Terurai: 65 Byte

`od -c` memperlihatkan permintaan persis seperti yang diterima peladen; setiap baris diakhiri `\r\n`, dan baris kosongnya adalah satu `\r\n` lagi.

```
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
```

Hitunglah: baris permintaan 20 karakter, baris `Host` 20, baris `Connection` 17, masing-masing ditambah 2, dan baris kosong 2 — Persamaan (1.5).

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

Angka di sebelah kiri adalah oktal, jadi `0000101` adalah byte ke-65, cocok dengan `wc -c`. Tinggalkan `\r\n` terakhir dan peladen menunggu selamanya akan baris kosong. Buku: Subbab 1.8.7.

Tanpa State, dan Berversi

WAWASAN KUNCI. HTTP bersifat **stateless**: setiap permintaan lengkap pada dirinya sendiri, dan peladen boleh melupakan segalanya begitu selesai menjawab. Itulah sebabnya satu dari seribu peladen mana pun dapat menjawab permintaan mana pun.

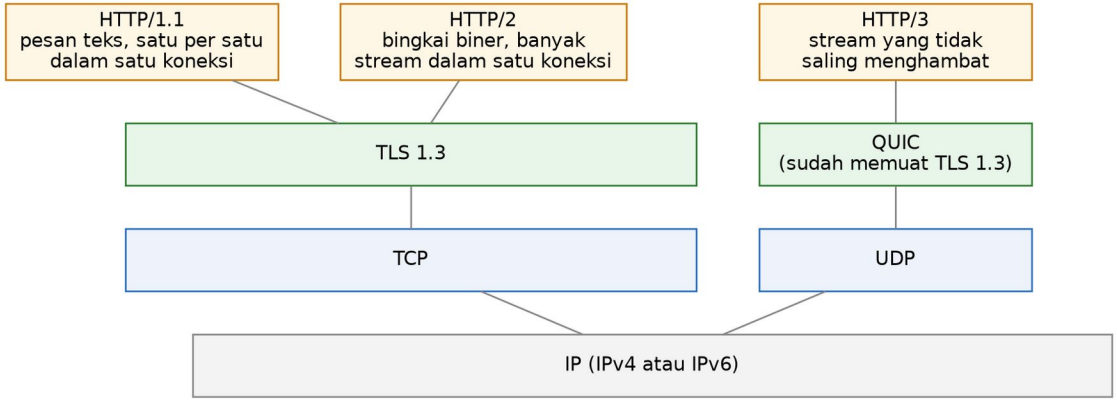
HTTP/1.1 (1997) memakai ulang koneksi dan mewajibkan `Host`; HTTP/2 (2015) membingkai pesan yang sama secara biner; HTTP/3 (2022) membawanya di atas QUIC. **HTTPS** adalah HTTP di dalam TLS 1.3, yang menambah satu perjalanan pulang-pergi, Persamaan (1.6).

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

CONTOH TERURAI. Dengan $r = 40$ ms: `http` memerlukan paling sedikit $2r = 80$ ms menurut Persamaan (1.3); `https` melalui TLS 1.3 dan TCP paling sedikit $3r = 120$ ms.

"Sudah login" dibangun ulang di atasnya dengan cookie: `Set-Cookie` dalam respons, `Cookie` dalam setiap permintaan berikutnya. Gembok berarti terenkripsi, dan sertifikatnya cocok dengan nama — bukan bahwa situsnya jujur. Buku: Subbab 1.8.8 dan 1.8.9.

Tiga Versi di Atas Lapisan



Gambar 1.3 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.

Gambar 1.3: QUIC menggabungkan jabat tangan transpor dan TLS, sehingga HTTP/3 kembali ke $2r$ *pada kunjungan pertama. Buku: Subbab 1.8.9.*

Peladen Web Adalah Perangkat Lunak

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

Produksi: **Apache** (di dalam XAMPP), **nginx**, **IIS**, **Kestrel** (ASP.NET Core), **Tomcat** (servlet dan JSP).

Hanya untuk pengembangan: `php -S`, `python -m http.server`, peladen Flask, `HttpServer` milik JDK.

Peladen web tidak “membuat situs web”: ia membuat berkas dan program *dapat dijangkau*. Apa isinya adalah urusan kita.

Satu mesin dapat menjalankan peladen web, peladen basis data, dan peladen surel sekaligus; laptop yang menjalankan program-program hari ini menampung lima peladen web. Buku: Subbab 1.9.

Sembilan Hal, Setiap Permintaan

1–3. Menerima koneksi; membaca baris permintaan (jika cacat: `400`); membaca header sampai baris kosong, dan tepat sebanyak `Content-Length` byte badan.

4. Memilih situs dari header `Host` — **virtual hosting**.

5–6. Memetakan path ke dalam **document root**, mencari **berkas indeks**; memastikan hasilnya masih di dalam root (`403`, `404`).

7–9. Berkas atau program? Memilih `Content-Type`; menulis baris status, header, baris kosong, badan.

> **JEBAKAN.** Tempelkan path ke root secara naif, dan `/../../../etc/passwd` membaca berkas pribadi: **path traversal**.

Langkah 7 adalah langkah 14 penelusuran, persimpangan jalan itu. Untuk berkas, `Content-Type` diambil dari ekstensinya (`.html` adalah `text/html`, yang tak dikenal `application/octet-stream`). Buku: Subbab 1.9.1.

Halaman Statis dan Dinamis

DEFINISI. Halaman statis adalah berkas di disk peladen, dikirim tanpa perubahan kepada setiap pengunjung. **Halaman dinamis** belum ada sampai diminta: sebuah program membangunnya pada saat permintaan datang.

Muat ulang halaman statis sepuluh kali: tidak ada yang berubah. Muat ulang papan PHP: *Page built by the server at …* berubah setiap kali.

Di jaringan: berkas memiliki `Last-Modified`, sehingga peramban boleh bertanya `If-Modified-Since` dan mendapat `304 Not Modified` **tanpa badan**.

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

Situs dengan banyak berkas statis mendapat sebagian besar kecepatannya dari pertukaran yang satu ini. Buku: Subbab 1.10.1 dan 1.10.2.

Memilih di Antara Keduanya

Tabel 1.9 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

Tabel 1.9: statis tidak berarti primitif dan dinamis tidak berarti dewasa. Aplikasi Laundry yang sudah jadi menyajikan logo, stylesheet, dan skripnya sebagai berkas, dan papan notanya sebagai program. Buku: Subbab 1.10.3.

Sisi Peladen dan Sisi Klien

> **DEFINISI.** Kode **sisi peladen** berjalan di mesin peladen, sebelum respons dikirim. Kode **sisi klien** berjalan di peramban pengunjung, setelah respons tiba.

1. **CGI** — program terpisah yang keluarannya *adalah* balasan itu.
2. **Skrip sisi peladen** — PHP, ASP, JSP, Python: kode di dalam dokumen atau di belakangnya; pengunjung hanya melihat keluarannya.
3. **Skrip sisi klien** — JavaScript (ECMAScript), dijalankan peramban tanpa bertanya kepada peladen.
4. **Plug-in** — applet, ActiveX, Flash: sudah punah, tetapi masih dijumpai dalam kode lama.

Keempat jenis itu disusun menurut siapa yang menjalankan kodenya, dan itulah pembagian yang tepat. Buku: Subbab 1.11 dan 1.11.1.

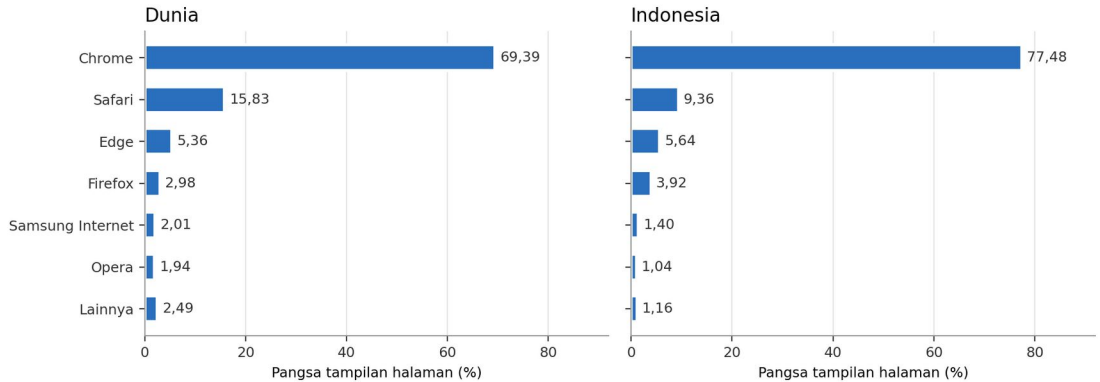
Apa yang Dapat Dilihat Masing-Masing Sisi

Tabel 1.10 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**

Tabel 1.10: tiga baris terakhir adalah aturan keamanan. Kode sisi klien adalah kemudahan, tidak pernah kendali: validasi di klien demi keramahan, dan sekali lagi di peladen demi kebenaran. Buku: Subbab 1.11.2 dan 1.11.3.

Peramban Masa Kini



Gambar 1.4 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.

Gambar 1.4: Chrome 69,39 persen di dunia dan 77,48 persen di Indonesia; semua peramban berasal dari tiga mesin — Blink, WebKit, Gecko. Ukurlah pengunjung Anda sendiri. Buku: Subbab 1.12.1 dan 1.12.2.

Alur Perenderan

1. **Mengurai HTML menjadi DOM** — sebuah pohon; tag penutup yang terlupa *diperbaiki*, belum tentu sesuai maksud Anda.
2. **Mengambil** setiap `<link>`, `<img>`, dan `<script>`; **3. mengurai CSS** menjadi CSSOM.
4. **Render tree** — apa yang akan digambar, beserta gaya terhitungnya (`display: none` ada di DOM, tidak di render tree).
5. **Tata letak** (*reflow*) — geometri yang mahal; **6. melukis**; **7. menyusun** (*composite*).

> **DUA AKIBAT.** `<script>` biasa menghentikan penguraian, maka skrip ditaruh di bawah atau diberi `defer`. Mengubah lebar memaksa tata letak diulang; mengubah warna saja cukup melukis ulang.

Byte halaman dengan skrip di `<head>` dan halaman dengan skrip di bawah itu identik; pengalamannya tidak. Buku: Subbab 1.12.3.

Arsitektur Web dalam Tingkatan

Tabel 1.11 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

Tabel 1.11: presentasi (HTML, CSS, JavaScript), aplikasi (PHP, servlet, Flask, ASP.NET), data (MySQL, SQLite). Masing-masing dapat diganti tanpa menyentuh yang lain; MVC di Bab 7 memperhalus gagasan yang sama. Buku: Subbab 1.13.

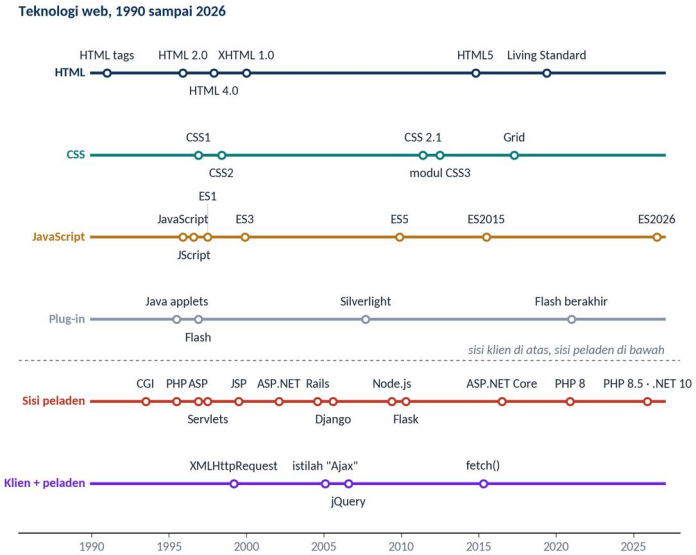
Halaman Web Terkecil



Gambar 1.5 Dokumen HTML pertama dan apa yang dibuat Chromium darinya. `<title>` hanya muncul di tab, tidak di halaman, dan `<b>first</b>` menjadi tebal.

Gambar 1.5: `<title>` ada di `<head>` dan tidak tampil di halaman; `<br>` tidak berpasangan; `<b>first</b>` membungkus yang dipengaruhinya — itulah yang membuat dokumen menjadi pohon. Buku: Subbab 1.14.2.

Dua Puluh Tahun dalam Satu Diagram



Gambar 1.6 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.

Gambar 1.6: sisi klien di atas garis putus-putus, sisi peladen di bawahnya, dan teknik klien-dan-peladen di bagian paling bawah — Subbab 1.11 yang digambar melintasi tiga setengah dasawarsa, dari tag HTML pertama pada 1991 sampai PHP 8.5 dan .NET 10 pada 2025. Garis markup: Scribe, SGML, HTML, XML, XHTML, lalu HTML Living Standard. Buku: Subbab 1.14.1 dan 1.14.3.

Membaca, Menulis, Makna, Agen



Gambar 1.7 Generasi-generasi dalam satu kata masing-masing: setiap generasi menambah satu kemampuan — baca, lalu tulis, lalu jalankan, lalu cerdas.

Gambar 1.7: Web 1.0 membaca, Web 2.0 menulis (prosumer), Web 3.0 makna (Semantic Web), Web 4.0 agen cerdas. Web3 berbeda: Web 3.0 tentang makna, Web3 tentang kepemilikan. Buku: Subbab 1.15.

Perangkat Kerja, Sekali Saja

VS Code dengan Live Server, PHP Intelephense, Extension Pack for Java, Python, dan C# Dev Kit. Bukalah sebuah *folder*, dan pakailah terminal bawaannya.

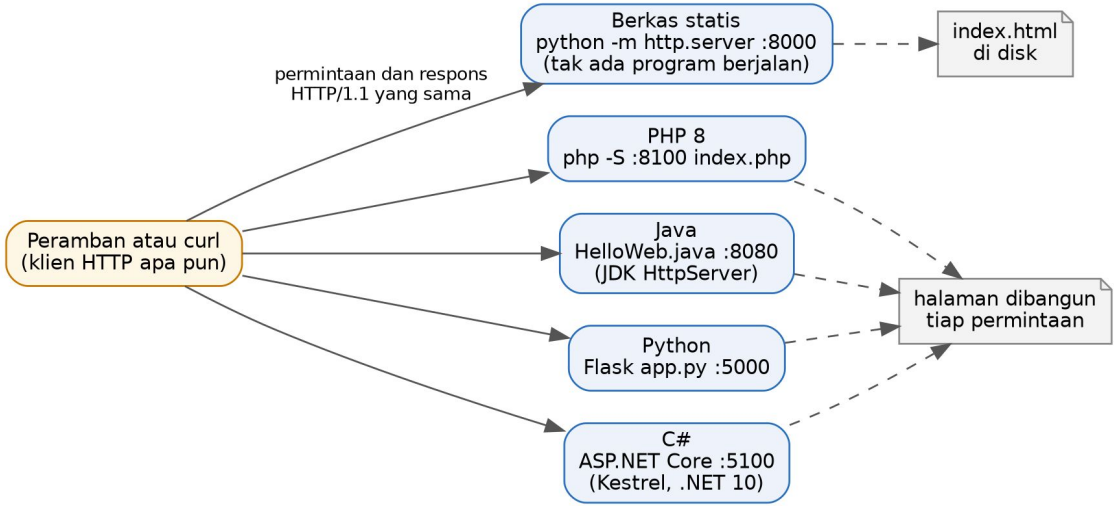
Developer tools peramban (F12): Elements adalah DOM, Network adalah Subbab 1.8 secara langsung, Console, Sources.

PHP 8.2+, rilis terbaru 8.5 (XAMPP 8.2.12 atau pengelola paket), **MySQL/MariaDB** melalui XAMPP, **JDK 25**, **Python 3.14 + Flask** dalam lingkungan virtual, **.NET 10 SDK**, **Git**.

Periksa di terminal *baru*: `php --version`, `javac -version`, `python3 -m flask --version`, `dotnet --version`, `git --version`.

Pasanglah kit, bukan runtime: `javac` harus menjawab, bukan hanya `java`, dan .NET SDK, bukan hanya runtime-nya. Buku: Subbab 1.16.1–1.16.9.

Satu Klien, Lima Peladen



Gambar 1.8 Satu klien, lima peladen. Peladen statis membaca berkas; empat lainnya membangun halaman untuk setiap permintaan. Kelimanya berbicara HTTP yang sama.

Gambar 1.8: peladen statis membaca berkas; PHP, Java, Python, dan C# membangun papan nota Keputih untuk setiap permintaan. Buku: Subbab 1.17.

Contoh Terurai: Biaya

Setiap peladen menghitung biaya setiap nota dengan Persamaan (1.7).

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

> **CONTOH.** B2943: 5 kg cuci, kering, dan lipat seharga Rp 8.000 menghasilkan $5 \times 8000 = 40\,000$. B2898: 2 kemeja *dry cleaning* seharga Rp 20.000 menghasilkan 40 000. B2920: 6 kg ekspres seharga Rp 12.000 menghasilkan 72 000.

Totalnya $40\,000 + 40\,000 + 72\,000 = 152\,000$, dicetak sebagai `3 bills, Rp 152,000 in total.`

Tambahkan nota B2951, satu bed cover seharga Rp 35.000, dan baris terakhir menjadi `4 bills, Rp 187,000 in total.` — tidak ada yang lain berubah, karena tabelnya dihitung dari data. Buku: Subbab 1.17 dan Soal Latihan 1.18.

Cara Pertama: Berkas Biasa

Kode 1.5 Badan `html/index.html` — halaman statis. Setiap pengunjung menerima byte yang sama, tanpa perubahan.

```
<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;amp; fold</td><td>Rp 8,000 per kg</td></tr>
    <tr><td>Express wash &amp;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>
```

Kode 1.5: sajikan dengan Live Server, `python3 -m http.server 8000`, atau `php -S localhost:8001`. Jangan klik dua kali: alamat `file://` berarti tidak ada peladen, tidak ada TCP, tidak ada HTTP. Buku: Subbab 1.17.1.

Cara Kedua: PHP Memutuskan Jawabannya

Kode 1.6 Dari `php/index.php`: metode dan path permintaan, lalu jawaban untuk metode yang salah dan untuk `/health`.

```
$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 1.6: `header(…)` dan `http_response_code(…)` harus dijalankan sebelum byte pertama badan. Menyebut `index.php` di baris perintah, `php -S localhost:8100 index.php`, menjadikannya skrip router yang menerima setiap permintaan. Buku: Subbab 1.17.2.

Cara Kedua: HTML yang Berlubang

Kode 1.7 Akhir `php/index.php`: papan nota, HTML yang disalin apa adanya dengan lubang-lubang `<?= … ?>` yang diisi pada saat permintaan.

```
<?php else:
    $total = 0;
    foreach ($bills as $b) {
        $total += $b['units'] * $b['price'];
    } ?>
  <h1>Profitina Laundry Keputih</h1>
  <p>Page built by the server at <?= date('j F Y, H:i:s') ?> WIB.</p>
  <table>
    <tr><th>Bill</th><th>Customer</th><th>Service</th><th>Units</th><th>Status</th><th>Charge</th></tr>
<?php foreach ($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'] * $b['price']) ?></td></tr>
<?php endforeach; ?>
  </table>
  <p><?= count($bills) ?> bills, <?= rupiah($total) ?> in total.</p>
```

Kode 1.7: `e()` membungkus `htmlspecialchars`, sehingga `&` dalam "Wash, dry & fold" tiba sebagai `&`. PHP tidak mengirim `Content-Length`: ia menutup koneksi untuk menandai akhir badan. Buku: Subbab 1.17.2.

Cara Ketiga: Java Mengarahkan Permintaan

Kode 1.8 Dari `route` di `java/HelloWeb.java`: setiap permintaan tiba di sini, dan satu dari lima jawaban dipilih.

```
        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");
        }
```

Kode 1.8 adalah langkah 2 dan 7 dari sembilan langkah, ditulis nyata. Peladen ini hanya memerlukan JDK: `javac HelloWeb.java`, lalu `java HelloWeb 8080`. Buku: Subbab 1.17.3.

Cara Ketiga: Menulis Respons

Kode 1.9 `send` di `java/HelloWeb.java`: baris status, header, baris kosong, badan — dengan `Content-Length` yang benar.

```
    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)");

        // 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();
    }
```

Kode 1.9: `-1` *berarti "tanpa badan", sehingga* `HEAD` *hanya mendapat header. Java menjawab* `Content-length: 1001` *ketika PHP menutup koneksi;* `/nope` *mendapat* `404`*, dan* `POST` *mendapat* `405` *dengan* `Allow: GET, HEAD`*. Buku: Subbab 1.17.3.*

Cara Keempat: Python dan Flask

Kode 1.10 Dari `python/app.py`: setiap `@app.route` memetakan pola URL ke sebuah fungsi.

```python
@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 1.10: `/bills/<bill_id>` memetakan satu keluarga URL ke satu fungsi — langkah 5 dari sembilan, dikerjakan oleh pola, bukan oleh berkas. HTML-nya adalah template Jinja, yang meng-escape setiap `{{ … }}` secara bawaan. Buku: Subbab 1.17.4.

Cara Kelima: C# dan ASP.NET Core

Kode 1.11 Dari `aspnet/Program.cs`: *minimal API* di .NET 10; setiap `MapGet` mengerjakan apa yang dikerjakan `@app.route` di Flask.

```
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);
/// <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;
}
```

Kode 1.11: `dotnet run` *membangun dan menjalankannya di Kestrel pada port 5100. Tanpa diminta lebih, nota yang tidak ada mendapat* `404` *dan* `POST` *mendapat* `405` *dengan* `Allow: GET`*, keduanya berbadan kosong. Buku: Subbab 1.17.5.*

Lima Cara, Satu Protokol

Tabel 1.12 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**

Tabel 1.12: segalanya berbeda kecuali baris terakhir, dan baris terakhir itulah intinya. Hentikan setiap peladen dengan Ctrl+C, atau peladen berikutnya di port itu gagal dengan address already in use. Buku: Subbab 1.17.6.

Profitina dalam Praktik

Pemindainya adalah klien HTTP: menerjemahkan nama, menyambung, TLS, `GET`, status, header, badan. Situs yang menjawab `500` tidak dapat dikutip mesin penjawab AI.

Apakah peladen mengirim kata-katanya, atau program yang menuliskannya? Teks yang ditulis JavaScript sisi klien hanya ada bagi klien yang menjalankan skrip.

`robots.txt`, `llms.txt`, data terstruktur — semuanya HTTP, dan semuanya harus ada di dalam respons.

Pemeriksaan kesehatan menjaga rilis, seperti rute `/health` hari ini; dan UU PDP 27/2022 berlaku sejak permintaan pertama.

Platform memberi saran; pemilik situs yang memutuskan. Studi kasus yang Anda bangun dengan papan ketik sendiri selalu Profitina Laundry. Buku: Subbab 1.18.

Kesalahan yang Sering Terjadi

“Internet” padahal maksudnya “web”; peladen web dikira mesin; berkas HTML dibuka dengan klik dua kali; berkas PHP dibuka dengan Live Server.

Mengharapkan fragment sampai di peladen; menyusun URL dari teks pengguna yang mentah.

`GET` **yang mengubah sesuatu**; memercayai validasi sisi klien; rahasia di dalam JavaScript.

`Content-Type` **yang salah**; baris kosong yang terlupa; `401` untuk `403`, atau memandangi peramban padahal log menjelaskan `500`.

Runtime alih-alih kit, terminal lama setelah PATH diubah, dua peladen di satu port.

Buku mendaftar tiga belas kesalahan, masing-masing dengan subbab yang mencegahnya. Buku: Subbab 1.19.

Ringkasan: Lima Hal untuk Diingat

1. Internet adalah jaringan dari jaringan-jaringan yang berbicara TCP/IP; web adalah layanan yang berbicara HTTP di atasnya.

2. IP mengalamati mesin dan tidak menjanjikan apa pun; DNS mengubah nama menjadi alamat; TCP (atau QUIC) membuat aliran yang andal dan berurutan.

3. Pesan HTTP terdiri atas baris pertama, header, baris kosong (`\r\n\r\n`), dan badan; `4xx` kesalahan klien, `5xx` kesalahan peladen; `Content-Type` menentukan apa badan itu.

4. Halaman statis adalah berkas, halaman dinamis adalah program; kode sisi peladen dapat menyimpan rahasia, kode sisi klien tidak dapat menyimpan satu pun.

5. Satu halaman yang disajikan dengan lima cara menghasilkan satu protokol yang sama setiap kali — pokok bahasan seluruh buku ini.

Fragment tidak pernah sampai di peladen, dan data di dalam URL di-percent-encode. Buku: Subbab 1.20.

Pertemuan Berikutnya

Bab 2 — HTML, CSS, dan Bahasa Dasar Web. Dokumen sebagai pohon, teks dan daftar, tautan, tabel, HTML5 semantik, formulir, GET dan POST di jaringan.

CSS dari aturan pertama sampai *cascade*, *box model*, flexbox, grid, dan *media query*; JavaScript, DOM, dan `fetch`.

Sebelum itu: pasang dan periksa perangkat kerjanya, buka panel Network pada situs yang sering Anda pakai, dan kerjakan Soal Latihan 1.9, 1.10, 1.11, dan 1.16.

Dokumen HTML pertama hari ini adalah titik awal Bab 2. Buku: Subbab 1.21 dan 1.22.

Kuliah 2 — HTML, CSS, dan Bahasa Dasar Web

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 2

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini kita membangun wajah publik Profitina Laundry dengan tangan — halaman harga, halaman status, dan formulir penjemputan — dan mempelajari fungsi setiap tag, aturan, dan baris skrip di dalamnya. Buku: Bab 2, Subbab 2.1–2.29.

Cakupan Pertemuan Ini

1. **HTML** — tag, elemen, dan atribut; pohon dokumen; teks, daftar, tautan, gambar, tabel; HTML5 semantis dan aksesibilitas.
2. **Formulir dan jaringan** — setiap kontrol, `name` dan `<label>`, atribut validasi; GET dan POST bita demi bita; pengodean persen; validasi di tiga tempat.
3. **CSS** — aturan dan selektor, warna dan satuan, properti kustom, *cascade* sebagai aritmetika, model kotak, *flexbox*, *grid*, dan *media query*.
4. **JavaScript** — bahasanya dalam sekali duduk, DOM, peristiwa, `fetch` dan JSON; struktur, presentasi, dan perilaku dipisahkan.
5. **Profitina Laundry** — tiga halaman publik, disajikan dan diperiksa secara identik oleh Java, Python, dan PHP.

Tidak ada yang ajaib: pada akhirnya Anda dapat menekan Ctrl+U di halaman mana pun dan membaca sumbernya. Buku: Subbab 2.1 dan 2.3.

Di Mana Kita Berhenti: Peramban Bertanya

Bab 1: **peramban bertanya, dan peladen menjawab**, dan jawabannya sebagian besar berupa teks. Hari ini tentang teks itu.

Statis: berkasnya sudah ada, dan peladen menyerahkan bitanya tanpa perubahan — tiga langkah.

CGI: peladen menjalankan program terpisah untuk setiap permintaan dan mengirim apa yang dicetaknya — lima langkah.

Skrip tertanam: berkas HTML dengan kantong-kantong kode yang dijalankan peladen — PHP, JSP, ASP.NET — empat langkah.

> **WAWASAN KUNCI.** Secerdas apa pun peladennya, ia hanya dapat mengirim apa yang dipahami peramban: HTML, CSS, JavaScript, JSON, gambar. Semua itu adalah sasaran tetap setiap teknologi sisi peladen dalam buku ini.

Panah terakhir selalu menunjuk ke peramban, dan yang melaju di atasnya selalu HTML. Buku: Subbab 2.1.

Apa Itu Bahasa Markah

> **DEFINISI.** *Markah* adalah teks yang ditambahkan pada dokumen untuk menggambarkan dokumen itu, bukan menjadi bagiannya. **Bahasa markah** adalah seperangkat aturan yang disepakati untuk menuliskan gambaran itu sehingga mesin dapat bertindak berdasarkannya.

HyperText — teks yang bertaut; **Markup Language** — gambaran sebuah dokumen. HTML dalam tiga kata: **intinya adalah tag**.

HTML tidak punya variabel, aritmetika, keputusan, atau perulangan; komputasi dimulai dengan JavaScript.

Empat tugas: menampilkan dokumen; mengatur isinya; membuat formulir; menyertakan gambar, audio, dan video.

Tulis HTML di penyunting teks polos seperti VS Code dengan Live Server, jangan pernah di Word, yang mengganti tanda petik lurus menjadi melengkung. Simpan sebagai `.html`, dalam UTF-8. Buku: Subbab 2.2 dan 2.3.

Tag, Elemen, Atribut

> **DEFINISI. Elemen** adalah tag pembuka, isinya, dan tag penutupnya, bersama-sama; **tag** hanyalah penandanya. `<p>` adalah tag; `<p>Hello</p>` adalah elemen.

```
<input type="text" id="name" name="name" maxlength="60" required>
```

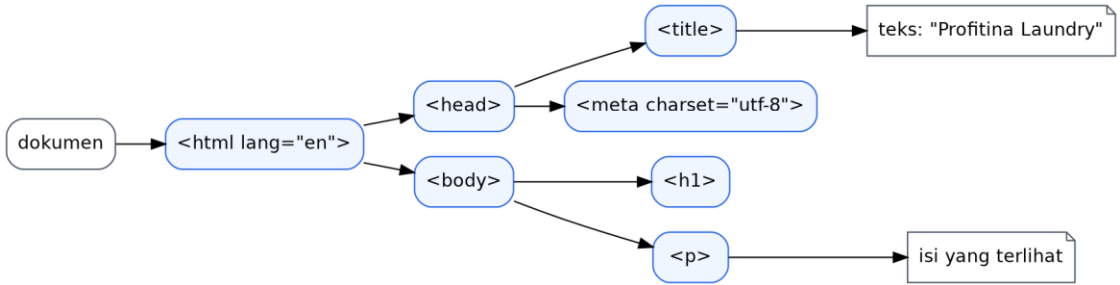
Polanya tidak pernah berubah: `<ELEMEN ATRIBUT="NILAI">`. Selalu beri tanda petik pada nilainya; `required` adalah atribut **boolean**.

Elemen void tidak punya isi dan tag penutup: `<br>`, `<img>`, `<hr>`, `<input>`, `<meta>`, `<link>`.

Nama tag tidak peka huruf besar-kecil; praktik modern menulis huruf kecil.

`<body bgcolor="blue">`: elemen `body`, atribut `bgcolor`, nilai `blue`. Buku: Subbab 2.4.1 dan 2.4.2.

Dokumen Adalah Pohon



Gambar 2.1 Dokumen HTML digambar sebagai pohon. Bagian kepala membawa informasi tentang halaman; bagian badan membawa segala yang dilihat pengunjung.

***Gambar 2.1**: setiap elemen kecuali `<html>` punya tepat satu induk, sehingga tag bersarang seperti kurung — `<p>A <b>bold word.</p>`, tidak pernah `<p>A <b>bold word.</p></b>`. Buku: Subbab 2.4.3.*

Kerangka Setiap Halaman

Kode 2.1 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>
```

`<!DOCTYPE html>` menghindari *quirks mode*; `lang` memandu pembaca layar; `charset` mencegah `é` berubah menjadi `Ã©`; *viewport* mencegah ponsel berpura-pura menjadi monitor.

***Kode 2.1**: `<title>` menjadi label tab, nama bookmark, dan judul hasil pencarian — untaian bernilai paling tinggi dalam dokumen. Buku: Subbab 2.4.4–2.4.7.*

Teks: Judul dan Paragraf

> **DEFINISI.** Elemen **tingkat blok** mengambil satu pita penuh halaman (paragraf, judul, daftar, tabel). Elemen **sebaris** duduk di dalam satu baris teks (tebal, miring, tautan, gambar).

`<h1>` sampai `<h6>`: **angkanya adalah peringkat, bukan ukuran.** Satu `<h1>` per halaman, jangan melompati tingkat ke bawah, jangan memilih tingkat karena rupanya.

HTML meringkas spasi putih: deretan spasi, tab, dan baris baru menjadi satu spasi. `<br>` mengakhiri baris dalam alamat; `<pre>` mempertahankan setiap spasi.

Pembaca layar, mesin pencari, dan mesin penjawab AI semuanya membaca kerangka judul.

Deretan `<br><br><br>` adalah ciri khas pemula: dua paragraf sudah berjarak dengan sendirinya. Buku: Subbab 2.5.1–2.5.4.

Katakan Maksud Anda

Tabel 2.1 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>`

Tag fisik — `<b>`, `<i>`, `<u>`, `<sup>`, `<sub>` — menamai rupa; tag logis menamai makna.

> **WAWASAN KUNCI.** Utamakan tag logis: suara tidak punya tinta, tetapi dapat menyampaikan kepentingan. `E=MC<sup>2</sup>` dan `H<sub>2</sub>O` adalah pemakaian tag fisik yang jujur, karena posisinya *adalah* maknanya.

***Tabel 2.1**: di layar `<strong>` tampak seperti `<b>`, tetapi satu aturan `strong { }` kelak dapat mengubah gaya setiap kata penting. Buku: Subbab 2.5.5 dan 2.5.6.*

Daftar

Kode 2.2 Badan `html/basics/list-nested.html`: `<ul>` bagian dalam tinggal di dalam sebuah `<li>`, tidak pernah di antara dua `<li>`.

```
  <h1>Our services</h1>
  <ul>
    <li>By weight
      <ul>
        <li>Wash, dry &amp;amp; fold</li>
        <li>Express wash &amp;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>
```

***Kode 2.2**: `<ul>` bila urutan tidak penting, `<ol>` bila penting — peramban yang menomori, jadi butir sisipan tidak membuat sisanya salah — dan `<dl>` dengan `<dt>` dan `<dd>` untuk pasangan nama dan nilai. Buku: Subbab 2.6.*

Tautan dan Alamat Relatif

Tabel 2.2 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

`<a href="#prices">` melompat ke `<h2 id="prices">` di dokumen yang sama.

Teks tautan menggambarkan tujuannya (jangan pernah *klik di sini*); `tel:+623100002101` menjadi ketuk-untuk-menelepon; jangan pernah menghilangkan `href`.

***Tabel 2.2**: ketika tautan tidak bekerja, berkasnya hampir selalu tidak berada di tempat yang Anda kira, relatif terhadap halaman yang menyebutnya. Buku: Subbab 2.7.*

Gambar: `alt`, `width`, `height`

Kode 2.3 Dari `html/basics/image.html`: logo resmi dengan teks alternatif dan dimensinya.

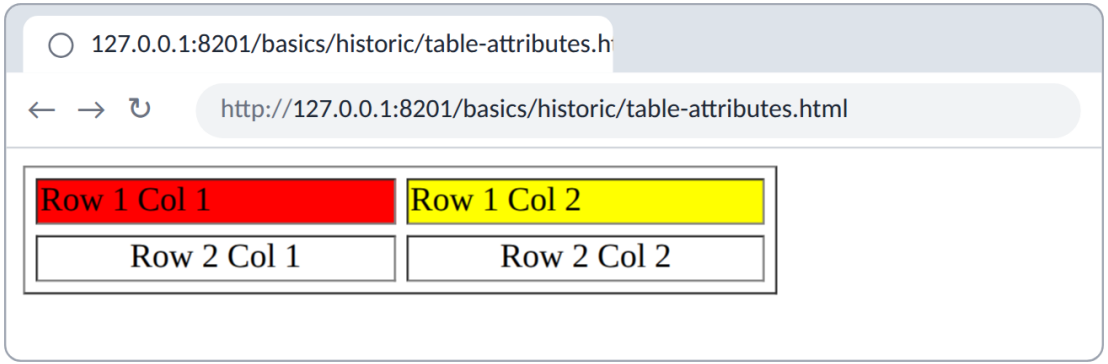
```
<img src="../assets/profitina-logo-horizontal.png" alt="Profitina" width="236" height="68">
```

> `alt` **TIDAK OPSIONAL.** Tuliskan apa yang *disampaikan* gambar (`alt="Profitina"`), dan `alt=""` untuk hiasan belaka. Tanpa atribut itu pembaca layar membacakan nama berkasnya.

`width` dan `height` **mencadangkan ruang**, sehingga halaman tidak melompat ketika gambar tiba. JPEG untuk foto, PNG untuk tepi tajam, SVG untuk ikon, WebP dan AVIF untuk foto yang lebih kecil.

***Kode 2.3**: `src` adalah alamat, relatif atau absolut, persis seperti `href`. Buku: Subbab 2.8.*

HTML Tidak Pernah Mengeluh



Gambar 2.2 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.

***Gambar 2.2**: `cellpading` hanya ber-D satu, jadi diabaikan diam-diam. Pemeriksa W3C (`vnu.jar`) menemukannya dalam sedetik. Buku: Subbab 2.9.3 dan 2.14.1.*

Tabel, Ditulis Masa Kini

Kode 2.4 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;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>
```

***Kode 2.4**: `<th scope="col">` dan `scope="row"` membuat pembaca layar menyebut `12.000` sebagai Price dari Express, 1 day; `<thead>` diulang di setiap halaman cetak; `colspan="3"` merentang tiga kolom. Tabel untuk data, tidak pernah untuk tata letak. Buku: Subbab 2.9.4.*

HTML5 Semantis

Kode 2.5 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>
```

Halaman yang sama yang dibangun dari `<div class="header">`, `<div class="nav">`, dan `<div class="text">` dapat tampak identik dan tidak bermakna apa pun.

***Kode 2.5**: pembaca layar langsung melompat ke `<main>`, dan mesin penjawab menemukan judul, paragraf, dan harganya. Pertanyaan penentunya: tanpa CSS sama sekali, dapatkah orang asing mengenali setiap bagian? Buku: Subbab 2.10.*

Aksesibilitas sejak Baris Pertama

`lang`, satu `<h1>`, *landmark*, tautan lompat, `alt`, `<label>` untuk setiap kontrol, cincin fokus yang terlihat, kontrol setinggi 44 piksel — masing-masing seharga satu atribut atau satu aturan.

Aturan pertama **ARIA**: jangan memakainya bila HTML biasa sudah mengatakannya. `<button>` adalah tombol; `<div role="button">` bukan tombol bagi papan ketik.

WCAG 2.2 meminta kontras teks paling sedikit 4,5 banding 1, menurut Persamaan (2.1).

$$\text{kontras} = \frac{L_{\text{terang}} + 0.05}{L_{\text{gelap}} + 0.05} \tag{2.1}$$

CONTOH TERURAI. Hitam di atas putih: $(1 + 0.05)/(0 + 0.05) = 21$. Abu-abu petunjuk kita `#4b5563`, $L \approx 0.089$: $1.05/0.139 \approx 7.6$, lolos. `#9ca3af`, $L \approx 0.364$: $1.05/0.414 \approx 2.5$, cocok untuk garis tepi, tak terbaca untuk teks.

Ujilah sebagaimana dipakai: singkirkan tetikus dan pakai Tab, Enter, dan Spasi; perbesar sampai 200 persen; dengarkan dengan Narrator, VoiceOver, atau Orca. Buku: Subbab 2.11.

Formulir: `action`, `method`, `name`

```
<form action="pickup" method="post">
  <input type="text" name="phone">
  <button type="submit">Send</button>
</form>
```

`action` — ke mana dikirim; `method` — `get` atau `post`; `enctype` — cara mengemas (berkas memerlukan `multipart/form-data`).

`name` **adalah intinya:** kontrol tanpa `name` tidak dikirim sama sekali, tanpa galat. `id` untuk CSS, JavaScript, dan `<label for>`.

`<label for>` menyebut `id` kontrolnya: sasaran ketuk yang lebih besar dan nama yang diucapkan. `placeholder` adalah petunjuk, tidak pernah label.

Formulir di dalam formulir lain dibuang diam-diam; `<fieldset>` dan `<legend>` memberi nama pada sekelompok kontrol. Buku: Subbab 2.12.1–2.12.4.

Jenis Input yang Mengejutkan

Tabel 2.3 Pilihan jenis `<input>`, kontrol yang dihasilkan masing-masing, dan apa yang diterima peladen.

type	Yang dilihat pengunjung	Yang dikirim
`text`	kotak satu baris	apa pun yang diketik
`email`	kotak teks; ponsel menampilkan papan ketik @	teksnya, diperiksa bentuk alamatnya
`date`	pemilih tanggal	`YYYY-MM-DD`
`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

***Tabel 2.3**: kotak centang yang tidak dicentang **tidak mengirim apa pun** — bukan string kosong; tombol radio dikelompokkan oleh `name`; field tersembunyi tidak terlihat, bukan rahasia. Buku: Subbab 2.12.5.*

Daftar Tarik-Turun Berkelompok

Kode 2.6 Dari `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;amp; fold - Rp 8.000/kg</option>
            <option value="S0002" data-price="12000">Express wash &amp;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>
          </optgroup>
        </select>
```

***Kode 2.6**: peladen menerima `S0001`, kunci tabel Service; opsi pertama ber-`value=""`, sehingga dengan `required` pengunjung wajib memilih; `data-price` disediakan untuk skrip. Selalu tulis `type` pada `<button>`. Buku: Subbab 2.12.6.*

Atribut yang Memvalidasi

Tabel 2.4 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

***Tabel 2.4**: `readonly` ikut dikirim, `disabled` tidak. Field telepon memakai `pattern="0[0-9 \-]{9,16}"`: sebuah nol, lalu sembilan sampai enam belas digit, spasi, atau tanda hubung. Buku: Subbab 2.12.7.*

Persetujuan Hanya Satu Baris HTML

Kode 2.7 *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>
```

***Kode 2.7**: UU PDP No. 27/2022 meminta persetujuan yang spesifik, berdasarkan informasi, dan bebas — kotak yang dicentang pelanggan setelah membaca apa yang disimpan dan mengapa, tidak pernah kotak yang sudah tercentang. Buku: Subbab 2.12.8.*

Contoh Terurai: Yang Dikirim Siti

> **CONTOH TERURAI.** Siti mengisi formulir penjemputan: nama, nomor, dan alamatnya, Keputih, *Wash, dry & fold*, 4,3 kg, *No perfume* dan *Shirts on hangers*, 2 Oktober 2026 pukul 09.30, sebuah catatan, persetujuan dicentang.

Peramban mengirim **daftar datar pasangan nama dan nilai**, bukan sebuah formulir.

`care` muncul **dua kali**, satu untuk setiap kotak yang dicentang.

`email` muncul sebagai string kosong, karena kotak teks kosong tetap mengirim namanya.

`consent=yes` — dan, bila tidak dicentang, pasangan itu sama sekali tidak ada.

Kelompok radio menjadi wajib lewat `required` pada tombol pertamanya, dan `step="0.1"` menerima 4,3 tetapi tidak 4,35. Buku: Subbab 2.12.8 dan 2.13.3.

GET Sungguhan, di Jaringan

Kode 2.8 Dari `captures/request-1.txt`: baris permintaan dan sebagian header sebuah GET sungguhan yang dikirim formulir status.

```
GET /status?bill=B2938&phone=0896+0000+1482 HTTP/1.1
Host: localhost:8155
Connection: keep-alive
Referer: http://localhost:8155/status.html
Accept-Encoding: gzip, deflate, br, zstd
```

`GET` adalah metodenya; `/status` adalah `action` formulir; semua setelah `?` adalah **query string** — data formulir.

Pasangan dihubungkan dengan `&`, nama dan nilai dengan `=`, dan spasi dalam nomor telepon menjadi `+`. Badan sebuah GET kosong.

***Kode 2.8**: tangkapan ini dibuat oleh `tools/capture.py`, peladen mungil di atas soket mentah yang mencetak setiap permintaan persis seperti tibanya; header yang dihilangkan adalah petunjuk peramban. Buku: Subbab 2.13.1.*

Pengodean Persen, Dikerjakan dengan Tangan

DEFINISI. Pengodean persen mengganti setiap bita yang bukan huruf, digit, atau salah satu dari `-`, `.`, `_`, `~` dengan `%` dan dua digit heksadesimal. Dalam data formulir, spasi boleh ditulis `+`.

$$v = 16h + \ell, \qquad h = \lfloor v/16 \rfloor, \quad \ell = v - 16h \tag{2.2}$$

CONTOH TERURAI. `#` adalah bita $35 = 2 \times 16 + 3$, jadi `%23`. Koma adalah $44 = 2 \times 16 + 12$, dan 12 adalah `C`: `%2C`. Titik dua adalah $58 = 3 \times 16 + 10$: `%3A`, sehingga `09:30` berangkat sebagai `09%3A30`.

Menurut Persamaan (2.2), `+` harfiah harus dikirim sebagai `%2B`, atau pencarian `C++` menjadi `C` dan dua spasi; `é` adalah dua bita UTF-8, `%C3%A9`. Buku: Subbab 2.13.2.

POST Sungguhan, di Jaringan

Kode 2.9 Dari `captures/request-2.txt`: formulir penjemputan yang dikirim dengan POST. Datanya pindah ke badan; formatnya tidak berubah.

```
POST /pickup HTTP/1.1
Content-Length: 258
Origin: http://localhost:8155
Content-Type: application/x-www-form-urlencoded

name=Siti+Rahmawati&phone=0812+0000+1234&email=&address=Jl.+Keputih+Tegal+Timur+12%2C+Sukolilo&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
```

***Kode 2.9**: `Content-Type` menyatakan cara membaca badan dan `Content-Length: 258` berapa bita yang dibaca; `Origin` menyebut situs formulirnya. Koma tiba sebagai `%2C`, `#3` sebagai `%233`, dan `care` dua kali. Buku: Subbab 2.13.3.*

Peladen Menerima Pasangan

> **WAWASAN KUNCI.** Sebuah nama bisa muncul nol kali, sekali, atau berkali-kali, sehingga setiap perkakas menawarkan dua pembaca: nilai pertama, dan semua nilai.

Java Servlet: `getParameter` dan `getParameterValues`.

Flask: `request.form.get` dan `request.form.getlist`.

PHP: `$_POST` hanya menyimpan nilai *terakhir* kecuali HTML menamai *field*-nya `care[]`.

Berkas: `enctype="multipart/form-data"` mengirim bagian-bagian yang dipisahkan oleh **boundary** acak, masing-masing dengan header dan bita mentahnya sendiri.

Memakai pembaca tunggal pada kelompok kotak centang adalah bug yang baru muncul ketika pengunjung mencentang dua kotak. Buku: Subbab 2.13.3 dan 2.13.4.

GET atau POST?

Tabel 2.5 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

***Tabel 2.5**: GET untuk membaca, POST untuk mengubah. Pertanyaan yang tidak pernah menyesatkan: apakah berbahaya bila dilakukan dua kali? Dan jangan pernah menaruh kata sandi dalam GET. Buku: Subbab 2.13.5.*

Validasi: Tiga Tempat, Tiga Tugas

Di HTML — `required`, `pattern`, `min`, `type="email"`: gratis, bekerja tanpa JavaScript, tetapi berjalan di mesin pengunjung.

Di JavaScript — setiap *field* yang salah ditandai sekaligus, dengan kata-kata kita, dibacakan pembaca layar; dapat dimatikan dalam sedetik.

Di peladen — satu-satunya pemeriksaan yang nyata.

> **PERINGATAN TERPENTING BAB INI.** Validasi sisi klien adalah kemudahan bagi pengunjung yang jujur. Siapa pun dapat mengirim permintaan yang tidak pernah melewati formulir, dengan `curl` atau dua puluh baris Python. Peladen mengulangi setiap pemeriksaan, pada setiap permintaan.

Dalam satu baris: pemeriksaan klien menghentikan kekeliruan; pemeriksaan peladen menghentikan serangan. Buku: Subbab 2.14.

CSS: Aturan, dan Tempat Tinggalnya

```
hr { color: red; height: 5px; width: 50%; }
```

Selektor `hr`; **deklarasi** `properti: nilai;` dipisahkan titik koma. Tulis juga titik koma terakhirnya.

Inline — `style="…"` pada satu tag: hampir tidak pernah. **Internal** — blok `<style>` di `<head>`: satu halaman.

Eksternal — `<link rel="stylesheet" href="css/style.css">`: satu berkas untuk semua halaman, diunduh sekali dan disimpan di cache. Inilah yang dipakai.

CSS hanya punya satu bentuk komentar, `/* … */`; `//` tidak bekerja. `h1, h2, h3, h4 { color: blue; }` memberi empat selektor satu deklarasi. Buku: Subbab 2.15.1 dan 2.15.2.

Selektor yang Akan Anda Pakai

Tabel 2.6 Selektor CSS yang paling sering dipakai. 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`
`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>`
`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
`tr:nth-child(even)`	setiap baris kedua

***Tabel 2.6**: `table.receipt` adalah tabel berkelas itu, `table .receipt` adalah apa pun berkelas itu di dalam tabel. Buku: Subbab 2.15.3.*

`id`, `class`, dan *Bug* yang Diam

stylecss04.html

```
<!-- stylecss04.html -->
<HTML>
  <HEAD>
    <TITLE>Profitina Laundry</TITLE>
    <STYLE TYPE="TEXT/CSS">
      #TOP {
        BACKGROUND-COLOR: #FF9;
        PADDING: 1em;
        TEXT-ALIGN: CENTER;
      }
      .MyLine {
        COLOR: BLUE;
        FRONT-WEIGHT: BOLD;
        MARGIN: 0;
        PADDING: 0;
      }
    </STYLE>
  </HEAD>
  <BODY>
    <DIV ID="TOP">
      <H1>Profitina Laundry</H1>
      <P CLASS="MyLine">Clean laundry, three outlets in Surabaya</P>
      <P CLASS="MyLine">Open every day, 07:00 to 21:00</P>
    </DIV>
    <P>Welcome to our website!</P>
  </BODY>
</HTML>
```

Gambar 2.3 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 2.3**: `#` memilih satu `id` yang unik, `.` sejumlah elemen berkelas sama. `FRONT-WEIGHT: BOLD` bukan properti, jadi dibuang tanpa sepatah kata. Utamakan kelas. Buku: Subbab 2.15.4.*

Warna dan Panjang

`#RRGGBB` adalah tiga bita, masing-masing `00` sampai `FF`; `#FF9` menggandakan setiap digit menjadi `#FFFF99`; `rgb(37 99 235 / 0.35)` menambahkan transparansi.

> **CONTOH TERURAI.** Biru tombol `#2563eb`: merah `25` $= 2 \times 16 + 5 = 37$, hijau `63` $= 6 \times 16 + 3 = 99$, biru `eb` $= 14 \times 16 + 11 = 235$. Sedikit merah, sebagian hijau, biru hampir penuh.

`rem` adalah ukuran fon akar dan tidak pernah berlipat; `em` adalah ukuran elemen ini, sehingga tiga daftar bersarang ber-`0.9em` menjadi $0.9^3 \approx 0.73$.

Utamakan `rem` untuk huruf dan jarak, `px` untuk garis tepi, `%` dan `fr` untuk tata letak.

216 warna web-safe, yang digitnya diambil dari 0, 3, 6, 9, C, dan F, milik layar lama dan lembar gaya lama. Buku: Subbab 2.15.5 dan 2.15.6.

Properti Kustom: Menamai Desain Sekali

DEFINISI. Properti kustom diawali dua tanda hubung, biasanya dideklarasikan pada `:root`, dan dibaca dengan `var(--blue)`; `var(--space, 1rem)` memberi nilai cadangan.

Kode 2.10 Dari `html/css/style.css`: tema gelap yang hanya mengubah token desain.

```
@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;
  }
}
```

Kode 2.10*: bagian lain lembar gaya tidak pernah menyebut warna secara langsung — `body { color: var(--ink); background: var(--bg); }` — sehingga delapan nilai token baru mengubah tema seluruh situs. Buku: Subbab 2.16.*

Spesifisitas Adalah Aritmetika

> **DEFINISI. Spesifisitas** sebuah selektor adalah (a, b, c): a selektor `id`; b kelas, selektor atribut, dan pseudo-kelas; c nama elemen dan pseudo-elemen. Bandingkan dari kiri ke kanan.

Untuk setiap selektor yang ditulis manusia, Persamaan (2.3) memberi urutan yang benar.

$$\text{skor} = 100a + 10b + c \tag{2.3}$$

Tahap 1 asal dan `!important`; **tahap 2** spesifisitas; **tahap 3** aturan yang lebih belakang menang bila seri; **tahap 4** pewarisan (properti teks diwariskan, properti kotak tidak).

Sebenarnya bukan penjumlahan: satu `id` mengalahkan berapa pun banyaknya kelas, karena perbandingannya per kolom. `!important` adalah jalan terakhir. Buku: Subbab 2.17.1 dan 2.17.2.

Spesifisitas Selektor Nyata

Tabel 2.7 Spesifisitas yang dihitung untuk selektor nyata dari lembar gaya. Hitung `id`, lalu kelas, atribut, dan pseudo-kelas, 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
`.card`	0	1	0	(0,1,0)	10
`.site-menu a`	0	1	1	(0,1,1)	11
`#price-table`	1	0	0	(1,0,0)	100
`#price-table tbody th`	1	0	2	(1,0,2)	102
atribut `style`	—	—	—	—	mengalahkan semua

***Tabel 2.7**: satu `id` bernilai 100 mengalahkan sembilan kelas bernilai 90; atribut `style` mengalahkan setiap selektor; `*` bernilai nol, persis yang diinginkan sebuah reset. Buku: Subbab 2.17.2.*

Contoh Terurai: Warna Mana yang Menang?

Kode 2.11 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 */
```

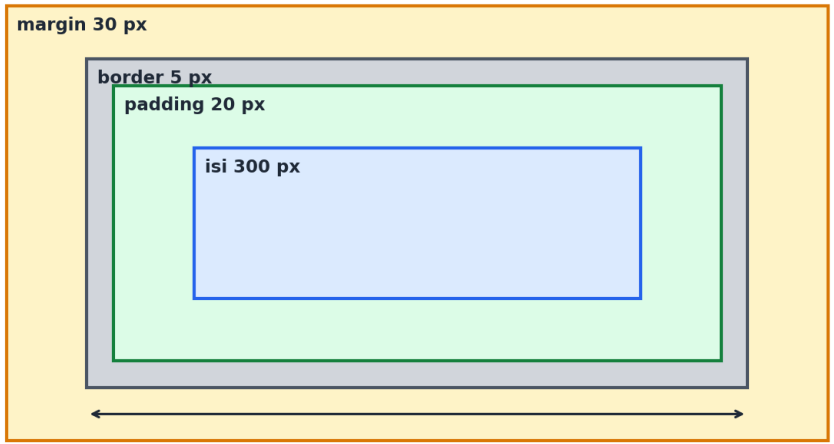
Tautannya `<a href="#" aria-current="page">Order</a>` di dalam `<nav class="site-nav">`. Skornya: 1, 11, 4, 21.

Aturan keempat menang dengan 21: `#1d4ed8`. Chromium menjawab `rgb(29, 78, 216)` — $29 = 1 \times 16 + 13$, $78 = 4 \times 16 + 14$, $216 = 13 \times 16 + 8$.

> **WAWASAN KUNCI.** Aturan yang “tidak bekerja” biasanya sudah bekerja lalu dikalahkan. Panel Styles mencoret yang kalah: itulah *cascade*, digambar.

***Kode 2.11**: `nav ul li a`, empat elemen, kalah dari `.site-nav a`, karena b dibandingkan sebelum c. Buku: Subbab 2.17.3.*

Model Kotak



Gambar 2.4 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*.

***Gambar 2.4**: isi, lalu padding (di dalam, berwarna latar), garis tepi, dan margin (di luar, transparan). Buku: Subbab 2.18.*

Contoh Terurai: 300 Piksel Menjadi 410

Kode 2.12 Dari `html/basics/box-model.css`: satu kartu, dengan lebar, *padding*, garis tepi, dan *margin*.

```
.card {
  width: 300px;
  padding: 20px;
  border: 5px solid #d1d5db;
  margin: 30px;
}
```

$$\text{lebar terlihat} = w + 2p + 2b = 300 + 2 \times 20 + 2 \times 5 = 350 \text{ px} \tag{2.4}$$

Dengan *margin*: $350 + 2 \times 30 = 410$ px terpakai. Dengan `box-sizing: border-box` kartunya tepat 300 px dan isinya menyusut menjadi $300 - 40 - 10 = 250$.

Kode 2.12 *dan Persamaan (2.4): Chromium mengukur kedua kartu 350 dan 300 piksel. Tulis* `*, *::before, *::after { box-sizing: border-box; }` *sekali; margin vertikal antarblok melebur menjadi yang terbesar. Buku: Subbab 2.18.*

Flexbox: Satu Dimensi

Kode 2.13 Dari `html/css/style.css`: kepala halaman dan menu yang ditata dengan *flexbox*.

```
/* ---- 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);
}
.site-menu ul { display: flex; flex-wrap: wrap; gap: 0.25rem; list-style: none;
                margin: 0; padding: 0; }
```

***Kode 2.13**: `justify-content` bekerja sepanjang baris, `align-items` melintanginya; `flex-wrap: wrap` membuat menu turun ke bawah logo di ponsel; tiga deklarasi mengubah `<ul>` menjadi menu mendatar. Buku: Subbab 2.19.1.*

Grid, dan *Mobile First*

Grid menata dua dimensi: `grid-template-columns: repeat(auto-fit, minmax(14rem, 1fr));` membuat kolom sebanyak yang muat, masing-masing paling sedikit 14 rem.

Kode 2.14 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 2.14 *adalah* ***mobile first****: tata letak sempit tidak memerlukan query, dan query-nya menambahkan. Buku: Subbab 2.19.2 dan 2.20.*

JavaScript: Perilaku

Tabel 2.8 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

Ia menanggapi peristiwa, membaca dan mengubah halaman melalui DOM, memvalidasi, berbicara dengan peladen lewat `fetch`, dan mengingat sesuatu di peramban.

***Tabel 2.8**: ditulis dalam sepuluh hari pada 1995 dan dibakukan sebagai ECMAScript. Sumbernya terlihat oleh siapa pun, sehingga tidak pernah dapat dipercaya — Subbab 2.14 dari sisi lain. Buku: Subbab 2.21 dan 2.21.1.*

Bahasanya dalam Sekali Duduk

Kode 2.15 Dari `html/basics/js-basics.js`: `const`, `let`, *template literal*, dan empat kejutan klasik. Jalankan dengan `node js-basics.js`.

```
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
```

```
string number boolean undefined
Profitina Laundry: 5 kg x 8000 = 40000
53 15 false 0.30000000000000004
```

***Kode 2.15**: selalu pakai `===`; `0.1 + 0.2` bukan `0.3`, sehingga uang dihitung dalam rupiah bulat. Buku: Subbab 2.21.2.*

DOM, Ditelusuri

Kode 2.16 Dari `html/basics/dom-walk.html`: penelusuran rekursif yang menulis satu baris per elemen.

```
    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");
```

```
html
  head
    meta
    title
  body
    h1
    p
      a
    pre
    script
```

***Kode 2.16**: pohon dokumen yang dimiringkan, `<script>` ikut di dalamnya. Buku: Subbab 2.22.*

Mengubah Halaman dengan Aman

Kode 2.17 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);
}
```

> **JANGAN `innerHTML` DENGAN DATA.** Deskripsi seperti `<img src=x onerror=…>` akan menjadi kode hidup: *cross-site scripting*. Buat elemen dan isi `textContent`.

Kode 2.17 *memakai kelima operasi: menemukan (`querySelector`), teks (`textContent`), atribut (`className`, `scope`), membuat (`createElement`), menyisipkan (`append`, `replaceChildren`). Buku: Subbab 2.22.1.*

Peristiwa

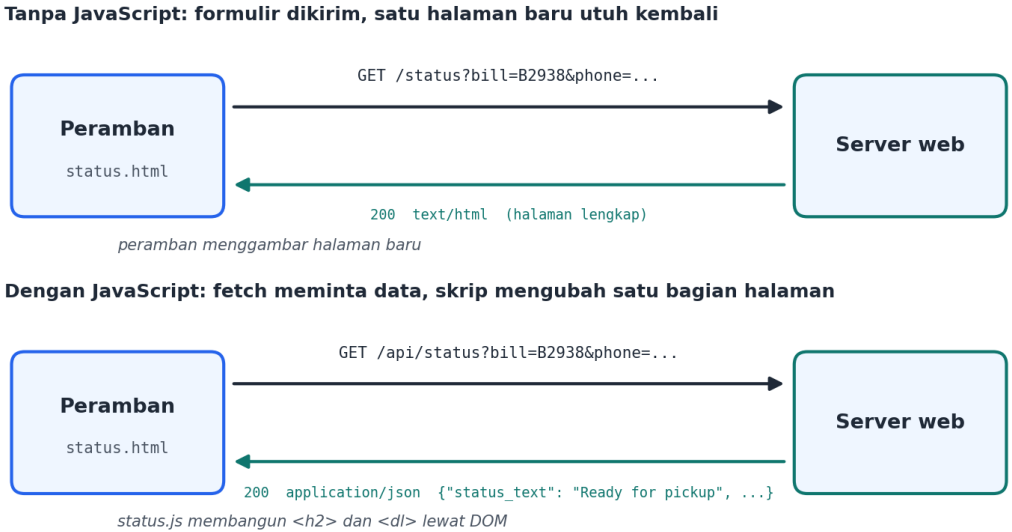
```
form.addEventListener("input", update);
```

Kode 2.18 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);
  }
```

***Kode 2.18**: peristiwa **menggelembung**, sehingga satu pendengar pada seluruh formulir mendengar kedua kontrol; 4,3 kg menjadi `5 x Rp 8.000 = Rp 40.000`. `event.preventDefault()` membatalkan apa yang akan dilakukan peramban berikutnya. Buku: Subbab 2.22.2.*

Formulir, atau `fetch`?



Gambar 2.5 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.

***Gambar 2.5**: formulir memuat halaman baru seutuhnya; `fetch` mengambil JSON, dan skrip mengubah satu bagian halaman yang sedang tampil. Buku: Subbab 2.23.*

`fetch`, JSON, dan Cadangan

Kode 2.19 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
}
```

***Kode 2.19**: `await` menunggu tanpa membekukan halaman; `response.ok` benar untuk 200 sampai 299; bila kedua alamat gagal, tabel yang tertulis di HTML tetap ada. Halaman bekerja tanpa skrip — **progressive enhancement**. Buku: Subbab 2.23.*

Status Galat Juga Sebuah Jawaban

Kode 2.20 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).");
  }
}
```

***Kode 2.20**: `fetch` hanya melempar galat bila tidak ada jawaban sama sekali; 404 membawa `{"error": {"code": "not_found", …}}` dan halaman menampilkan pesan peladen. Jawabannya tidak memuat nama atau alamat: minimisasi data, diputuskan di peladen. Buku: Subbab 2.23.*

Struktur, Presentasi, Perilaku

Versi 1 (1999): `<font>`, `<center>`, tabel tata letak, `Arial` diketik empat kali, dan efek *hover* lewat dua atribut JavaScript.

Versi 2: elemen semantis, seluruh rupa dalam satu blok `<style>`, *hover* dengan `a:hover`.

Versi 3: struktur di HTML, presentasi di `home.css`, perilaku di `home.js`.

Kode 2.21 Dari `html/steps/home.js`: tarifnya dibaca dari `data-price` di halaman, sehingga angka 8000 hanya ada satu.

```
// 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();
```

***Kode 2.21**: hapus `home.js` dan halaman tetap terbaca sempurna; hapus juga `home.css` dan ia tetap dokumen dengan judul, daftar, dan tautan. Setiap lapisan merosot menjadi sesuatu yang berguna. Buku: Subbab 2.24.*

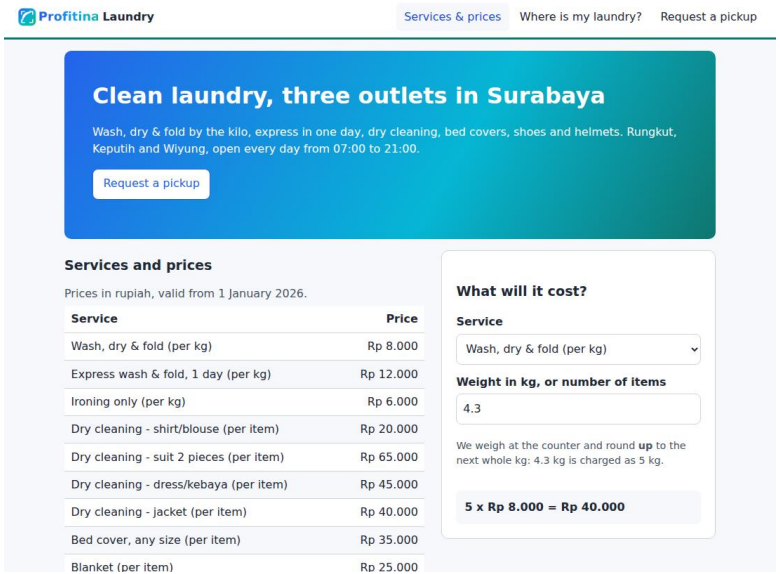
Halaman Publik Profitina Laundry

Tabel 2.9 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)

***Tabel 2.9**: tujuh langkah — kerangka dan kepala, isi dalam HTML saja, lembar gaya, perilaku, halaman status, formulir penjemputan, dan pengujian dengan Node.js (`node --test`), sembilan dari sembilan lolos. Buku: Subbab 2.25.*

Halaman Harga yang Sudah Jadi



Gambar 2.6 Halaman harga yang sudah jadi. Tabelnya berasal dari `GET /api/services`, dan kalkulatornya sudah menghargai 4,3 kilogram sebagai lima.

***Gambar 2.6**: tabelnya berasal dari `GET /api/services`; tanpa lembar gaya dan skrip, halamannya tetap daftar harga yang lengkap. Buku: Subbab 2.25.*

Menghentikan Permintaan Penjemputan yang Salah

Kode 2.22 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();
    }
  });
```

***Kode 2.22**: `values()` membaca kelompok kotak centang dengan `getAll("care")`; setiap pesan masuk ke span galat field-nya dengan `aria-invalid`; `preventDefault()` menghentikan permintaan dan field salah pertama mendapat fokus. Buku: Subbab 2.25.*

Tiga Peladen, Satu Pengurai

Java `java LaundryServer` (8080), Flask `python app.py` (5000), PHP `php -S localhost:8000 -t ../html router.php` (8000): halaman yang sama, jawaban yang sama.

Kode 2.23 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 2.23 *adalah Subbab 2.13 yang dijadikan kode: nilai boleh memuat `=`, dan setiap nama menyimpan sebuah **list**. PHP melakukan hal yang sama dengan `explode`, `urldecode`, dan `$out[$k][] = $v`. Buku: Subbab 2.26.1 dan 2.26.3.*

Flask Mengerjakan Penguraiannya

Kode 2.24 Dari `python/app.py`: pemeriksaan status JSON, dengan jawaban `400` dan `404`.

```
@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))
```

Kode 2.25 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
```

***Kode 2.24**: `request.args` adalah query yang sudah diurai. **Kode 2.25**: `getlist`, lalu hanya nilai yang dikenal — `care=free` dari `curl` dibuang. Penjemputan yang salah mendapat `422`. Satu uji asap mengirim dua belas permintaan; ketiga peladen lolos semuanya. Buku: Subbab 2.26.2 dan 2.26.4.*

Profitina dalam Praktik

Struktur adalah sinyalnya: dari tumpukan `<div>` hanya sedikit yang dapat diekstrak; dari `<main>`, judul, dan paragraf mesin penjawab dapat mengutip.

Halaman harus bekerja sebelum skrip berjalan: harga yang baru ada setelah JavaScript berjalan bisa tak terlihat oleh *crawler*.

Keterekstrakan beririsan dengan aksesibilitas: `alt`, satu `<h1>`, judul berurutan, `<th scope>` — apa yang dapat diurai pembaca layar dapat diurai model bahasa.

Data terstruktur juga markah: `schema.org` dan `robots.txt`, dirancang oleh model yang berjalan di peladen Profitina sendiri.

Studi kasus yang dapat dijalankan selalu Profitina Laundry. Buku: Subbab 2.27.

Kesalahan yang Sering Terjadi

Kontrol tanpa `name`; GET yang mengubah data; pembaca tunggal untuk kelompok kotak centang; mengharapkan kotak tak tercentang mengirim `""`.

Memercayai validasi peramban; teks pengunjung di halaman tanpa di-*escape*; `innerHTML` dengan data.

Judul dipilih karena ukurannya; tabel untuk tata letak; `<div>` padahal ada elemen yang tepat; satu `id` dua kali.

Tanpa `border-box`; `!important` alih-alih membaca spesifisitas; satu warna diketik dua belas kali.

`==` di JavaScript; lupa bahwa `fetch` tidak gagal pada 404; tidak pernah menjalankan validator.

Buku mendaftar enam belas kesalahan, masing-masing dengan cara memperbaikinya. Buku: Subbab 2.28.

Ringkasan: Lima Hal untuk Diingat

1. HTML menggambarkan *apa* sesuatu itu; halaman adalah pohon yang dimulai dengan `<!DOCTYPE html>`, `lang`, `charset`, *viewport*, dan `<title>` yang sungguhan.

2. Formulir mengirim pasangan nama dan nilai: tanpa `name`, tanpa nilai; GET membaca, POST mengubah; keduanya memakai pengodean persen (Persamaan (2.2)).

3. Validasi di HTML, di JavaScript, dan di peladen — satu-satunya pemeriksaan yang dapat dipercaya.

4. Konflik CSS diselesaikan oleh kepentingan, spesifisitas (Persamaan (2.3)), lalu urutan; setiap elemen adalah kotak (Persamaan (2.4)); *flexbox*, *grid*, dan *media query* menata letak, *mobile first*.

5. JavaScript mengubah halaman melalui DOM, dengan `textContent`, dan berbicara dengan peladen lewat `fetch` — halaman tetap bekerja tanpanya.

Struktur di HTML, presentasi di CSS, perilaku di JavaScript. Buku: Subbab 2.29.

Pertemuan Berikutnya

Bab 3 — PHP dari Nol. Berkas dengan kantong-kantong kode, variabel dan tipe, operator, keputusan dan perulangan, fungsi, *array*, dan string.

Superglobal, formulir, validasi dan *escaping*, cookie dan sesi, kata sandi, unggahan berkas — dengan halaman yang sama di Java dan Flask.

Sebelum itu: kerjakan Soal Latihan 2.3, 2.8, 2.9, 2.11, dan 2.18, dan jalankan pemeriksa W3C pada halaman buatan Anda sendiri.

Segala sesuatu sejak Bab 3 adalah kode yang membaca pasangan yang kita kirim hari ini. Buku: Subbab 2.30 dan 2.31.

Kuliah 3 — PHP dari Nol

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 3

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini peladen mulai berpikir: kita menulis PHP 8.4 dari baris pertama, membaca apa yang dikirim pengunjung, menolak yang buruk, dan mengingat pengunjung di antara permintaan — setiap gagasan web ditunjukkan dalam PHP, Java, dan Flask. Buku: Bab 3, Subbab 3.1–3.27.

Cakupan Pertemuan Ini

1. **Gagasan sisi peladen** — program di tengah jalan permintaan; Hello World dari tiga peladen.
2. **Bahasanya** — bentuk berkas, variabel dan tipe, *juggling* dan `===`, operator dan presedens, keputusan, perulangan, fungsi, *array*, *string*, `require_once`.
3. **Permintaannya** — *superglobal*, GET dan POST, formulir, dan kalkulator harga dalam PHP, Java, dan Flask.
4. **Disiplinnya** — validasi dan *escaping*, *cookie* dan sesi, token formulir, *post/redirect/get*, *hash* kata sandi, unggahan berkas, membaca galat.
5. **Profitina Laundry** — kalkulator, formulir kontak, dan halaman staf yang membaca pesan.

Subbab 3.19, validasi dan escaping, adalah bagian terpenting hari ini. Buku: Subbab 3.1.

Sampai di Mana Kita: Sama bagi Semua Orang

Bab 1: **peramban bertanya, peladen menjawab**. Bab 2: HTML dan CSS yang layak digambar.

Namun setiap halaman **sama bagi semua orang**: harga baru berarti menyunting berkas, dan formulir yang dikirim tidak sampai ke apa pun yang mendengarkan.

Halaman statis dapat berkata “tarif kami Rp 8.000 per kilogram”. Ia tidak dapat berkata “Sri, empat kilogram akan dikenai Rp 32.000”.

Tiga hal yang hilang: **tempat untuk logika**, **cara menerima apa yang diketik pengunjung**, dan **disiplin yang harus menyertainya**.

Begitu orang asing dapat mengirim data ke dalam program, anggapan-anggapan program itu menjadi sasaran. Buku: Subbab 3.1.

Program di Tengah Jalan

DEFINISI. Halaman statis adalah halaman yang bitanya sudah ditentukan sebelum ada yang memintanya: berkas di disk dan badan respons adalah hal yang sama.

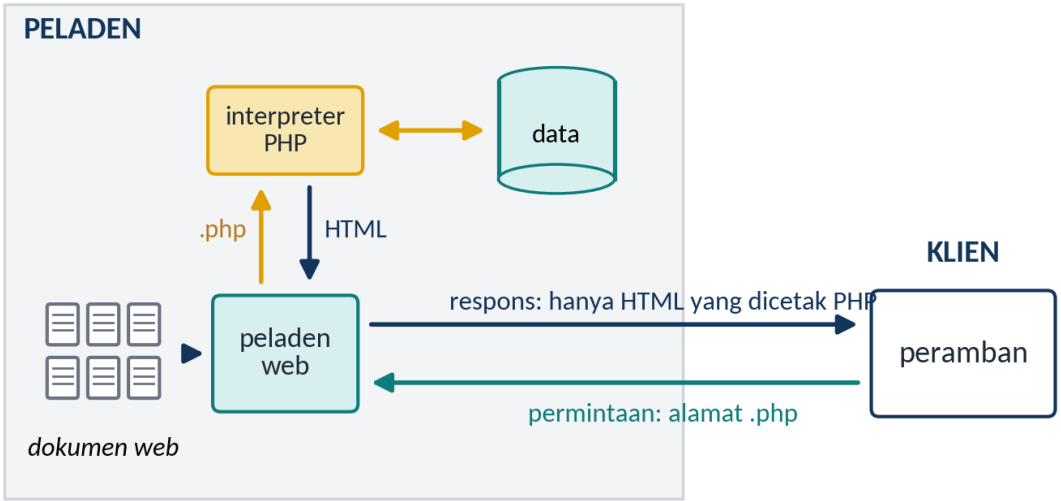
Untuk `.php` peladen menyerahkan berkasnya kepada **interpreter PHP**, menunggu, lalu mengirim balik apa pun yang dicetaknya.

WAWASAN KUNCI. PHP tidak berjalan di peramban. Ia berjalan di peladen, selesai, lalu hilang; "View source" hanya menampilkan HTML yang dicetaknya.

PHP, JSP, ASP.NET, dan Python menempati slot yang sama; PHP menawarkan jalan terpendek dari nol sampai halaman yang bekerja.

Java dan Python menyertai PHP di setiap bab, agar kita mempelajari gagasannya, bukan dialeknya. Buku: Subbab 3.2.1–3.2.3.

Arsitektur PHP



Gambar 3.1 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.

***Gambar 3.1**: klien mengirim permintaan biasa; peladen menjalankan interpreter; program berjalan dari atas ke bawah, mungkin sambil membaca data; keluarannya menjadi respons. Buku: Subbab 3.2.2.*

Riwayat Satu Permintaan

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

***Tabel 3.1**: interpreter menjalankan program **sekali, lalu melupakan segalanya**, sehingga apa yang harus bertahan disimpan di berkas, basis data, atau sesi; dan program tidak pernah berbicara langsung dengan peramban — ia mencetak, peladen yang mengantarkan. Buku: Subbab 3.2.4.*

Hello World, Melalui Peladen

Kode 3.1 `hello.php`: HTML di bagian luar, satu instruksi PHP di bagian dalam.

```
<html>
    <body>
        <?php
            echo "Hello World!";
        ?>
    </body>
</html>
```

```
<html>
    <body>
        Hello World!    </body>
</html>
```

Segala sesuatu di luar `<?php … ?>` disalin apa adanya: berkas PHP adalah berkas HTML yang dilubangi. Setiap pernyataan diakhiri `;`.

***Kode 3.1**: jalankan `php -S 127.0.0.1:8080` di foldernya dan minta lewat `http://`; alamat `file:///` tidak pernah menjalankan PHP. Berkas yang murni PHP tidak memakai penutup `?>`. Buku: Subbab 3.2.5 dan 3.3.3.*

Hello World dari Tiga Peladen

Kode 3.2 `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)
```

Di PHP **URL dipetakan ke berkas**; di Java ke **handler** yang didaftarkan dengan `createContext`; di Flask ke **fungsi** lewat `@app.route`.

***Kode 3.2**: PHP menyediakan `200 OK` dan `Content-Type` sendiri; peladen JDK membuat kita menulis status, header, dan panjang badannya; Flask berada di tengah, melengkapi string yang dikembalikan. Buku: Subbab 3.3.4.*

Cara-Cara Membuka Blok

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 3.2 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.

***Gambar 3.2**: pakailah `<?php … ?>` dan singkatan echo-nya `<?= $x ?>`; tag pendek bergantung pada pengaturan, dan bentuk script serta ASP sudah dihapus di PHP 7. Buku: Subbab 3.4.1.*

Hitung di Atas, Cetak di Tengah

Kode 3.3 `mixing.php`: menghitung di atas, mencetak di tengah. Dua lubang `<?= … ?>` diisi ketika halaman disusun.

```php
<?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>
```

Kode 3.3 *mengirim* `<p>Customer: Sri Wahyuni</p>` *dan* `<p>Weight: 3 kg</p>`*: tidak ada PHP dalam respons, karena PHP sudah datang dan pergi. Komentar:* `//`, `#`, `/* … */`*. Nama buatan kita peka huruf besar-kecil; nama bawaan PHP tidak. Buku: Subbab 3.4.2–3.4.4 dan 3.5.*

Variabel dan Konstanta

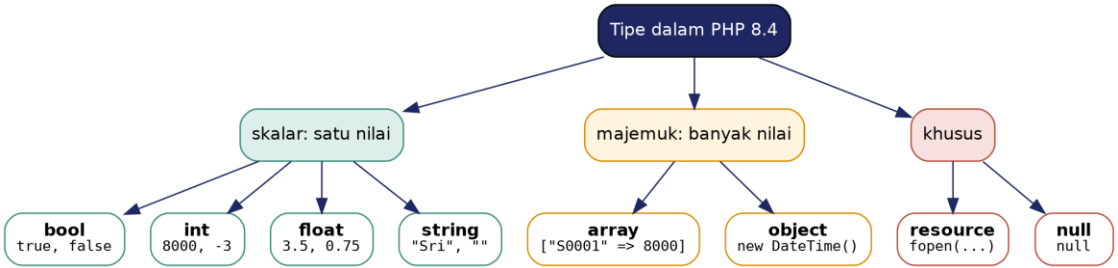
Variabel diawali `$`, tidak perlu dideklarasikan, peka huruf besar-kecil, dan tercipta oleh penugasan — sehingga `$totl = $total + 2000;` diam-diam membuat variabel baru.

> **DEFINISI. Penugasan** `$name = expression;` mengevaluasi sisi kanan, lalu menyimpan hasilnya di bawah nama di sisi kiri. `=` berarti "menjadi"; kesamaan adalah `==` atau `===`.

Konstanta tanpa `$`: `const PRICE_PER_KG = 8000;` atau `define('SHOP_NAME', 'Profitina Laundry');` — global, dan disunting di satu tempat.

Java mendeklarasikan, `int units = 4;`, dan salah ketik menjadi galat kompilasi; Python, seperti PHP, menciptakan variabel saat penugasan tetapi tanpa `$`. Buku: Subbab 3.6 dan 3.8.

Tipe-Tipe PHP



Gambar 3.3 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.

***Gambar 3.3**: tipe melekat pada nilai, bukan pada nama. Float disimpan biner, sehingga `0.1 + 0.2` bukan `0.3` dan uang tidak pernah disimpan dalam float; `null` bukan `0` dan bukan `""`. Buku: Subbab 3.7.1 dan 3.7.2.*

Type Juggling, dan Dua Macam Sama Dengan

`"10" + 5` adalah `15`: `+` hanya menjumlah, sehingga *string*-nya dibaca sebagai bilangan. `"10" . 5` adalah `"105"`: `.` hanya menyambung.

Operatornya memilih konteks, dan konteks itulah yang menggerakkan konversi.

`"10" == 10` adalah `true` (longgar); `"10" === 10` adalah `false` (ketat: tipe lebih dulu). `"abc" == 0` bernilai `false` di PHP 8, `true` di PHP 7.

Konversikan dengan sengaja: `intval("3.9 kg")` bernilai 3, `floatval` 3,9, dan `is_numeric` dengan jujur berkata `false`.

> **PAKAI `===` DAN `!==`** kecuali ada alasan untuk tidak memakainya.

JavaScript memakai `+` untuk kedua tugas itu; `.` yang terpisah adalah salah satu gagasan terbaik PHP. Buku: Subbab 3.7.3 dan 3.7.4.

Apa yang Dianggap Salah

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**

***Tabel 3.2**: `"0"` bernilai salah, tidak seperti di Python; satu spasi bernilai benar, sehingga field berisi satu spasi dianggap "benar". Itulah sebabnya setiap validator memanggil `trim()` lebih dulu. Buku: Subbab 3.7.5.*

Contoh Terurai: Presedens

Kerjakan `2 + 3 * 4 ** 2` dengan tangan, presedens tertinggi lebih dulu:

> **CONTOH TERURAI.** `**` mengikat paling kuat: $4^2 = 16$, tersisa `2 + 3 * 16`. Lalu `*` mengalahkan `+`: $3 \times 16 = 48$. Akhirnya $2 + 48 = 50$ — dan `03_operators.php` mencetak 50.

Jebakannya: `and` mengikat *lebih lemah* daripada `=`, sehingga `$x = true and false;` berarti `($x = true) and false;` dan `$x` bernilai `true`. Tulislah `&&` dan `||`.

Hubung-singkat: dalam `isset($_POST['units']) && $_POST['units'] > 0` uji kedua tidak pernah berjalan bila *field*-nya tidak ada. `??` memberi nilai bawaan: `$_POST['note'] ?? ''`.

`17 / 5` adalah `3.4` dan `intdiv(17, 5)` adalah 3; `17 % 5` adalah 2; `<=>` mengembalikan negatif, nol, atau positif, persis yang diperlukan pengurutan. Buku: Subbab 3.9.

Membuat Keputusan

```
if ($units <= 0) {
    $band = "nothing to wash";
} elseif ($units < 5) {
```

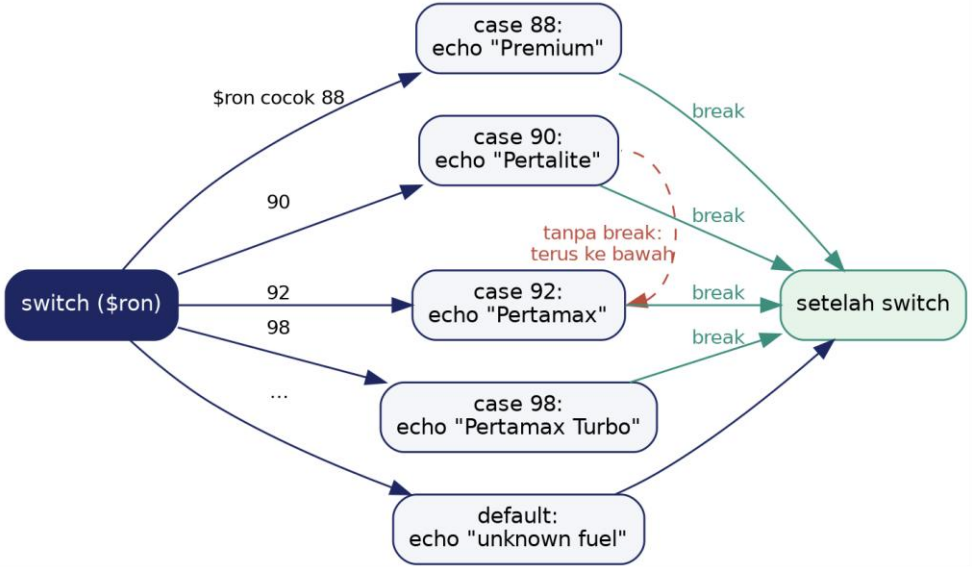
`elseif` mengambil syarat **pertama** yang benar dan melewati sisanya, sehingga urutan dari yang tersempit ke yang terluas membuat setiap uji tetap sederhana.

> `=` **BUKAN** `==`**.** `if ($age = 17)` menugaskan 17, selalu benar, dan merusak variabelnya. `if (17 == $age)` mengubah kekeliruan itu menjadi galat sintaksis.

Ternary memilih nilai; **sintaksis alternatif** `if (…): … endif;` menjaga *template* tetap terbaca.

Setiap halaman situs Laundry memakai sintaksis alternatif, karena `endif`, `endforeach`, dan `endwhile` menyebut apa yang sedang ditutup. Buku: Subbab 3.10.1 dan 3.10.2.

Cara `switch` Berjalan



Gambar 3.4 Cara `switch` berjalan. Kendali melompat ke `case` yang cocok; `break` keluar dari pernyataan. Tanpa `break`, eksekusi *terus mengalir* ke `case` berikutnya.

***Gambar 3.4**: bila RON 90 cocok dan tidak ada `break`, PHP mencetak `PertalitePertamaxPertamax Turbo` — ia terus mengalir melewati setiap case di bawahnya. Buku: Subbab 3.10.3.*

`switch` atau `match`?

```
echo match ($ron) {
    88 => "Premium", 90 => "Pertalite", 92 => "Pertamax", 98 => "Pertamax Turbo",
    default => "unknown fuel",
};
```

Tabel 3.3 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

***Tabel 3.3**: `switch` membandingkan dengan `==` longgar dan memerlukan `break`; `match` membandingkan dengan `===`, tanpa `break`, menghasilkan nilai, dan melempar galat bila tidak ada yang cocok tanpa `default`. Buku: Subbab 3.10.3.*

Empat Perulangan, Satu Pertanyaan

Pertanyaannya: **apakah Anda tahu sebelumnya berapa kali?**

`for` — awal, uji, dan langkah dalam satu baris: `for ($i = 1; $i <= 7; $i++)` mencetak `1234567`.

`while` — bila banyaknya tidak diketahui; lupakan inkremennya dan halaman berputar sampai `max_execution_time`.

`do … while` — menguji di bawah, sehingga berjalan **paling sedikit sekali**: ia mencetak `100` walaupun `100 <= 7` salah.

`foreach` — menelusuri *array* tanpa indeks dan tanpa cara untuk terlewat dari ujungnya.

Java punya keempat bentuk yang sama; Python hanya punya `for x in …` dan `while`, tanpa `do … while`. Buku: Subbab 3.11.

`break` dan `continue`

break_continue.php

```php
<html>
    <body>
        <?php
            for ($i = 0; $i <= 10; $i++) {
                if ($i == 3) {
                    continue;
                }
                if ($i == 7) {
                    break;
                }
                echo "$i ";
            }
        ?>
    </body>
</html>
```

Gambar 3.5 `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 3.5**: perulangannya ditulis dari 0 sampai 10, tetapi `continue` pada 3 dan `break` pada 7 mengubahnya. Buku: Subbab 3.11.1.*

Contoh Terurai: Penelusurannya

Tabel 3.4 Perulangan pada **Gambar 3.5** 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`

***Tabel 3.4**: keluarannya `0 1 2 4 5 6`. Ketika `continue` bekerja, `$i++` pada kepala `for` tetap berjalan; pada 7 `break` bekerja sebelum `echo`. Buku: Subbab 3.11.1.*

Fungsi dan Cakupan

`function name(parameters) { …; return value; }` — `return` menghentikan fungsi dan menyerahkan nilainya kepada *pemanggil*; ia tidak mencetaknya.

Tipe masuk dan keluar: `function circleArea(float $radius): float`; dengan `declare(strict_types=1);` tipe yang salah ditolak, bukan dikonversi.

Nilai bawaan dan argumen bernama: `laundryCost(units: 4, delivery: 10000)` mempertahankan harga bawaan.

Cakupan: variabel skrip utama sama sekali tidak ada di dalam fungsi. Kirimkan sebagai parameter; variabel `static` mempertahankan nilainya antarpanggilan, dalam satu permintaan.

Prosedur adalah fungsi yang `return`-nya tidak pernah ditulis; ia mengembalikan `null`. Buku: Subbab 3.12.1–3.12.3.

Dengan Nilai, dengan Referensi, dan Rekursi

Dengan nilai (bawaan) fungsi mengubah salinannya sendiri; dengan `int &$total` ia menerima **variabel itu sendiri**.

Utamakan dengan nilai: `addTipByReference($bill);` tidak memberi petunjuk di tempat pemanggilan bahwa `$bill` akan berubah.

***Array* disalin saat penugasan; objek tidak.**

> **CONTOH TERURAI.** `factorial(5)` memanggil `factorial(4)`, terus sampai `factorial(1)`, yang mengembalikan 1; jawaban-jawabannya dikalikan kembali ke atas menjadi $1 \times 2 \times 3 \times 4 \times 5 = 120$.

Dalam `05_functions.php` kedua baris `$bill` memperlihatkannya: dengan nilai variabel pemanggil selamat, dengan referensi tidak. Buku: Subbab 3.12.4.

Array Adalah Peta Berurutan

DEFINISI. ***Array*** dalam PHP adalah kumpulan berurutan pasangan kunci–nilai; setiap kunci berupa bilangan bulat atau *string*, dan setiap nilai boleh apa saja, termasuk *array* lain.

Berindeks: `["durian", "mango", "mangosteen", "pineapple"]` dihitung dari **0**; `$a[] = "red";` menambahkan di akhir. Meminta `[4]` memberi peringatan dan `null`.

Asosiatif: `["Yarik" => 80, "Naya" => 90]`, dibaca *dipetakan ke*; urutan penyisipan dipertahankan, sehingga `foreach` dapat diramalkan.

Kunci bersifat unik; `"7"` dan `7` adalah satu kunci; baca kunci yang mungkin tidak ada dengan `isset($a['k'])` atau `$a['k'] ?? $default`.

Dengan kunci bilangan bulat ia adalah array Java; dengan kunci string, `dict` Python atau `LinkedHashMap` Java. Buku: Subbab 3.13.1–3.13.3.

Menelusuri, Melihat, dan Mengurutkan

`foreach ($grade as $name => $grd)` — `foreach` untuk *array* dan `for` untuk bilangan. `each()` sudah dihapus di PHP 8.

Tabel 3.5 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

BILA KUNCINYA BERMAKNA, JANGAN MEMANGGIL `sort()`. Ia menghancurkan setiap nama dan menomori ulang dari 0.

***Tabel 3.5**: `asort()` mempertahankan nama; `print_r()` memperlihatkan strukturnya. Buku: Subbab 3.13.4–3.13.6.*

String

Tanda petik ganda melakukan interpolasi dan mengenali `\n`; tanda petik tunggal tidak; `"{$name}'s laundry"`; *heredoc* dan *nowdoc* untuk banyak baris.

`mb_strlen()` menghitung karakter, `strlen()` bita; `trim()`, `str_contains()`, `substr()`, `str_replace()`, `sprintf()`.

`number_format(24500, 0, ',', '.')` menjadi `24.500`: bahasa Indonesia memberi titik di antara ribuan.

> **POSISI 0 BERNILAI SALAH.** `if (strpos($s, $n))` bernilai salah baik ketika jarumnya tidak ada maupun ketika ia berada di awal. Tulislah `!== false`, atau pakai `str_contains()`.

`indexOf()` di Java dan `find()` di Python mengembalikan `-1`, sehingga tidak punya jebakan ini. Buku: Subbab 3.14.

Memecah Program ke Beberapa Berkas

Tabel 3.6 Keempat pernyataan penyertaan. Pakailah `require_once` untuk apa pun yang tanpanya program tidak dapat berjalan, yaitu hampir semuanya.

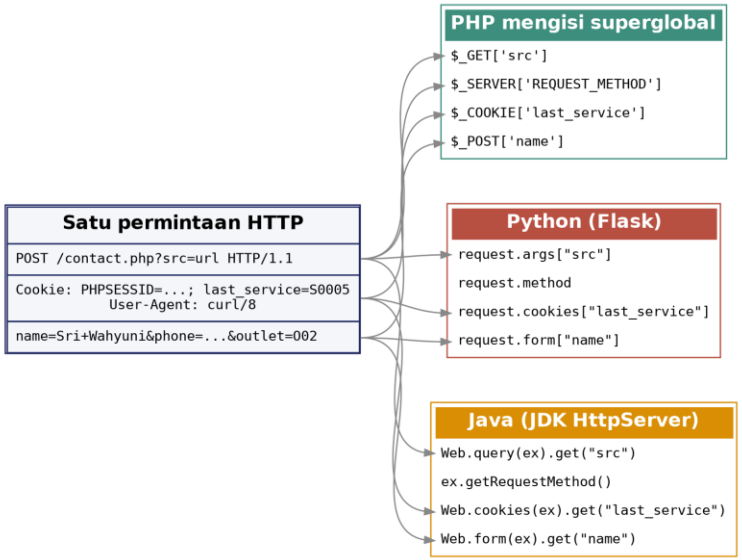
Pernyataan	Jika berkas tidak ada	Jika sudah dimuat
`include`	peringatan, eksekusi berlanjut	memuatnya lagi
`require`	galat fatal, eksekusi berhenti	memuatnya lagi
`include_once`	peringatan, eksekusi berlanjut	tidak melakukan apa-apa
`require_once`	galat fatal, eksekusi berhenti	tidak melakukan apa-apa

Kode 3.4 Dari `lib/money.php`: aturan harga BR-1, harga = harga satuan × jumlah satuan.

```
/** Rule BR-1 of the finished application: price = unit price x units. */
function price(int $unitPrice, int $units): int {
    return $unitPrice * $units;
}
```

Tabel 3.6*, *Kode 3.4: *tiga jas (S0005),* $3 \times 65{,}000 = 195{,}000$*. Buku: Subbab 3.16.*

Superglobal



Gambar 3.6 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.

***Gambar 3.6**: baris permintaan mengisi `$_GET` dan `$_SERVER`, header `Cookie` mengisi `$_COOKIE`, badan mengisi `$_POST`; Java dan Flask menyajikan bagian yang sama dengan nama lain. Buku: Subbab 3.17.1.*

Isi Masing-Masing

Tabel 3.7 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

HINDARI `$_REQUEST`. Ia mencampur GET dan POST tanpa menyebut asalnya, sehingga penyerang dapat memasok nilai hanya dengan mengirim sebuah tautan.

***Tabel 3.7**: **superglobal** adalah array asosiatif bawaan yang diisi sebelum kode kita berjalan dan terlihat di mana pun. Buku: Subbab 3.17.1.*

Semuanya Tiba sebagai *String*

```
curl -s -A 'curl-demo/1.0' "http://127.0.0.1:8080/09_superglobals.php?name=Sri&units=3"
```

`?` memulai *query string*, `&` memisahkan pasangan: `$_GET['name']` adalah `"Sri"` dan `$_GET['units']` adalah ***string*** `"3"`. Badan POST mengisi `$_POST`, *cookie* `-b` mengisi `$_COOKIE`; `Sri+Wahyuni` tiba sudah didekode menjadi `Sri Wahyuni`.

Java: `Web.query(ex)`, `Web.form(ex)`, `Web.cookies(ex)` dalam fungsi bantu yang ditulis sekali; Flask: `request.args`, `request.form`, `request.cookies`.

Tiga program, satu permintaan, empat fakta yang sama: metode, query, formulir, dan cookie. Buku: Subbab 3.17.2 dan 3.17.3.

GET Bertanya, POST Berbuat

Tabel 3.8 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

***Tabel 3.8**: kalkulator memakai GET, karena memuat ulang sebuah harga tidak berbahaya; formulir kontak memakai POST. POST tidak terenkripsi — hanya HTTPS yang mengenkripsi. Buku: Subbab 3.17.4.*

Formulir, dan Fungsi Bantunya

> **WAWASAN KUNCI.** `<input name="units">` tiba sebagai `$_GET['units']`, `Web.query(ex).get("units")`, `request.args["units"]`. Tanpa `name`, tanpa kunci.

Kode 3.5 Dari `lib/validate.php`: `field()` membaca satu nilai dengan aman, `h()` melakukan *escaping* untuk 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');
}
```

***Kode 3.5**: kotak centang yang tidak dicentang dan radio yang tidak dipilih tidak mengirim apa pun; `name="extras[]"` membuat PHP mengirim array, yang ditolak `field()`. `whole()` hanya menerima `^\d{1,9}$` dalam suatu rentang. Buku: Subbab 3.18.1 dan 3.18.2.*

Kalkulator Harga: Baca, Validasi

Kode 3.6 Dari `calculator.php`: salin masukan ke `$old`, validasi, dan hitung hasilnya dari daftar harga milik peladen sendiri.

```
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']),
        ];
```

***Kode 3.6**: pengunjung memilih layanan mana; peladen yang memutuskan harganya. Menyalin ke `$old` adalah langkah yang dilewatkan pemula, sehingga banyak formulir mengosongkan dirinya setelah satu kesalahan. Buku: Subbab 3.18.3.*

Contoh Terurai: Tiga Permintaan

`?service=S0005&units=3`: `3 items x Rp 65.000 = Rp 195.000`, dan `Set-Cookie: last_service=S0005; … HttpOnly; SameSite=Lax`.

`?service=S0001&units=4`: `4 kg x Rp 8.000 = Rp 32.000`, dengan `Wash, dry & fold` di-*escape* oleh `h()`.

`?service=S0099&units=3.5`: dua pesan, dan `3.5` milik pengunjung tetap di kotaknya.

```
  <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>
```

`LaundrySite` di Java dan `app.py` di Flask menjawab ketiga permintaan yang sama dengan baris-baris yang sama: tiga bahasa, satu jawaban, Rp 195.000. Buku: Subbab 3.18.3–3.18.5.

Kalkulator dalam Flask

Kode 3.7 Dari `app.py`: kalkulator dalam Flask; `response.set_cookie` menulis header yang sama dengan `setcookie()`.

```
@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
```

***Kode 3.7**: Jinja meng-escape setiap `{{ … }}` dengan sendirinya. Buku: Subbab 3.18.5.*

Lubangnya, Diperagakan

Kode 3.8 `unsafe.php`, sengaja dibuat rusak. Halaman ini berfungsi bagi setiap pengunjung yang berkelakuan baik, dan justru itulah yang membuatnya berbahaya.

```php
<?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>
```

```
<p>Hello, <script>alert("stolen")</script>!</p>
```

***Kode 3.8**: pengunjung memasok teks, program memperlakukannya sebagai isi, dan peramban sebagai kode. Inilah **cross-site scripting** (XSS); bila disimpan dan ditampilkan kepada orang lain, setiap pembaca menjalankannya. Buku: Subbab 3.19.1.*

Perbaikannya

Kode 3.9 `safe.php`: satu-satunya perubahan adalah `h(…)` di sekeliling nilainya.

```php
<?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>
```

```
<p>Hello, &lt;script&gt;alert(&quot;stolen&quot;)&lt;/script&gt;!</p>
```

> **WAWASAN KUNCI.** Lakukan *escaping* pada **keluaran**, saat mencetak, sesuai tujuan pencetakannya — setiap nilai, setiap kali.

***Kode 3.9**: `htmlspecialchars()` mengubah `&`, `<`, `>`, `"`, dan `'` menjadi entitas, dan kelima karakter itulah seluruh mekanisme yang membuat teks menjadi markah. Buku: Subbab 3.19.2.*

Validasi Bukan *Escaping*

DEFINISI. Validasi memutuskan apakah suatu nilai dapat diterima *sama sekali*, dan menolaknya bila tidak. ***Escaping*** membuat nilai yang sudah diterima aman untuk *satu tujuan tertentu*.

Tabel 3.9 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

***Tabel 3.9**: `' OR '1'='1` dalam string SQL yang dirangkai adalah injeksi SQL, dijawab di Bab 5 dengan prepared statement. Setiap celah injeksi adalah satu kalimat: data dibiarkan menjadi kode. Buku: Subbab 3.19.3.*

Enam Keputusan, Dua Tembok

1. Sisa program hanya memakai `$clean`. **2.** Kumpulkan setiap galat, bukan hanya yang pertama.

3. Periksa gerai terhadap daftarnya sendiri: ***allowlist*** mengalahkan *blocklist*. **4.** Bersihkan nomor telepon, lalu nilai: `^\d{8,15}$`.

5. Hitung karakter, bukan bita: `mb_strlen()`. **6.** Nama menerima huruf dari semua abjad, `\p{L}\p{M}`, serta `'`, `.`, `-`, dan spasi.

```
<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 tampilan ulangnya tidak berbahaya: salah satu saja sudah cukup, dan memiliki keduanya adalah **defense in depth**. Pemeriksaan peramban adalah sopan santun; pemeriksaan peladen adalah pagarnya. Buku: Subbab 3.19.4.*

Cookie: Satu Header ke Setiap Arah

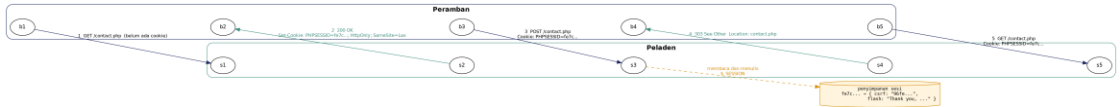
```
response:  Set-Cookie: last_service=S0005; Max-Age=2592000; Path=/; HttpOnly; SameSite=Lax
requests:  Cookie: last_service=S0005
```

`Max-Age` dalam detik: $30 \times 24 \times 3600 = 2{,}592{,}000$. `HttpOnly` menyembunyikannya dari skrip; `SameSite=Lax` menjauhkannya dari formulir situs lain; tambahkan `Secure` di HTTPS.

Cookie adalah header, sehingga dikirim **sebelum keluaran apa pun**; dan ia masukan pengunjung, sama tidak tepercayanya dengan `$_GET`.

HTTP tidak menyimpan state dan PHP melupakan setiap variabel, padahal toko memerlukan ingatan; `setcookie()`, `Web.setCookie()`, dan `response.set_cookie()` menulis header yang sama. Buku: Subbab 3.20.1 dan 3.20.2.

Sesi, Langkah demi Langkah



Gambar 3.7 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.

***Gambar 3.7**: peramban hanya memegang id acak, `PHPSESSID`; datanya tinggal di peladen, dan `session_start()` memuatnya ke `$_SESSION`. Buku: Subbab 3.20.3.*

Cookie atau Sesi?

Tabel 3.10 *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`

***Tabel 3.10**: login tidak boleh tinggal dalam cookie yang dapat diubah pengunjung. `session` milik Flask berbeda jenis — ditandatangani, tidak dienkripsi, disimpan dalam cookie itu sendiri — jadi jangan pernah menaruh rahasia di dalamnya. Buku: Subbab 3.20.3 dan 3.20.6.*

Token Formulir dan Pesan Sekali Tampil

Kode 3.10 Dari `lib/session.php`: pemeriksaan CSRF dan pesan “flash” yang hanya tampil sekali.

```
/** 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;
}
```

***Kode 3.10**: situs lain dapat membuat peramban mengirim cookie tetapi tidak dapat membaca tokennya. `hash_equals()` berjalan dalam waktu tetap, dan `$token !== ''` mencegah token kosong cocok dengan field kosong — bug yang tertangkap saat menulis versi Java. Buku: Subbab 3.20.3 dan 3.20.4.*

Post, Redirect, Get

Kode 3.11 Dari `contact.php`: validasi, simpan, titipkan ucapan terima kasih di sesi, lalu jawab `303 See Other`.

```
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;
    }
}
```

Kode 3.11*: setelah pengalihan, Refresh hanya mengulang GET yang tidak berbahaya. Hasil nyata: empat field salah memberi empat pesan; tanpa token muncul "This form has expired"; kiriman yang benar mendapat* `303` *dan* `Location: contact.php`*; ucapan terima kasih tampil sekali. Buku: Subbab 3.20.5.*

Jangan Pernah Menyimpan Kata Sandi

> **DEFINISI. *Hash* kata sandi** adalah keluaran fungsi satu arah yang sengaja dibuat lambat dan dicampur ***salt*** acak. Dari *hash* Anda tidak dapat menghitung kata sandinya, hanya menguji tebakan, satu tebakan lambat demi satu tebakan lambat.

Kode 3.12 Dari `make_hash.php`: `password_hash()` dan `password_verify()`.

```
$hash = password_hash($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
```

```
length: 60
algorithm: bcrypt
verify right password: true
verify wrong password: false
same hash twice? false
```

***Kode 3.12**: `$2y$12$...` menyebut bcrypt dengan cost 12, lalu salt dan hash-nya; kata sandi yang sama tidak pernah menghasilkan hash yang sama. SHA-256 saja bukan hash kata sandi: pakai Argon2id, scrypt, bcrypt, atau PBKDF2. Buku: Subbab 3.21.1 dan 3.21.2.*

Login Staf

Kode 3.13 Dari `login.php`: periksa token, verifikasi terhadap *hash*, dan beri sesi id baru.

```
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
    }
}
```

***Kode 3.13**: satu pesan untuk kata sandi salah dan e-mail tak dikenal; kata sandi **tidak di-trim**; `session_regenerate_id(true)` menggagalkan session fixation; e-mail tak dikenal tetap menjalankan `password_verify()` tiruan agar waktu tidak membocorkan apa pun. Buku: Subbab 3.21.3.*

Login yang Sama di Java

Kode 3.14 Dari `Passwords.java`: verifikasi PBKDF2 dalam format aplikasi jadi, dibandingkan dalam waktu tetap.

```
    /** 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
    }
```

Format tersimpan: PHP bcrypt `$2y$12$…` (60 karakter), Java `pbkdf2_sha256$600000$…` (90), Werkzeug di Flask `scrypt:32768:8:1$…` (162).

***Kode 3.14**: aplikasi jadi menyimpan satu format PBKDF2 yang dipahami semua jalurnya — keputusan rancangan, bukan fitur bahasa. Buku: Subbab 3.21.4.*

Menerima Berkas

Kode 3.15 Dari `upload.php`: tipe sebenarnya diambil dari bita-bitanya, dan namanya dibuat sendiri.

```
        // 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 . ').';
```

```
<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>
```

***Kode 3.15**: `enctype="multipart/form-data"`; uji `error` lebih dulu; `is_uploaded_file()` menggagalkan path palsu; ekstensinya diambil dari allowlist. Peramban berkata `image/png`, bitanya berkata PHP, dan bitanya yang menang. Buku: Subbab 3.22.*

Membaca Galat

```
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
```

Parse error: tidak ada yang berjalan. Bila disebut baris n, lihatlah baris $n-1$ — di sini `;` yang hilang setelah `echo "one"`.

Warning: "ada yang salah dan saya terus berjalan" — *bug* yang belum disadari.

Fatal error: "saya tidak dapat melanjutkan"; baca *stack trace* dari bawah ke atas. `TypeError` dari `strict_types` adalah galat yang *Anda inginkan*.

"Headers already sent": ada yang dicetak sebelum `header()`, `setcookie()`, atau `session_start()`.

`display_errors` menyala selama pengembangan dan mati di produksi; `log_errors` selalu menyala; `error_reporting` pada `E_ALL`. Buku: Subbab 3.23.

Satu Gagasan, Tiga Tumpukan

Tabel 3.11 Gagasan utama bab ini dalam tiga tumpukan teknologi. Kuasai barisnya, dan kolom tempat Anda 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")`
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

***Tabel 3.11**: kuasai barisnya, dan kolomnya tidak lagi penting. Buku: Subbab 3.24.*

Profitina dalam Praktik

Setiap masukan adalah pagar: alamat situs yang diketik pemilik merek diurai, dibentuk, dan diperiksa terhadap *allowlist* — bentuk `validateContact()`.

***Escaping* pada keluaran, setiap kali:** umpan komunitas menampilkan teks seorang pengguna kepada semua pengguna lain.

Sesi dan kata sandi sesuai kaidah: *hash* Argon2id, dan *login* yang dibatasi lajunya.

Data pribadi adalah hukum: di bawah UU PDP 27/2022, persetujuan eksplisit, ekspor, penggantian kata sandi, dan penghapusan akun sepenuhnya.

Yang terbawa adalah bentuknya: masukan tak tepercaya, validator, escaper, sesi, hash. Studi kasus yang dapat dijalankan selalu Profitina Laundry. Buku: Subbab 3.25.

Kesalahan yang Sering Terjadi

`=` untuk `==`; membaca `$_POST` pada GET; memercayai validasi peramban atau *cookie*.

Mencetak masukan tanpa `h()`; menjawab POST yang berhasil dengan halaman, bukan 303.

Token formulir yang dapat dicocokkan `''`; menyimpan kata sandi atau *hash* cepatnya; men-*trim* kata sandi.

Mempertahankan id sesi melewati *login*; memercayai `$_FILES['x']['type']` atau `['name']`.

Keluaran sebelum `header()`, `setcookie()`, atau `session_start()`: tinggalkan penutup `?>` pada berkas yang murni PHP.

Buku mendaftar sepuluh kesalahan, masing-masing dengan cara memperbaikinya. Buku: Subbab 3.26.

Ringkasan: Lima Hal untuk Diingat

1. Bahasa sisi peladen berjalan di peladen, dan keluarannya adalah halaman (**Gambar 3.1**); setiap permintaan dijawab sendiri-sendiri.

2. PHP melakukan *juggling* tipe, jadi bandingkan dengan `===`; `foreach` untuk *array*; fungsi menerima salinan kecuali parameternya ber-`&`; `sort()` menghancurkan kunci *string*.

3. *Superglobal* membawa permintaan masuk, dan setiap nilai adalah *string* dari orang asing. GET bertanya; POST berbuat, dan POST yang berhasil dialihkan.

4. Validasi pada masukan, *escaping* pada keluaran (**Kode 3.8** berbanding **Kode 3.9**).

5. *Cookie* di peramban, sesi di peladen; token pada formulir; id sesi baru saat *login*; simpan hanya *hash* kata sandi yang lambat dan ber-*salt*.

Bacalah OWASP Top 10 sekali tahun ini, serta cheat sheet-nya tentang XSS dan penyimpanan kata sandi sebelum memasang apa pun. Buku: Subbab 3.27.

Pertemuan Berikutnya

Bab 4 — Evaluasi dan Pembahasan I. Sebelas soal tentang Bab 1 sampai 3, dari situs pribadi lima halaman sampai formulir penyerahan yang tervalidasi dan ter-*escape*.

Setiap soal dikerjakan tuntas: dipahami dengan tangan, diselesaikan dalam teknologi yang disebutnya serta dalam Java dan Python, lalu ditunjukkan penyelesaian keliru yang menggoda.

Sebelum itu: kerjakan Soal Latihan 3.1, 3.2, 3.9, 3.14, 3.15, dan 3.16, dan jalankan ketiga jalur halaman Laundry.

Bab 4 melatih setiap gagasan dari tiga bab ini, di jaringan, di peramban, dan di peladen. Buku: Subbab 3.28 dan 3.29.

Kuliah 4 — Evaluasi dan Pembahasan I: Dari Situs Statis ke Formulir yang Tervalidasi

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 4

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini tanpa fitur bahasa baru: sebelas soal utuh tentang Bab 1 sampai 3, masing-masing diselesaikan dari nol, diperiksa oleh program, lalu disusul penyelesaian keliru yang menggoda. Buku: Bab 4, Subbab 4.1–4.17.

Cakupan Pertemuan Ini

1. **Sebuah metode** — delapan langkah untuk soal web kecil apa pun, dan tiga instrumen yang memperlihatkan apa yang sebenarnya terjadi.
2. **Soal 4.1–4.4, satu proyek** — situs pribadi lima halaman, URL bersih, laporan yang berdiri di atas bukti, dan penerbitan beserta kode sumbernya.
3. **Soal 4.5–4.8, jaringan dan halaman** — satu permintaan status byte demi byte, GET atau POST, daftar harga sebagai tabel sungguhan, *cascade* dan model kotak dengan tangan.
4. **Soal 4.9–4.11, angka dan formulir** — perkiraan harga langsung yang tepat, nota yang dihitung dengan PHP, dan formulir penitipan yang divalidasi dan di-*escape*.
5. **Soal-soal berdampingan**, dan jebakan yang mereka bagi.

Setiap soal diselesaikan dalam teknologi yang dimintanya dan sekali lagi dalam Java dan Python; setiap keluaran yang dikutip adalah keluaran nyata. Buku: Subbab 4.1.

Dari Mana Kita Berangkat: Perkakasnya

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

Tabel 4.1 *adalah peta: bila HTTP masih terasa abstrak, mulailah dari 4.5 dan 4.6; bila CSS mengejutkan Anda, 4.8; bila array PHP masih goyah, 4.10. Buku: Subbab 4.1.*

Delapan Langkah untuk Soal Web Apa Pun



Gambar 4.1 Delapan langkah yang dipakai untuk setiap soal dalam bab ini; panah putus-putus adalah putaran antara membangun dan mengamati.

***Gambar 4.1**: empat gerakan Pólya — memahami, merencanakan, melaksanakan, meninjau kembali — direntangkan menjadi delapan langkah; panah putus-putus adalah putaran antara membangun dan mengamati. Buku: Subbab 4.2.*

Langkah-Langkahnya, dan yang Paling Penting

1 Nyatakan ulang: apa yang masuk, apa yang harus keluar. **2** Kerjakan contohnya dengan tangan. **3** Putuskan apa berjalan di mana.

4 Sketsakan berkas, URL, dan data. **5** Tulis pemeriksaannya lebih dulu. **6** Bangun dalam langkah kecil.

7 Jalankan dan amati — perbedaan pertama dari langkah 2 *adalah bug*-nya. **8** Tinjau kembali: jalan keliru yang menggoda, dan sebuah varian.

> **MENGAPA "APA BERJALAN DI MANA" DIDAHULUKAN.** Kebanyakan *bug* web pemula adalah pekerjaan yang ditaruh di sisi yang salah: harga yang hanya dihitung JavaScript, formulir yang hanya diperiksa HTML, tautan bersih yang tidak dikenali server.

Tiga instrumen: `curl -i` *atau* `-v` *memperlihatkan percakapan sebagai teks; developer tools memperlihatkannya sebagaimana dialami peramban; peramban headless (Playwright) menjadikan developer tools sebuah program. Buku: Subbab 4.2.1–4.2.3.*

Soal 4.1: Lima Halaman

> **SOAL 4.1.** Bangunlah situs pribadi statis dengan HTML, CSS, dan JavaScript yang terdiri atas halaman beranda, profil diri, kampung halaman, makanan khas daerah itu, dan tempat wisatanya.

Statis: server mengembalikan berkas yang sudah ada; tanpa PHP, tanpa basis data, tanpa formulir yang mengirim apa pun.

Halaman: lima dokumen, masing-masing dengan judul sendiri dan, di Soal 4.2, alamat sendiri — bukan lima bagian dari satu halaman.

HTML, CSS, dan JavaScript: satu perilaku kecil yang jujur; lengkap tanpa skrip, sedikit lebih baik dengannya.

Contohnya Ayu Pramesti, mahasiswi Informatika di Surabaya yang berasal dari Malang: satu bingkai — kepala, navigasi, kaki — di sekeliling lima bagian `main` yang berbeda, satu `site.css`, dan satu `site.js`. Buku: Subbab 4.3.1 dan 4.3.2.

Satu Skrip, Dua Tugas Kecil

Kode 4.1 Dari `web/portfolio/assets/site.js`: menandai halaman aktif dan menulis tahun berjalan.

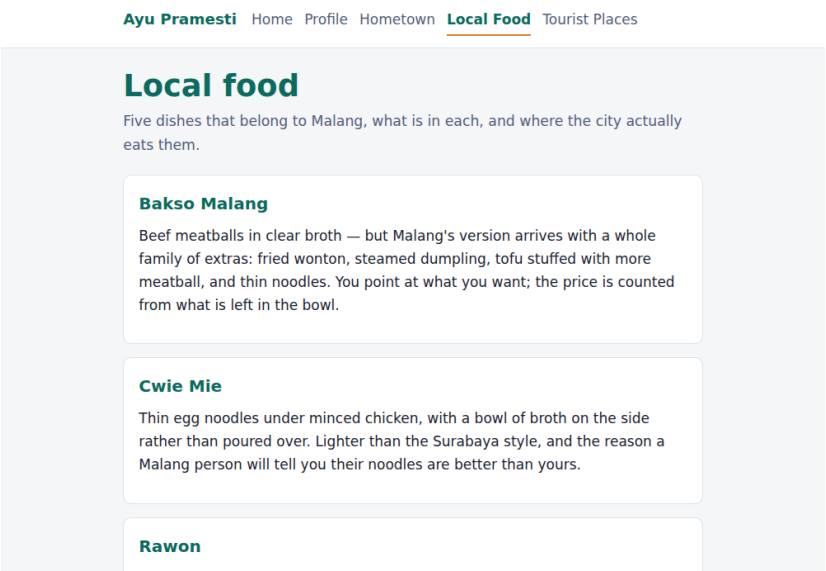
```
(function () {
  "use strict";

  // 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()); }
}());
```

***Kode 4.1**: hapus berkas itu dan situsnya tetap bekerja — progressive enhancement. Buku: Subbab 4.3.3–4.3.5.*

Yang Ditampilkan Peramban



Gambar 4.2 Halaman Local Food selebar 1000 piksel; skrip telah menandai *Local Food* sebagai halaman saat ini dan lembar gaya menggarisbawahinya.

***Gambar 4.2**: Local Food ditandai sebagai halaman yang sedang dibuka; di bawah 720 piksel, lembar gaya yang ditulis mobile first memberi satu kolom dan navigasi yang terlipat. Buku: Subbab 4.3.6.*

Java dan Flask: Bingkai Ditulis Sekali

Kode 4.2 Dari `java/PortfolioServer.java`: menu ditulis sekali, dan halaman yang sedang dibuka ditandai di server.

```
    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");
        }
```

Di Flask bingkainya satu *template* Jinja; `{% if key == slug %} aria-current="page"{% endif %}` mengerjakan tugas yang sama.

`curl -s localhost:8402/portfolio/food | sed 's/ aria-current="page"//' | cmp - …/food/index.html` tidak mencetak apa pun: byte demi byte sama dengan berkas statisnya.

***Kode 4.2**: halaman keenam berarti satu berkas baru dan satu entri peta di Java, berbanding tujuh berkas HTML yang disunting di situs statis — itulah seluruh alasan menulis bingkai sekali saja. Buku: Subbab 4.3.7 dan 4.3.8.*

Keliru yang Menggoda: Satu Halaman, Lima Fragmen

Lima `section` dalam satu berkas, ditampilkan satu per satu oleh `section:target { display: block; }`: pendek, dan di peramban tampak seperti lima halaman.

```
> GET /wrong/onepage.html HTTP/1.1
> Host: localhost:8404
```

Permintaannya berbunyi `GET /wrong/onepage.html`: `#food` tetap di peramban, sehingga server maupun mesin pencari tidak dapat membedakan “halaman-halaman” itu.

Satu judul, lima `h1`, dan halaman kosong bila dibuka tanpa fragmen.

Fragmen tidak pernah dikirim ke server, sehingga rancangan ini tidak akan pernah memenuhi Soal 4.2. Buku: Subbab 4.3.9.

Soal 4.2: URL Bersih

> **SOAL 4.2.** Kelima halaman Soal 4.1 harus dapat dicapai di `(Domain Anda)/portfolio`, `/portfolio/profile`, `/portfolio/hometown`, `/portfolio/food`, dan `/portfolio/tourist`.

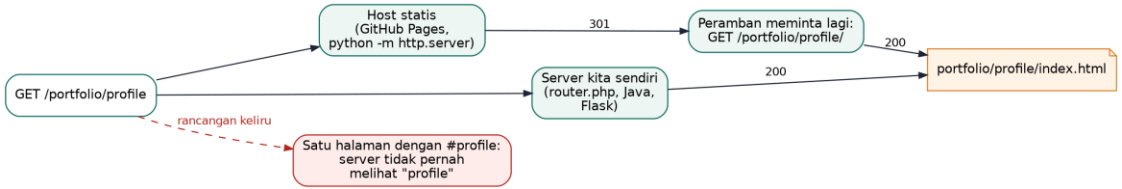
Dari mana `/portfolio` berasal? Dari folder bernama demikian — atau, di GitHub Pages, dari repositori bernama `portfolio`.

Bagaimana alamat yang tidak menyebut berkas menemukan berkasnya? Aturan tertua setiap server web: permintaan untuk sebuah **direktori** dijawab dengan `index.html`-nya.

Bagaimana tautan bertahan saat situs pindah ke *host*? Dengan membawa awalan lengkapnya.

Semua ini bukan urusan HTML: ia diputuskan oleh susunan folder, konfigurasi host, atau sebuah program. Buku: Subbab 4.4.1 dan 4.4.2.

Tiga Nasib Satu Permintaan



Gambar 4.3 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.

***Gambar 4.3**: host statis menjawab `301` ke alamat bergaris miring lalu `200`; server buatan kita menjawab `200` seketika; rancangan satu halaman tidak pernah membiarkan server tahu halaman mana yang diminta. Buku: Subbab 4.4.2.*

Tautan Mana yang Bertahan?

Kode 4.3 Dari `web/tools/resolve.mjs`: aturan peramban sendiri, `new URL(href, page)`, pada halaman Profile situs yang sudah terbit.

```
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
```

***Kode 4.3**: `/food/` menunjuk ke atas situs, `food/` mendarat di dalam folder Profile, `../food/` hanya benar dari sini. Hanya `/portfolio/food/` yang benar dari setiap halaman. Buku: Subbab 4.4.2.*

Router untuk `php -S`

Kode 4.4 Dari `php/router.php`: aturan-aturan *host* dituliskan; Aturan 0 menjaga setiap permintaan tetap di dalam folder `web`.

```
// 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);
```

***Kode 4.4**: Aturan 2 menjawab direktori dengan `index.html`-nya, dengan atau tanpa garis miring, sehingga `/portfolio/profile` mendapat `200` seketika; Aturan 3 mengirim `404.html` milik situs dengan status 404. Buku: Subbab 4.4.3.*

Empat Cara Menuju Alamat Bersih

Tabel 4.2 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`

***Tabel 4.2**: rewrite mempertahankan alamat dalam satu permintaan; redirect mengubahnya dalam dua. Netlify menulis `_redirects`, Apache membaca `.htaccess`; kedua berkas itu lengkap, tetapi tidak ada yang dijalankan untuk buku ini. Buku: Subbab 4.4.3.*

Contoh Terurai: Memeriksa Kelimanya

```
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 biasa, setiap path yang diminta menjawab `301` lalu halamannya: “ok after redirect”. Lima dari lima, dengan kedua cara.

Kesalahan yang menggoda ada di tautannya: `href="/profile/"` bekerja di akar `localhost` dan rusak di bawah `/portfolio/`. Ujilah sebagaimana situs akan disajikan — sajikan folder induknya. Buku: Subbab 4.4.4–4.4.6.

Soal 4.3: Laporan di Atas Bukti

> **SOAL 4.3.** Tulislah laporan tentang situs Soal 4.1 dan 4.2 yang memuat konsep rancangan dan strukturnya, tangkapan layar, penjelasan kode dan pilihan implementasinya, analisis atau evaluasi, serta kesimpulan.

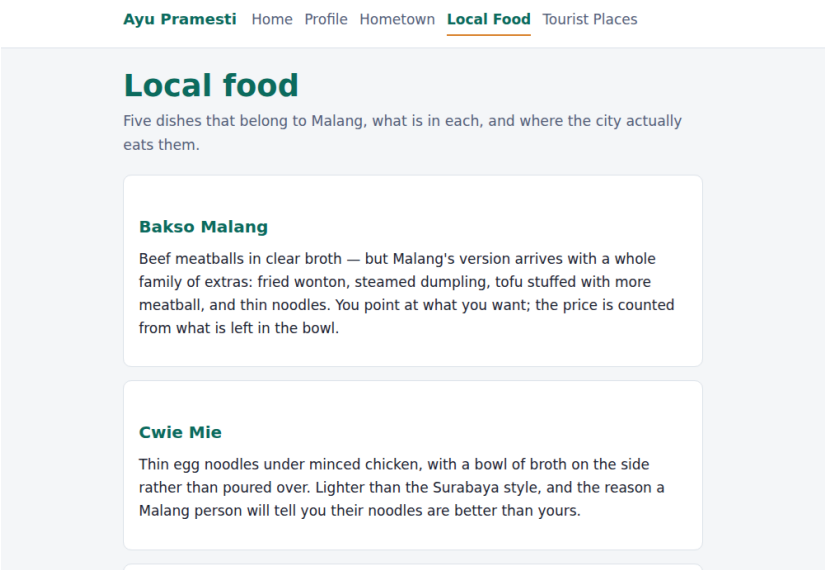
"Situsnya cepat dan tidak ada tautan rusak" adalah pendapat. "Lima halaman, 13.354 byte HTML; ketujuh alamat internal menjawab `200`" adalah evaluasi, karena pembaca dapat memeriksanya.

Jadi soal menulis ini berinti pemrograman: **ukurlah situsnya**, melalui HTTP, sebagaimana pengunjung.

```
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
```

Audit yang sama dalam Java, diarahkan ke server Java, mendapati setiap halaman tepat 20 byte lebih besar: panjang `aria-current="page"`. Ukuran yang menjelaskan dirinya sampai ke byte. Buku: Subbab 4.5.1–4.5.5.

Sebelum dan Sesudah



Gambar 4.4 Halaman Local Food sebelum diperbaiki: jarak 47 piksel antara tepi atas setiap kartu dan judulnya, padahal rancangannya menghendaki 19.

***Gambar 4.4**: garis tepi 1 + padding 18 + margin atas judul 28 = 47 piksel. Setelah* `.card > :first-child { margin-top: 0; }` *peramban mengukur 19. Pengamatan, sebab dengan aritmetika, perubahan, pengukuran: satu paragraf evaluasi. Buku: Subbab 4.5.6.*

Soal 4.4: Menayangkan dan Menerbitkan Kode Sumber

> **SOAL 4.4.** Tayangkan situs yang sudah selesai di layanan *hosting* web gratis, dan unggah seluruh kode sumbernya ke repositori GitHub publik.

Kode 4.5 Dari `web/tools/publish-demo.sh`: *commit* situsnya, lalu *push* ke repositori *bare* yang berperan sebagai GitHub.

```
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
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
```

***Kode 4.5**: akar repositori adalah akar situs; Pages lalu menyajikannya di `https://ayu-pramesti.github.io/portfolio/`. Gladi ini meng-clone-nya, menyajikannya di port 8405, dan menjalankan pemeriksaan rute. Buku: Subbab 4.6.1–4.6.3.*

Halaman Dibekukan, dan Dua Godaan

Pages tidak menjalankan program, jadi situs Flask **dibekukan**: `app.test_client()` meminta setiap halaman, dan setiap jawaban menjadi berkas; `/portfolio/nonesuch` menjadi `404.html`.

Folder yang salah: repositori dari folder induk menerbitkan situs satu tingkat terlalu dalam, `…/portfolio/portfolio/`.

Rahasia: kata sandi yang di-*commit* ke repositori publik tetap ada di riwayatnya walaupun sudah dihapus; daftarkan berkas semacam itu di `.gitignore` sejak *commit* pertama.

> **UJI DIRI.** Salah ketik diperbaiki di `food/index.html`: `git add food/index.html`, `git commit -m "…"`, `git push`; lalu `curl -s …/portfolio/food/ | grep -c "kata yang dibetulkan"` mencetak 1 begitu versi baru tayang.

Pembangkit situs statis seperti Jekyll adalah gagasan pembekuan ini dalam bentuk dewasa. Buku: Subbab 4.6.4 dan 4.6.5.

Soal 4.5: Satu Permintaan di Jaringan

> **SOAL 4.5.** Tulislah server untuk `/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 `Allow` untuk metode lain.

Pakai nota B0120 (0856-0000-8080, sudah diambil), B2938 (0896-0000-1482, siap diambil), dan B2946 (0811-0000-8361, sedang diproses).

Hanya path dan query yang menempuh baris permintaan: skema dan port menentukan cara menyambung, host masuk ke header `Host`, dan fragmen tidak pernah meninggalkan peramban. Buku: Subbab 4.7.1 dan 4.7.2.

Dikerjakan dengan Tangan: Enam Permintaan

Tabel 4.3 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`

***Tabel 4.3**: jawabannya diputuskan sebelum ada kode, lalu menjadi pemeriksaannya. Buku: Subbab 4.7.2.*

Kode Status Membawa Putusannya

Tabel 4.4 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

***Tabel 4.4**: telepon yang salah mendapat jawaban yang sama dengan nota tak dikenal, sehingga mengetik nomor nota satu per satu tidak membocorkan nota mana yang ada — tuntutan pelindungan data pribadi. Buku: Subbab 4.7.2.*

Cek Status dalam PHP

Kode 4.6 Dari `php/status.php`: baca, bersihkan, cari, lalu jawab dengan kode yang tepat.

```
    $bill  = strtoupper(trim($_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']);
    }
```

```
B2938: Ready for pickup
[200]
B2938: Ready for pickup
[200]
No bill found for this number and phone.
[404]
```

***Kode 4.6**: `?? ''` mencegah parameter yang tidak ada memicu peringatan; `strtoupper(trim(…))` menerima `b2938`; `preg_replace('/[\s-]/', '', …)` membuang spasi dan tanda hubung. Setiap baris cocok dengan tabel yang dikerjakan dengan tangan. Buku: Subbab 4.7.3.*

Contoh Terurai: Byte Persisnya

```
> 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
```

Baris permintaan, header `Host` yang diwajibkan HTTP/1.1, `Connection: close`, dan baris kosong yang mengakhiri header; lalu baris status, header, baris kosong, dan badan.

Web Bab 1 dalam wujud nyata. Java mengurai `getRawQuery()` sendiri; Flask mengembalikan `(body, status, headers)` dan menulis `405 METHOD NOT ALLOWED` dengan huruf kapital — program hanya membaca angkanya. Buku: Subbab 4.7.4–4.7.6.

Keliru yang Menggoda: `200 OK`, "Tidak Ditemukan"

```
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 cache, mesin pencari (***soft 404***), serta `fetch` memercayai baris statusnya.

Ia juga menjawab `DELETE` seperti `GET`, dan membandingkan telepon secara persis, sehingga `0896-0000-1482` seperti tercetak di nota dianggap "tidak ditemukan".

Uji diri: `?phone=089600001482&bill=B2946&lang=id` mendapat `200 OK` dan `B2946: In process` — `$_GET` dicari menurut kunci, dan `lang` yang tidak dibaca diabaikan. Buku: Subbab 4.7.7.

Soal 4.6: GET atau POST?

SOAL 4.6. Satu formulir *mencari* status nota; satu lagi *membuat* nota dari nama, telepon, layanan, dan jumlah. Pilihlah metode masing-masing; beri nomor B3001, B3002, …; pastikan memuat ulang halaman tidak pernah membuat nota kedua.

Tabel 4.5 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

***Tabel 4.5**: pencarian tidak mengubah apa pun, jadi memakai GET dan dapat ditandai; penitipan membuat nota, jadi memakai POST — tidak aman dan tidak idempoten. Buku: Subbab 4.8.1 dan 4.8.2.*

Contoh Terurai: 61 Karakter, lalu 303

Siti Aminah menitipkan 3,5 kg S0001. Spasi menjadi `+`, koma menjadi `%2C`: 61 karakter, sehingga `Content-Length: 61`.

```
0052: Content-Length: 61
0066: Content-Type: application/x-www-form-urlencoded
0000: name=Siti+Aminah&phone=0812-0000-1111&service=S0001&qty=3%2C5
```

Post/Redirect/Get: jawab POST dengan `303 See Other` dan `Location: /orders/B3001`; peramban mengambilnya dengan GET, dan memuat ulang hanya mengulang GET itu.

Kode 4.7 Dari `php/order.php`: beri nomor nota, tambahkan ke berkas CSV, lalu alihkan.

```
    $id = sprintf('B%04d', 3001 + count(orders()));   // B3001, B3002, ...
    $fh = fopen(STORE, 'a');
    fputcsv($fh, [$id, ...array_values($fields)], ',', '"', '');
    fclose($fh);
    // Post/Redirect/Get: 303 menyuruh peramban MENGAMBIL nota dengan GET.
    http_response_code(303);
    header('Location: /orders/' . $id);
```

Kode 4.7*: cabang POST tidak mencetak badan sama sekali. Mengambil nota dua kali tidak membuat apa pun: B3002 tidak ada. Java membuat* `handle` *menjadi* `synchronized`*; Flask menulis* `redirect(…, code=303)`*. Buku: Subbab 4.8.2–4.8.5.*

Keliru yang Menggoda: Tautan yang Membuat Nota

```
Created B3001
Created B3002
HTTP/1.1 200 OK
B3001 Siti
B3002 Siti
B3003 Siti
```

Dua kunjungan ke tautan itu, seperti yang dilakukan Reload, dan satu `curl -I` — `HEAD` yang tidak meminta isi apa pun — membuat tiga nota untuk satu penitipan.

GET didefinisikan aman, sehingga tab yang dipulihkan, pratinjau di aplikasi obrolan, dan *crawler* boleh mengulanginya.

UJI DIRI. "Tandai B2946 selesai" adalah POST, dijawab `303 See Other` dengan `Location` yang menunjuk ke halaman nota.

GET yang mengubah data adalah bug di server, bukan kesalahan pengunjung. Buku: Subbab 4.8.6.

Soal 4.7: Daftar Harga sebagai Tabel

> **SOAL 4.7.** Terbitkan daftar harga dua belas layanan sebagai dokumen HTML5 lengkap dengan tabel sungguhan — keterangan tabel, baris judul kolom, kode sebagai judul baris, harga bertitik ribuan — lalu bangkitkan dokumen yang sama dengan gelung dalam PHP, Java, dan Python, byte demi byte.

`th scope="row"` dan `scope="col"` menjadikan 12.000 *sel* baris S0002 dan kolom *Price (Rp)* — bagi pembaca layar, program, dan CSS.

Dua keputusan dengan tangan: *Wash, dry & fold* harus ditulis `&`, dan 12000 harus tampil sebagai `12.000`.

Halaman tulisan tangan adalah berkas statis; ketiga pembangkit berjalan di server dan mengirim persis berkas itu. Buku: Subbab 4.9.1 dan 4.9.2.

Baris-Baris yang Ditulis Gelung

Kode 4.8 Dari `php/prices.php`: satu `foreach` menulis kedua belas baris, dan `e()` meng-*escape* setiap nilai.

```
    <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>
<?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>
```

***Kode 4.8**: `foreach ($services as $code => [$name, $unit, $price])` mengurai setiap entri; `number_format($price, 0, ',', '.')` menghasilkan `12.000`. `run_all.sh` membandingkan ketiga pembangkit dengan berkas tulisan tangan: keluaran PHP, Java, dan Flask identik. Buku: Subbab 4.9.3–4.9.5.*

Daftar Harga di Peramban Sungguhan



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 4.5 Daftar harga di Chromium *headless*: kotak catatan, tabel dengan baris judul dan judul baris, baris ekspres berwarna merah, dan keterangan di bawah tabel.

***Gambar 4.5**: kotak catatan, baris judul, judul baris, baris ekspres berwarna merah, dan keterangan di bawah — warnanya adalah Soal 4.8. Buku: Subbab 4.9.3.*

Keliru yang Menggoda: Sel tanpa Baris

```
<table id="prices">
    <tbody><tr><th>Code</th><th>Service</th><th>Price</th>
    <td>S0001</td><td>Wash, dry &amp;amp; fold</td><td>8.000</td>
    <td>S0002</td><td>Express wash &amp;amp; fold, 1 day</td><td>12.000</td>
  </tr></tbody></table>
rows: 1
```

Peramban tidak menolak tabel berisi sel tanpa `tr`: standar HTML menetapkan cara memperbaikinya, dan perbaikan itu menyembunyikan kesalahannya.

Satu `tbody` sisipan, **satu** `tr`, sembilan sel berjajar: `tbody tr:nth-child(2)` tidak cocok dengan apa pun, dan pembaca layar mengumumkan satu baris sembilan kolom.

> **UJI DIRI.** *Carpet < 5 m^2* harus tampil sebagai `Carpet < 5 m²`: *escaping* milik saat keluaran, dikerjakan pembangkitnya, bukan orang yang mengetik datanya.

`dom_dump.py` bertanya kepada Chromium headless pohon apa yang dibangunnya, yang tidak sama dengan yang diketik. Buku: Subbab 4.9.6.

Soal 4.8: Aturan Mana yang Menang?

Tanpa peramban: (a) warna sel harga S0001 dan S0002; (b) warna sel lainnya; (c) perataan harga; (d) ukuran kotak catatan. Lalu periksa, dan jawab (a) lagi tanpa R5.

Kode 4.9 Dari `web/laundry/prices.css`: aturan-aturan yang menentukan warna dan perataan, bernomor R1 sampai R8.

```
/* 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; }
```

***Kode 4.9**: R10 memberi kotak catatan `width: 320px`, `padding: 12px 16px`, dan `border: 4px`. Buku: Subbab 4.10.1.*

Spesifisitas, Dibandingkan dari Kiri

$$(a_1, b_1, c_1) > (a_2, b_2, c_2) \Leftrightarrow a_1 > a_2 \ \vee \ (a_1 = a_2 \wedge b_1 > b_2) \ \vee \ (a_1 = a_2 \wedge b_1 = b_2 \wedge c_1 > c_2) \tag{4.1}$$

Tabel 4.6 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

*Persamaan (4.1) dan **Tabel 4.6**: a id; b kelas, atribut, dan pseudo-kelas; c tipe dan pseudo-elemen; satu id mengalahkan berapa pun banyaknya kelas,* $(1,0,0) > (0,9,9)$*. Urutan hanya memutus seri. Buku: Subbab 4.10.2.*

Contoh Terurai: *Cascade* dengan Tangan

(a) Sel harga S0001 cocok dengan R2, R3, R5, R6; yang tertinggi R5 (1,1,0): `#1d2433`. S0002 menambah R4 (0,1,2), masih di bawah: **R5 lagi**. R6 datang belakangan dan tetap kalah.

(b) Sel lain S0001: hanya R2, **abu-abu**. Sel S0002: R4 (0,1,2) mengalahkan R2, **merah**. Sel kode adalah `th`: mewarisi R1.

(c) R3 `.price` (0,1,0) mengalahkan `td` milik R8 (0,0,1) walaupun R8 lebih belakang: **rata kanan**.

Tanpa R5: harga S0001 menjadi **oranye** (R6 (0,1,1)); harga S0002 menjadi **merah** (R4 (0,1,2) mengalahkan R6 pada angka ketiga).

Chromium headless menjawab `rgb(29, 36, 51)`, `rgb(179, 38, 30)`, dan `rgb(85, 96, 122)`: `#1d2433`, `#b3261e`, `#55607a` — setiap jawaban benar. Buku: Subbab 4.10.2 dan 4.10.3.

Contoh Terurai: Selebar Apa Kotaknya?

$$W = w + p_{\text{kiri}} + p_{\text{kanan}} + b_{\text{kiri}} + b_{\text{kanan}} \tag{4.2}$$

CONTOH TERURAI. $W = 320 + 16 + 16 + 4 + 4 = 360$ piksel. Satu baris tingginya $16 \times 1.5 = 24$ piksel; catatannya terlipat menjadi dua baris, sehingga isinya 48 dan $H = 48 + 12 + 12 + 4 + 4 = 80$ piksel.

```
note box: 360 x 80 px on screen
table rows: 13 | head rows: 1 | caption: True
```

Persamaan (4.2) berlaku untuk `content-box`*; dengan* `box-sizing: border-box` *kotaknya tepat 320 piksel dan isinya mendapat* $320 - 40 = 280$*. Margin berada di luar ukuran kotak itu sendiri. Buku: Subbab 4.10.2 dan 4.10.3.*

Cascade sebagai Program

Kode 4.10 Dari `python/specificity.py`: spesifisitas sebagai tiga hitungan, dan pemenangnya menurut `(specificity, position)`.

```python
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]]))
```

***Kode 4.10**: Python membandingkan tuple persis seperti cascade membandingkan spesifisitas, dari kiri ke kanan, sehingga `max(…, key=…)` adalah aturannya. Versi JavaScript dan Java mencetak tujuh baris yang sama. Buku: Subbab 4.10.4.*

Soal 4.9: Perkiraan Harga Langsung

SOAL 4.9. Tampilkan perkiraan harga *selagi pelanggan mengetik*, tepat sampai ke rupiah: koma atau titik sebagai pemisah, paling banyak satu desimal untuk kg, bilangan bulat untuk layanan per buah, minimal 1 kg, dan Rp 10.000 untuk jemput-antar. Lalu hitung ulang di server.

$$T = P \cdot q + D, \qquad q = \begin{cases} \max(q_{\text{diketik}}, 1) & \text{untuk layanan per kg} \\ q_{\text{diketik}} & \text{untuk layanan per buah} \end{cases} \tag{4.3}$$

S0001 `3,5`: 8000 × 3.5 = 28000. S0003 `4,1`: 24600. S0001 `0,5`: minimal 1 kg, 8000. S0002 `2.5` dengan antar: 40000. S0008 `1,5`: ditolak.

Persamaan (4.3): "selagi mengetik" hanya dapat dikerjakan JavaScript, tetapi peramban milik pelanggan, sehingga harga yang ditagihkan dihitung ulang di server. Buku: Subbab 4.11.1 dan 4.11.2.

Uang yang Tepat: Hitung dalam Persepuluhan

Kode 4.11 Dari `web/laundry/estimate.js`: baca jumlahnya sebagai persepuluhan bulat, lalu hitung dengan bilangan bulat.

```
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");
}
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 4.11**: `4,1` adalah 41 persepuluhan, dan $6000 \times 41/10 = 24600$ tepat, karena setiap harga berakhiran 000. Java memakai `long`, Python `//`. Buku: Subbab 4.11.2–4.11.4.*

Uji Lolos; Versi yang Menggoda Tidak

```
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

price * kg        = 24599.999999999996
Math.floor(...)   = 24599
24600 + field     = 2460010000
price * Number('') = 0
```

Floating point memberi 24599.999999999996, dan `Math.floor` kehilangan satu rupiah; `+` pada string sebuah field menghasilkan 2.460.010.000; `Number("")` bernilai 0, pesanan gratis. Satu pola ketat dan aritmetika bilangan bulat menghindari ketiganya. Buku: Subbab 4.11.3 dan 4.11.5.

Soal 4.10: Nota dengan PHP

> **SOAL 4.10.** Hitung setiap baris titipan seorang pelanggan, lewati kode yang tidak dikenal, berhenti di penanda akhir, bayar sebanyak mungkin dari deposit, lalu daftarkan layanan dari yang termahal beserta kodenya. C0001 memiliki deposit Rp 52.000.

`S0001` 35 persepuluhan: $8000 \times 35/10 = 28000$. `S0008`: 35000. `S0X99`: tidak dikenal, lewati. `S0011` dua pasang: 90000. Penanda akhir: berhenti; `S0009` tidak pernah dicapai.

Total $28000 + 35000 + 90000 = 153000$; dari deposit $\min(52000,153000) = 52000$; sisa yang harus dibayar 101000.

*Kode tak dikenal adalah kesalahan di satu baris, jadi gelungnya **melewatinya** — `continue`. Penanda akhir berarti tidak ada baris lagi, jadi gelungnya **berhenti** — `break`. Buku: Subbab 4.12.1 dan 4.12.2.*

Gelungnya, dan Apa yang Dicetaknya

Kode 4.12 Dari `php/bill.php`: `break` berhenti, `continue` melewati, `intdiv` menjaga uang tetap tepat.

```
$total = 0;
foreach ($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));
}
```

```
-------------------------------------------------------------------
Total                                                 Rp 153.000
Paid from deposit                                      Rp 52.000
Still to pay                                          Rp 101.000
```

***Kode 4.12**: setiap angka sama dengan hitungan tangan. Buku: Subbab 4.12.3.*

Keliru yang Menggoda: `rsort`

Kode 4.13 `php/wrong_sort.php`: `rsort` mengurutkan harga dan mengganti kodenya dengan 0, 1, 2, dan seterusnya.

```php
<?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
Dearest service: 0
```

***Kode 4.13**: harganya berurutan dan kode-kodenya hilang; "layanan termahal" menjadi `0`. Jaga kunci dengan `asort`, `arsort`, `ksort`, `krsort`. Dengan `continue` menggantikan `break`, selimutnya ikut ditagih: total* 178000*, sisa Rp 126.000. Buku: Subbab 4.12.5.*

Soal 4.11: Formulir Tervalidasi dan Ter-*escape*

SOAL 4.11. Formulir penitipan yang mengirim ke dirinya sendiri. Nama: 2–60 huruf, boleh spasi, titik, apostrof, dan tanda hubung; ponsel: 10–13 angka diawali 08; layanan yang terdaftar; jumlah seperti Soal 4.9. Pesan di samping setiap *field* yang salah, isian tetap ada, status yang berkata "tidak dapat diterima", dan semuanya di-*escape*.

Tabel 4.7 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

***Tabel 4.7**: Ma'ruf adalah alasan apostrof diizinkan. Formulir yang tiba tetapi tidak dapat diterima mendapat `422 Unprocessable Content`. Formulirnya memakai POST, jadi programnya membaca `$_POST` dan tidak pernah `$_REQUEST`. Buku: Subbab 4.13.1 dan 4.13.2.*

Validasi Saat Tiba

Kode 4.14 Dari `php/dropoff.php`: satu pesan untuk setiap *field* yang salah; *array* kosong berarti semuanya benar.

```
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 4.14**: `\p{L}` adalah huruf dalam abjad apa pun, dan penanda `u` membuat PHP membaca UTF-8, sehingga Zoë lolos. `re` di Python menulis kelas yang sama sebagai `[^\W\d_]`. Tag script pada baris ketiga kembali sebagai `<script>…`, teks yang tidak pernah berjalan. Buku: Subbab 4.13.3.*

Keliru yang Menggoda: Divalidasi, Tidak Di-*escape*

```
    <p><label>Name <input name="name" value="" autofocus onfocus="alert(document.cookie)"></label> Please check
the name.</p>
wrong_dropoff.php  input attributes: name, value, autofocus, onfocus alerts: ['hijacked']
dropoff.php        input attributes: name, value                     alerts: []
```

Serangannya tidak memerlukan `<script>`, cukup sebuah tanda kutip: ia menutup `value` dan menambahkan `autofocus onfocus=…`. Namanya *ditolak*, dan serangannya tetap berjalan.

`strip_tags` tidak menolong: serangannya tanpa tag. Hanya escaping tanda kutip, `htmlspecialchars(…, ENT_QUOTES)`, yang menjaga nilai tetap di dalam atribut — penyaring selalu kurang satu karakter. Buku: Subbab 4.13.4 dan 4.13.5.

Soal-Soal Berdampingan

Tabel 4.8 Soal 4.1 sampai 4.6: 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

***Tabel 4.8**: enam soal pertama — situs, alamat, laporan, penerbitan, dan jaringan. Buku: Subbab 4.14.*

Soal-Soal Berdampingan (Lanjutan)

Tabel 4.9 Soal 4.7 sampai 4.11: gagasan rancangan yang diajarkan masing-masing dan jebakan yang diperlihatkan penyelesaian kelirunya.

Soal	Gagasan rancangan	Jebakan yang diperlihatkan penyelesaian keliru yang menggoda
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

***Tabel 4.9**: hampir setiap jebakan hanya muncul di tepi — sebuah awalan, permintaan yang diulang, tanda kutip, 4,1 kilogram — dan hampir setiap jawaban yang benar diamati, bukan diandaikan. Buku: Subbab 4.14.*

Profitina dalam Praktik

Satu alamat per halaman, kode status yang jujur: topik di balik `#fragmen` adalah satu halaman bagi *crawler*, dan `200 OK` yang berarti "tidak ditemukan" diindeks sebagai isi.

Struktur yang menyatakan apa sesuatu itu: tabel sungguhan, dan data terstruktur `schema.org` yang menyatakan sebuah usaha beserta harganya.

***Escaping* atas tulisan orang lain:** umpan komunitas, yang dialirkan dengan Server-Sent Events, membawa teks pengguna ke halaman orang lain.

Bukti alih-alih kata sifat: setiap skor datang bersama pengukurannya. Aplikasi memberi saran; pemilik yang memutuskan.

Studi kasus yang dapat dijalankan tetap Profitina Laundry, yang aplikasi jadinya ada di `ProfitinaLaundry/App`. Buku: Subbab 4.15.

Kesalahan yang Sering Terjadi

Tautan yang melupakan awalan; halaman palsu dari fragmen; pendapat di tempat yang memerlukan bukti; menerbitkan folder yang salah atau kata sandi.

Putusan yang disembunyikan di badan; mengubah data dengan GET.

Memercayai peramban memperbaiki markah; membaca CSS dari atas ke bawah; melupakan *padding* dan garis tepi dalam lebar.

Uang dalam *floating point*; `+` pada *field* formulir; harga yang hanya dihitung di peramban.

Mengurutkan sampai kuncinya hilang; memvalidasi tanpa *escaping*.

Buku mendaftar empat belas kesalahan, masing-masing dengan soal yang memperlihatkannya. Buku: Subbab 4.16.

Ringkasan: Lima Hal untuk Diingat

1. Delapan langkah menyelesaikan soal web kecil; langkah 3, apa berjalan di mana, mencegah kesalahan yang paling mahal. Amati dengan `curl`, *developer tools*, dan peramban *headless*.

2. Direktori berisi `index.html` memberi setiap halaman alamat bersih di *host* mana pun; tautan membawa awalan lengkap; laporan berdiri di atas pengukuran.

3. Kode status membawa putusannya; GET membaca dan boleh diulang; POST mengubah dan dijawab dengan `303`.

4. Tabel memerlukan baris dan judul sungguhan; *cascade* memilih menurut spesifisitas (Persamaan (4.1)), lalu urutan; kotak adalah isi ditambah *padding* ditambah garis tepi (Persamaan (4.2)).

5. Uang dalam bilangan bulat, dihitung ulang di server; `continue` melewati, `break` menghentikan, `arsort` menjaga kunci; validasi saat tiba, *escaping* saat keluar.

Kebiasaan melihat dengan instrumen adalah hal paling berharga yang ditinggalkan kesebelas soal ini. Buku: Subbab 4.17.

Pertemuan Berikutnya

Bab 5 — PHP dan MySQL, serta Menjalankan Server. Basis data Profitina Laundry, SQL, *prepared statement* melawan injeksi SQL, dan server sungguhan.

Halaman-halaman Bab 3 dan 4 berhenti membaca konstanta dan berkas CSV, dan mulai membaca tabel `Service`, `Bill`, dan `Customer`.

Sebelum itu: kerjakan Soal Latihan 4.2, 4.5, 4.6, 4.7, dan 4.10, dan pasang XAMPP atau MySQL seperti yang dijelaskan Subbab 1.16.5.

Bab 5 menjawab pertanyaan yang ditunda setiap soal hari ini: di mana datanya tinggal? Buku: Subbab 4.18 dan 4.19.

Kuliah 5 — PHP dan MySQL, serta Menjalankan Server

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 5

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini halaman kita berhenti lupa: peladen yang kita kendalikan, ingatan di antara permintaan, basis data, dan satu aturan keamanan yang paling penting. Buku: Bab 5, Subbab 5.1–5.22.

Cakupan Pertemuan Ini

1. **Peladen yang Anda kendalikan** — Apache, PHP, dan MySQL; XAMPP, LAMP, *document root*, *port*, `php.ini`.
2. **Ingatan di antara permintaan** — *redirect*, *cookie*, sesi, dan keluar dari sesi.
3. **Basis data dan bahasanya** — tabel dan kunci, tujuh tabel Profitina Laundry, SQL dari `CREATE` hingga `GROUP BY`, transaksi.
4. **PHP bertemu basis data** — `mysqli`, PDO dengan *prepared statement*, CRUD untuk pelanggan dan tagihan.
5. ***SQL injection*** — diperagakan dengan sebelas karakter, lalu ditutup untuk selamanya; kode yang sama dalam Java dan Python.

Urutannya mengikuti ketergantungan: tidak ada basis data tanpa peladen, tidak ada halaman aman tanpa prepared statement. Buku: Subbab 5.1.

Posisi Kita Sejauh Ini

Bab 3 memberi kita PHP: program yang keluarannya halaman, `$_GET` dan `$_POST`, validasi, dan *escaping* saat nilai masuk ke HTML.

Kelemahannya: begitu skrip selesai, semua yang diketahuinya hilang. Kasir yang sudah masuk di satu halaman menjadi orang asing di halaman berikutnya.

Profitina Laundry adalah sebuah usaha: tiga gerai di Surabaya, empat belas karyawan, 256 pelanggan, hampir tiga ribu tagihan setahun.

Tagihan mana di Keputih yang masih menunggu? Berapa pendapatan Rungkut pada bulan Maret? Siapa yang masih berutang? Dengan basis data, masing-masing cukup satu kalimat SQL.

> **WAWASAN KUNCI: INTI BAB INI.** Mulai sekarang semuanya tentang *state*: fakta yang bertahan lebih lama daripada permintaan yang menciptakannya. Sesi bertahan satu kunjungan; baris basis data bertahan sampai seseorang menghapusnya.

Kebanyakan bug yang sulit dalam pemrograman web ternyata berupa kekeliruan antara kedua jenis state ini. Buku: Subbab 5.1.

Web Server Adalah Tiga Program

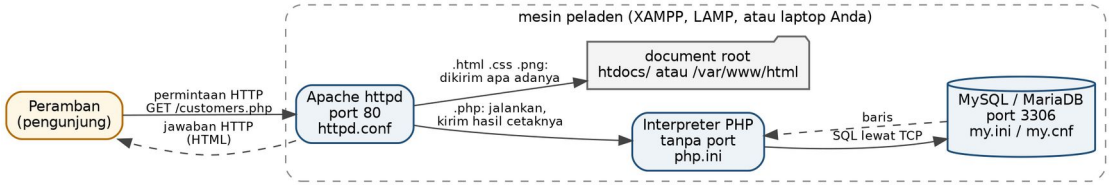
DEFINISI. Web server (peladen web) mendengarkan pada sebuah *port* dan menjawab HTTP (Apache, Nginx, IIS). **Penerjemah bahasa** menjalankan kode Anda (PHP); ia tidak punya *port*. **Peladen basis data** menyimpan data dan menjawab SQL pada *port*-nya sendiri (MySQL atau MariaDB).

Apache mengangkat telepon, PHP berpikir, MySQL mengingat.

Halaman yang tidak mau dimuat menunjuk ke Apache; halaman kosong atau kode PHP yang tampak menunjuk ke PHP; "could not connect" atau "access denied" menunjuk ke MySQL.

Mengetahui program mana yang mana mengubah satu sore menatap halaman kosong menjadi perbaikan satu menit. Buku: Subbab 5.2.1.

Tiga Program dalam Gambar



Gambar 5.1 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.

Pada Gambar 5.1 peramban tidak pernah melihat PHP atau MySQL: ia hanya berbicara dengan Apache, yang menyerahkan berkas `.php` kepada PHP, yang berbicara dengan basis data lewat koneksinya sendiri. Buku: Subbab 5.2.1.

Port dan Berkas Konfigurasi

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`

Tabel 5.1: hampir setiap masalah pemasangan selesai dengan menyunting salah satu dari tiga berkas di kolom terakhir, lalu menyalakan ulang program yang membacanya. Buku: Subbab 5.2.1.

Memasang Perangkatnya

Windows: XAMPP ke `C:\xampp`; nyalakan Apache, lalu MySQL; periksa `http://localhost/` dan `http://localhost/phpmyadmin/`; tambahkan `C:\xampp\php` dan `C:\xampp\mysql\bin` ke **Path**.

macOS: XAMPP dari berkas `.dmg` dan *manager-osx*, atau Homebrew: `brew install php mariadb`.

Linux (LAMP): `sudo apt install apache2 php libapache2-mod-php php-mysql php-sqlite3 php-mbstring mariadb-server`, lalu `sudo systemctl enable --now apache2 mariadb`.

Tanpa memasang apa pun: `php -S localhost:8080` melayani sebuah folder, dan SQLite menyimpan seluruh basis data dalam satu berkas, `laundry.db`.

> **Pasang satu *web server*, bukan dua.** “Port 80 in use by another application” berarti IIS, Skype, atau Apache kedua sudah lebih dulu memakainya.

Pengguna `root` MySQL pada XAMPP memiliki kata sandi kosong: wajar di laptop, tidak dapat diterima di tempat lain. Document root berpindah antarsistem; tiga program, port-nya, dan `http://localhost/` tidak. Buku: Subbab 5.2.2–5.2.5 dan 5.2.12.

Konfigurasi Apache

Kode 5.1 mengumpulkan segelintir direktif `httpd.conf` yang penting sekarang.

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, `DocumentRoot` folder yang menjadi situs web, `DirectoryIndex` berkas yang dikirim untuk sebuah folder. Apache membaca berkas ini sekali, saat menyala: sunting, nyalakan ulang, dan baca log galat bila ia menolak. Buku: Subbab 5.2.6.

Document Root

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*

CONTOH TERKERJAKAN. Halaman yang disimpan sebagai `C:\xampp\htdocs\profitina\web\customers.php` menghasilkan 404. Buang *document root*, ubah `\` menjadi `/`, awali dengan `http://localhost/`: **`http://localhost/profitina/web/customers.php`**.

Baris terakhir Tabel 5.2 adalah sifat keamanan: simpan `config.php`, yang memuat kata sandi, di luar document root dan `require` dari PHP, yang membaca disk secara langsung. Buku: Subbab 5.2.6.

Port, dan `php.ini` untuk Pengembangan

Hanya satu program yang boleh mendengarkan pada satu *port*: `netstat -ano | findstr :80` (Windows) atau `sudo lsof -i :80` menemukan program lainnya; atau pindahkan Apache ke `Listen 8080`.

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
```

Pada Kode 5.2, "mengaktifkan ekstensi" berarti menghapus titik koma di depannya; tanpa `date.timezone`, setiap cap waktu Surabaya tertinggal tujuh jam. Di peladen publik, galat disembunyikan dan dicatat ke log. Buku: Subbab 5.2.7 dan 5.2.8.

Redirect

DEFINISI. Redirect adalah jawaban tanpa halaman, hanya perintah: status **302 Found** dan header `Location:`. Peramban lalu membuat permintaan kedua yang terpisah.

Kode 5.3 `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
<?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;
```

Kode 5.3 menunjukkan dua aturan: `header()` harus berjalan sebelum keluaran apa pun (“headers already sent”), dan `exit` harus mengikutinya, karena `header()` hanya mengantrekan sebuah header. Buku: Subbab 5.3.1.

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*.

Rancangan yang baik: peladen *stateless* dapat dinyalakan ulang atau digandakan di belakang *load balancer* tanpa ada yang menyadarinya.

Ingatan tinggal di salah satu dari dua tempat. **Di klien:** *cookie*. **Di peladen:** sesi, yang nomor tiketnya bepergian di dalam *cookie*.

Keranjang belanja menuju kasir, kasir yang masuk sekali dan bekerja satu jam: aplikasilah yang harus membangun ingatan itu. Buku: Subbab 5.3.2.

Cookie: Membuat, Membaca, Menghapus

> **DEFINISI. Cookie** adalah sepotong kecil teks bernama yang diminta peladen untuk disimpan peramban (`Set-Cookie:`) dan dikirim kembali pada setiap permintaan berikutnya ke situs yang sama (`Cookie:`).

Kode 5.4 Dari `php/state/cookies.php`: `setcookie()` sebelum keluaran apa pun untuk membuat, dan waktu kedaluwarsa di masa lalu untuk menghapus.

```
$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);
}
```

Sesudah Kode 5.4, halaman mencetak `$_COOKIE`, yaitu cookie yang dibawa permintaan ini, sebagai teks biasa agar transkripnya mudah dibaca. Buku: Subbab 5.3.3.

Percakapan *Cookie*

```
$ 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
```

*Jawaban pertama memasang dua cookie dan menerima nol: `$_COOKIE` berisi apa yang dikirim peramban, sehingga cookie baru muncul pada permintaan **berikutnya**. Menghapus memasang nilai `deleted` dengan kedaluwarsa tahun 1970. Pasang `HttpOnly`, `Secure`, dan `SameSite=Lax`. Buku: Subbab 5.3.3.*

Apa yang Salah dengan *Cookie*

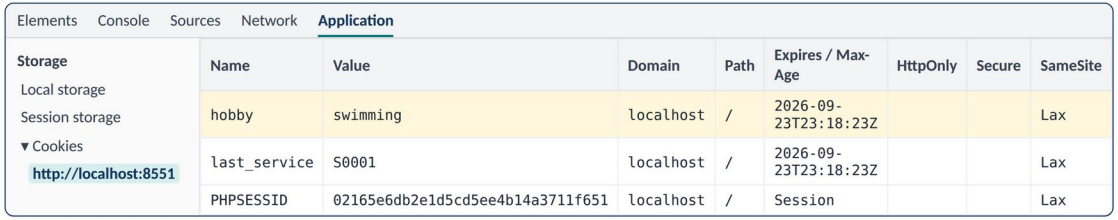
Tabel 5.3 Empat masalah klasik cookie. Yang kedua adalah masalah yang berujung menjadi insiden keamanan.

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

> **Jangan pernah menaruh di *cookie* sesuatu yang tidak boleh diubah pengunjung.** Bukan harga, bukan id pelanggan yang memberi akses, bukan `is_owner=0`.

Baris kedua Tabel 5.3 adalah yang berubah menjadi insiden keamanan. Cookie boleh memuat preferensi yang tidak berbahaya dan nomor tiket yang maknanya hanya diketahui peladen. Buku: Subbab 5.3.4.

Cookie di Mata Pengunjung



Gambar 5.2 Cookie milik `http://localhost:8551` di alat pengembang peramban (*Application* → *Storage* → *Cookies*), setelah `cookies.php?action=set` dan `session.php` dijalankan. Tidak ada isi cookie yang tersembunyi dari orang di depan komputer.

Gambar 5.2: F12 memperlihatkan setiap nama dan setiap nilai dalam teks biasa kepada siapa saja yang duduk di depan komputer, dan masing-masing dapat disunting. Buku: Subbab 5.3.4.

Sesi

DEFINISI. Session (sesi) adalah rangkaian permintaan dari satu peramban yang diperlakukan peladen sebagai satu kesatuan. PHP memberinya **session id**, menyimpan tempat simpanan pribadi di peladen, dan menampilkannya sebagai `$_SESSION`.

Kode 5.5 `php/state/session.php`: penghitung kunjungan yang disimpan di sesi. Angkanya naik hanya jika `$_SESSION` benar-benar bertahan dari satu permintaan ke permintaan berikutnya.

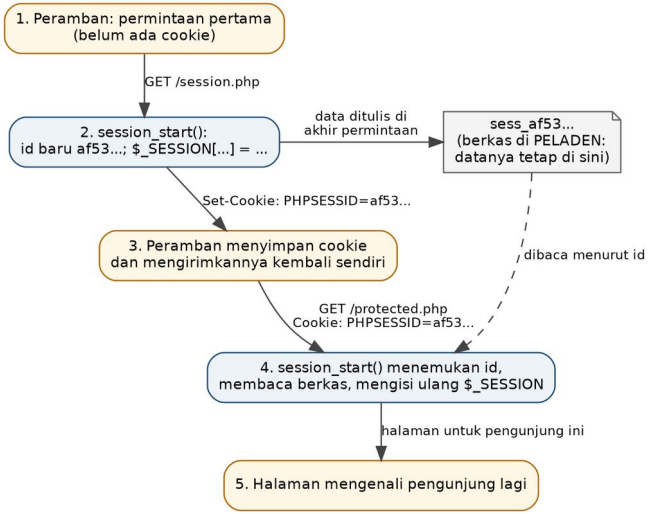
```php
<?php
// session.php: penghitung kunjungan yang disimpan di sesi, bukan di cookie.
declare(strict_types=1);
session_start();

$_SESSION['visits'] = ($_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;
```

Pada Kode 5.5 angka hanya naik bila `$_SESSION` benar-benar bertahan dari satu permintaan ke permintaan berikutnya. `session_start()` harus mendahului keluaran apa pun. Buku: Subbab 5.3.5.

Cara Kerja Sesi



Gambar 5.3 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.

*Gambar 5.3: `session_start()` membuat id dan mengirimkannya dalam cookie; datanya ditulis ke berkas di peladen; `session_start()` berikutnya mengisi ulang `$_SESSION`. **Data tidak pernah meninggalkan peladen.** Buku: Subbab 5.3.6.*

Sesi, Diamati dengan `curl`

```
$ 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
```

Wadah cookie yang sama memberi id yang sama dan penghitung 2, lalu mencapai halaman terlindung; permintaan tanpa wadah adalah peramban baru: id baru, penghitung 1. Sesi milik peramban, bukan milik orang. Siapa pun yang mengetahui session id menjadi pengunjung itu. Buku: Subbab 5.3.6.

Keluar dari Sesi

Kode 5.6 `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;
```

> **WAWASAN KUNCI: COOKIE ATAU SESI?** Preferensi yang tidak berbahaya bila diubah: *cookie*. Apa pun yang menyatakan siapa seseorang atau apa yang boleh dilakukannya: sesi.

Kode 5.6 mengosongkan larik, menghapus cookie dengan memasangnya berwaktu lampau, dan memusnahkan simpanan di peladen. Menaruh id di URL (`?PHPSESSID=`) dimatikan pada PHP modern, karena alamat disalin dan dicatat. Buku: Subbab 5.3.7.

Tabel, Baris, dan Kolom

Buku catatan di meja kasir binatu adalah sebuah tabel: satu kolom untuk setiap jenis fakta, satu baris untuk setiap kantong. Basis data relasional adalah buku itu yang dibuat cermat.

> **DEFINISI. Tabel** (*relation*) adalah himpunan baris dengan kolom bernama yang sama. **Kolom** memiliki nama dan **tipe**. **Baris** adalah satu set nilai lengkap, satu untuk setiap kolom.

Relasional menamai gagasan Codd tahun 1970: fakta dihubungkan *hanya* dengan mencocokkan nilai. Tagihan `B2931` memuat `C0056`; tabel lain memuat `C0056` di samping Gilang Nasution.

Baris adalah himpunan: tidak berurutan kecuali Anda memintanya dengan `ORDER BY`. `C0056` tetap `C0056` di mesin mana pun, di salinan mana pun, pada tahun berapa pun; pointer rusak ketika benda berpindah. Buku: Subbab 5.4.1 dan 5.4.2.

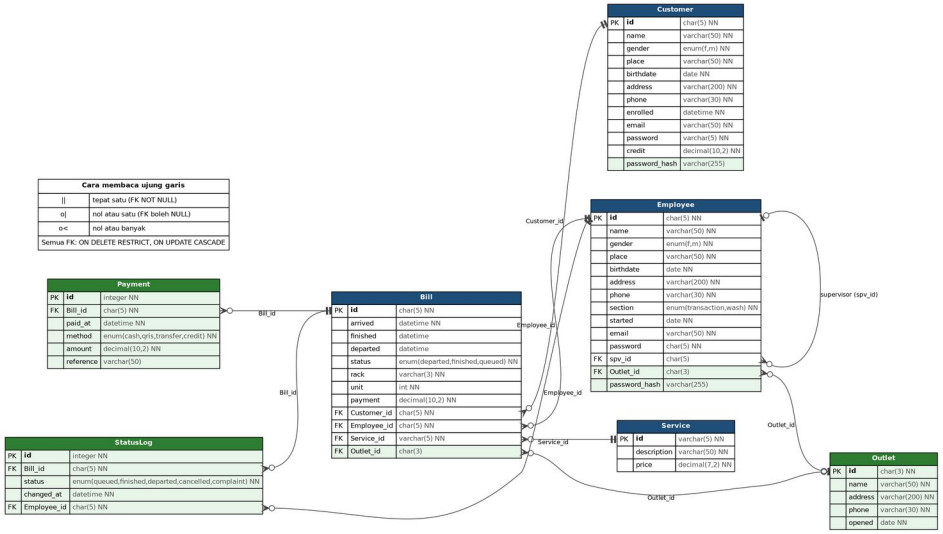
Kunci

DEFINISI. Primary key (kunci primer) bernilai berbeda dan tidak kosong di setiap baris: dengannya baris dinamai. **Foreign key** (kunci tamu) adalah kolom yang nilainya harus merupakan *primary key* sebuah baris di tabel lain; basis data menegakkannya.

CONTOH TERKERJAKAN. `Customer.id = 'C0056'` adalah Gilang Nasution. `Bill.Customer_id` merujuknya: tagihan `B2931` membawa `'C0056'`, dan satu nilai itu adalah seluruh hubungannya. Tagihan untuk `C9999` bukan sekadar tidak ditemukan; ia ditolak.

Binatu memakai kode surrogate seperti `C0056`, bukan kunci natural seperti nomor telepon, karena kunci natural berubah dan kunci yang berubah menyeret setiap foreign key di belakangnya. Buku: Subbab 5.4.3.

Tujuh Tabel Profitina Laundry



Gambar 5.4 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”.

Gambar 5.4: setiap kotak sebuah tabel, setiap garis sebuah foreign key; kaki gagak berarti “banyak”, garis ganda “tepat satu”, lingkaran “mungkin tidak ada”. Buku: Subbab 5.4.4.

Satu Baris Adalah …

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

Bila Anda tidak dapat menyelesaikan kalimat “satu baris adalah …” untuk sebuah tabel, tabel itu salah. `Payment` dan `StatusLog` adalah tabel, bukan kolom, karena satu tagihan dapat memiliki beberapa (Tabel 5.4). Buku: Subbab 5.4.4.

Jumlah Tagihan dan Sisa Bayar

`Bill.payment` disimpan walaupun `Service` memuat harganya, karena ia adalah *jumlah yang ditagihkan pada hari itu* (Persamaan (5.1)):

$$payment = price \times unit \tag{5.1}$$

Uang yang masih terutang mengikuti darinya dan dari tabel `Payment` (Persamaan (5.2)):

$$balance = payment - \sum_{p \in Payment,\, p.Bill_id = id} p.amount \tag{5.2}$$

Bila cuci, kering, dan lipat naik dari Rp 8.000 menjadi Rp 9.000, nota tahun lalu tidak boleh berubah: nilai sejarah dicatat, bukan dicari ulang. Buku: Subbab 5.4.4.

Contoh Terkerjakan 1: Buku Kasir

Kode 5.7 membangun ulang buku kasir satu tabel dari data sungguhan, lalu melakukan satu perubahan biasa.

Kode 5.7 `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;
```

`CREATE TABLE ... AS SELECT` membuat tabel baru dari jawaban sebuah kueri; pelanggan dan harga diulang di setiap baris. Buku: Subbab 5.4.5.

Contoh Terkerjakan 1: Satu Perubahan, Dua Kebenaran

```
Query OK, 1 row affected

+------------------+----------------+
| customer         | phone          |
+------------------+----------------+
| Endah Ika Rahman | 0877-0000-2053 |
| Endah Ika Rahman | 0813-0000-9999 |
+------------------+----------------+
```

> **WAWASAN KUNCI: OBATNYA, NORMALISASI.** Fakta tentang *satu hal* disimpan di baris tentang *hal lain*. Letakkan setiap fakta di tabel tentang hal yang dijelaskannya, dan hubungkan tabel-tabel dengan kunci.

*Sesudah satu `UPDATE` yang wajar, Endah Ika Rahman memiliki dua nomor telepon: **anomali pembaruan**; anomali penghapusan dan penyisipan adalah kerabatnya. 1NF: satu nilai per sel; 2NF: bergantung pada seluruh kunci; 3NF: tidak ada kolom yang bergantung pada kolom bukan-kunci lain. Buku: Subbab 5.4.5.*

Tipe Adalah Janji, *Constraint* Adalah Aturan

Tabel 5.5 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`

DEFINISI. `NULL` berarti *tidak ada nilai yang tercatat*. Ia tidak sama dengan apa pun, bahkan dengan `NULL`: tulis `finished IS NULL`, jangan `finished = NULL`.

Tabel 5.5: uang adalah `DECIMAL`, tidak pernah `FLOAT`; nomor telepon adalah teks. Constraint (`NOT NULL`, `PRIMARY KEY`, `UNIQUE`, `FOREIGN KEY`, `CHECK`) adalah aturan yang tidak boleh dilanggar siapa pun. Buku: Subbab 5.5.3 dan 5.5.4.

Jantung Skema: `Bill`

Kode 5.8 *Constraint* tabel `Bill` dalam `sql/24_create_bill.mysql.sql`: aturan usaha binatu, ditulis sekali dan ditegakkan untuk setiap program.

```
    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;
```

Di atas constraint Kode 5.8 ada dua belas kolom: `arrived` berbawaan saat ini, `finished` dan `departed` tetap `NULL`. Tiga `CHECK` terakhir menyandikan riwayat tagihan: tak bisa selesai sebelum tiba, dan statusnya harus cocok dengan waktunya. Buku: Subbab 5.5.5.

Foreign Key dan `ON DELETE`

Kode 5.9 `sql/25_bill_foreign_keys.mysql.sql`: empat *foreign key* milik `Bill`, dalam gaya rancangan asli, satu `ALTER TABLE` per rujukan.

```sql
-- 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;
```

DEFINISI. `ON DELETE RESTRICT` menolak menghapus baris yang masih dirujuk; `CASCADE` ikut menghapus baris yang merujuk; `SET NULL` melupakan rujukannya. `ON UPDATE CASCADE` menyalin kunci yang berubah ke setiap baris yang merujuk.

Bacalah Kode 5.9 sebagai kalimat: setiap `Bill.Customer_id` harus merupakan `id` di `Customer`. Binatu memilih `RESTRICT` di mana-mana: bila basis data menolak, manusialah yang memutuskan. Buku: Subbab 5.5.6.

`INSERT`: Pelanggan Baru Datang

Kode 5.10 `sql/06_insert.sql`: satu pelanggan baru dan satu tagihan baru, lalu melihat keduanya.

```sql
-- 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';
```

Selalu sebutkan kolom pada Kode 5.10; kolom yang dilewati memakai nilai bawaannya. Angka 24000 pada tagihan adalah Persamaan (5.1) yang dihitung tangan: $8000 \times 3 = 24000$. *Tanda terimanya:* `Query OK, 1 row affected`. *Buku: Subbab 5.7.1.*

Ketika Aturan Dilanggar

Empat pernyataan terlarang di buku mencoba `C0001` ganda, tagihan untuk `C9999`, tagihan nol kilogram, dan menghapus pelanggan yang masih punya tagihan. MariaDB menolak keempatnya:

```
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)')
```

> **WAWASAN KUNCI: TIGA KESALAHAN YANG LAYAK DIHAFAL.** 1062 kunci ganda; 1452 *foreign key* merujuk ke ketiadaan; 1451 penghapusan terhalang baris yang merujuk ke sini. PDO melaporkan ketiganya sebagai SQLSTATE `23000`.

4025 menyebut `CHECK` yang gagal, `Bill_unit_pos`; karena itulah constraint layak diberi nama. SQLite berkata `UNIQUE constraint failed`, `FOREIGN KEY constraint failed`, dan `CHECK constraint failed`. Setiap penolakan adalah bug yang tertangkap sebelum menjadi data yang salah. Buku: Subbab 5.7.2.

UPDATE dan DELETE dengan Aman

Kode 5.11 `sql/08_update.sql`: lihat, ubah, lihat lagi, dan coba perubahan yang sama untuk kedua kalinya.

```sql
-- 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';
```

`UPDATE` tanpa `WHERE` mengubah setiap baris. Tulis `WHERE` lebih dulu, jalankan sebagai `SELECT`, baru ubah menjadi `UPDATE` atau `DELETE`.

*Pada Kode 5.11 `WHERE` juga menyebut statusnya, sehingga `UPDATE` kedua yang sama melaporkan **0 baris**: orang lain sudah menyelesaikan tagihan ini. Tagihan yang sudah dibayar tidak dapat dihapus (`Payment_Bill` bersifat `RESTRICT`); tagihan yang dibatalkan tetap menyimpan barisnya. Buku: Subbab 5.7.3 dan 5.7.4.*

`SELECT ... WHERE`: Antrean Keputih

Kode 5.12 `sql/03_where.sql`: antrean gerai Keputih, yang terlama lebih dulu.

```sql
-- 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;
```

Kode 5.12 mengembalikan sembilan baris, `B2909` sampai `B2951`: sembilan kantong menunggu di sembilan rak. `AND`, `OR`, `NOT`, `BETWEEN`, `IN`, `LIKE 'Ayu%'`, dan `IS NULL` memilih baris; kesamaan ditulis dengan satu `=`. Klausa ditulis `SELECT`, `FROM`, `WHERE`, `GROUP BY`, `HAVING`, `ORDER BY`, `LIMIT`, tetapi dievaluasi mulai dari `FROM`. Buku: Subbab 5.8.1–5.8.3.

Join: Satu Nota dari Tiga Tabel

> **DEFINISI. Join** memasangkan baris sebuah tabel dengan baris tabel lain di mana pun syarat `ON`, hampir selalu "*foreign key* sama dengan *primary key*", bernilai benar.

Kode 5.13 `sql/10_join.sql`: satu tagihan dengan nama pelanggan serta deskripsi dan harga layanannya.

```sql
-- 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
```

Pada Kode 5.13, `b`, `c`, dan `s` adalah alias. Karena `Customer.id` adalah primary key, setiap tagihan menemukan tepat satu pelanggan. $8000 \times 9 = 72000$ *memeriksa Persamaan (5.1) sekali lagi. Buku: Subbab 5.9.1 dan 5.9.2.*

`LEFT JOIN`: Pertahankan Baris Tanpa Pasangan

Kode 5.14 `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
```

Kode 5.14 menemukan tiga pelanggan yang tidak pernah membawa kantong, hal yang tidak mampu dilakukan inner join. Hitung `COUNT(b.id)`, bukan `COUNT(*)`: baris `NULL` milik pelanggan tanpa tagihan akan terhitung 1. Buku: Subbab 5.9.3.

`GROUP BY`: Apa yang Menghasilkan Uang?

> **DEFINISI. Fungsi agregat** (`COUNT`, `SUM`, `AVG`, `MIN`, `MAX`) mengubah banyak baris menjadi satu nilai. `GROUP BY c` membuat satu tumpukan per nilai `c` dan mengagregasi setiap tumpukan.

Kode 5.15 `sql/14_group_service.sql`: pendapatan per layanan selama setahun.

```sql
-- 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;
```

Setiap kolom dalam daftar `SELECT` pada Kode 5.15 dikelompokkan atau diagregasi. Cuci, kering, dan lipat menghasilkan Rp 67.632.000, separuh usaha; periksa setrika dengan tangan: $1251 \times 6000 = 7{,}506{,}000$. *Buku: Subbab 5.10.1 dan 5.10.2.*

Contoh Terkerjakan 2: Siapa yang Masih Berutang?

Kode 5.16 `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;
```

Kode 5.16 adalah Persamaan (5.2) dalam SQL. Subkueri menjumlahkan pembayaran per tagihan sebelum join: menggabungkan `Payment` *dan* `StatusLog` *secara langsung akan melipatgandakan baris,* $2 \times 3 = 6$. `LEFT JOIN` *mempertahankan tagihan yang belum dibayar;* `COALESCE` *mengubah* `NULL`*-nya menjadi 0. Buku: Subbab 5.10.5.*

Contoh Terkerjakan 2: Jawabannya

```
+-------+--------------------+---------+------+---------+
| 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
```

Periksa `B2931` dengan tangan: $9 \times 8000 = 72000$ ditagihkan, belum ada yang dibayar, jadi sisanya $72000 - 0 = 72000$.

Sepanjang tahun ada 38 tagihan yang masih bersisa, seluruhnya Rp 1.529.000, sebagian besar masih di toko. Buku: Subbab 5.10.5.

Transaksi

DEFINISI. Transaction (transaksi) adalah sekelompok pernyataan yang diperlakukan sebagai satu tindakan tak terbagi: `BEGIN` membukanya, `COMMIT` membuat setiap perubahan permanen sekaligus, `ROLLBACK` membatalkan setiap perubahan. **ACID:** *atomic*, *consistent*, *isolated*, *durable*.

Kode 5.17 Transaksi dalam `sql/18_transaction_commit.sql`: tagihan, entri log pertamanya, dan uang muka, bersama-sama atau tidak sama sekali.

```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;
```

Pada Kode 5.17 dua bed cover seharga Rp 70.000 menurut Persamaan (5.1); Rp 35.000 dibayar; Persamaan (5.2) menyisakan Rp 35.000. Dengan `ROLLBACK`, hitungan tagihan menjadi 2.953 sebelum, 2.954 di dalam, 2.953 sesudah. Buku: Subbab 5.11.1 dan 5.11.2.

PHP dan MySQL Cara Klasik

DEFINISI. Result set (himpunan hasil) adalah jawaban atas `SELECT`, dibaca satu baris demi satu baris lewat kursor; setiap pengambilan menyerahkan baris lalu maju, dan `null` bila tidak ada lagi.

Kode 5.18 Dari `php/mysqli/query.php`: `mysqli_query`, pemeriksaan kegagalan, dan perulangan `while` atas `fetch_row()`.

```
$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;
}
```

Kode 5.18 membaca empat baris sebagai `$row[0]`; sesudah `data_seek(0)` program membacanya lagi sebagai `$row['description']` dengan `MYSQLI_ASSOC`; utamakan nama. `free()` dan `mysqli_close()` membereskannya. Pengaturan diambil dari `config.php`. Buku: Subbab 5.12.1–5.12.4.

PDO: Satu Antarmuka, Banyak Basis Data

DEFINISI. PDO (*PHP Data Objects*) adalah satu antarmuka untuk MySQL, MariaDB, SQLite, PostgreSQL, dan lainnya; yang berubah hanya **DSN**, *data source name*.

Kode 5.19 Awal `connect()` dalam `php/pdo/Db.php`: tiga opsi yang memperbaiki bawaan yang buruk, dan DSN MySQL.

```
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;
    }
```

Kode 5.19: kegagalan melempar eksepsi, baris diberi kunci nama kolom, prepared statement sungguhan. Cabang SQLite menyalin `laundry.db` dan menjalankan `PRAGMA foreign_keys = ON`. Polanya: `prepare` dengan `?`, `execute([...])`, lalu `fetch`, `fetchAll`, atau `rowCount`. Dengan mode galat senyap, kegagalan hanyalah `false` yang tak diperiksa siapa pun. Buku: Subbab 5.14.1, 5.14.2, dan 5.14.5.

Pelanggan: Membuat

Kode 5.20 `createCustomer()` dari `php/pdo/Customers.php`: sembilan nilai diikat ke sembilan *placeholder*.

```
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;
}
```

Pada Kode 5.20, `'*****'` dan `0` adalah konstanta pemrogram, bukan masukan; `findCustomer()`, `updateCustomerPhone()`, dan `deleteCustomer()` berbentuk sama. Pencarian mengikat `'%' . $text . '%'` dan `LIMIT` sebagai `PDO::PARAM_INT`; menghapus `C0094`, yang punya tagihan, ditolak dengan SQLSTATE `23000`. Buku: Subbab 5.14.3.

`createBill()`: Harga Dihitung

Kode 5.21 Awal `createBill()` dalam `php/pdo/Bills.php`: mulai transaksi, baca harga, terapkan rumus harga.

```
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
```

Harga bukan argumen Kode 5.21: `createBill()` membacanya dari `Service` dan menerapkan Persamaan (5.1) sendiri, sehingga tak ada formulir yang dapat mengirim tagihan Rp 1. Berikutnya `INSERT` untuk baris `Bill` itu sendiri. Buku: Subbab 5.14.4.

`createBill()`: *Commit* atau *Rollback*

Kode 5.22 Akhir `createBill()`: log status, pembayaran opsional, `commit()`, dan `rollBack()` pada setiap `Throwable`.

```
        $pdo->prepare(
            "INSERT INTO StatusLog (Bill_id, status, changed_at, Employee_id)
             VALUES (?, 'queued', ?, ?)")
            ->execute([$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;
    }
}
```

Bentuk Kode 5.22 tidak pernah berubah: `beginTransaction()`; pekerjaan di dalam `try`; `commit()` paling akhir; `rollBack()` dan `throw $e` di dalam `catch (Throwable $e)`. Buku: Subbab 5.14.4.

Riwayat Sebuah Tagihan, Dijalankan

```
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: `Bill` dan `StatusLog` sudah ditulis, pembayaran gagal karena `gopay` tidak diizinkan, dan hitungan menunjukkan tidak ada sisa tagihan itu. `finishBill()` mengembalikan `false` pada kali kedua. Buku: Subbab 5.14.4.

Halaman Masuk yang Terbuka Lebar

Kode 5.23 Dari `php/injection/unsafe.php`: pemeriksaan masuk dan pencarian status tagihan yang merekatkan ketikan pengunjung ke dalam teks SQL.

```
// 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);
```

*Tidak ada yang tampak mengkhawatirkan pada Kode 5.23. Sebuah kueri punya **instruksi**, yang Anda tulis, dan **data**, yang diketik orang asing; merekatkan string mencampurkan keduanya, dan basis data tidak dapat membedakan bagian mana berasal dari siapa. Buku: Subbab 5.15.1.*

Sebelas Karakter

```
=== 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
```

Apostrof pertama menutup string; sisanya berlanjut sebagai SQL (Persamaan (5.3)):

$$(email = \text{'kasir1@...'} \wedge password = \text{''}) \vee (\text{'1'} = \text{'1'}) \tag{5.3}$$

`AND` mengikat lebih dulu daripada `OR`: separuh kiri salah untuk setiap baris, separuh kanan benar untuk setiap baris, sehingga karyawan pertama, sang pemilik, dibiarkan masuk. `--` diikuti spasi menjadikan uji kata sandi sebuah komentar. Buku: Subbab 5.15.1 dan 5.15.2.

Membaca Tabel yang Tak Pernah Disebut Halaman

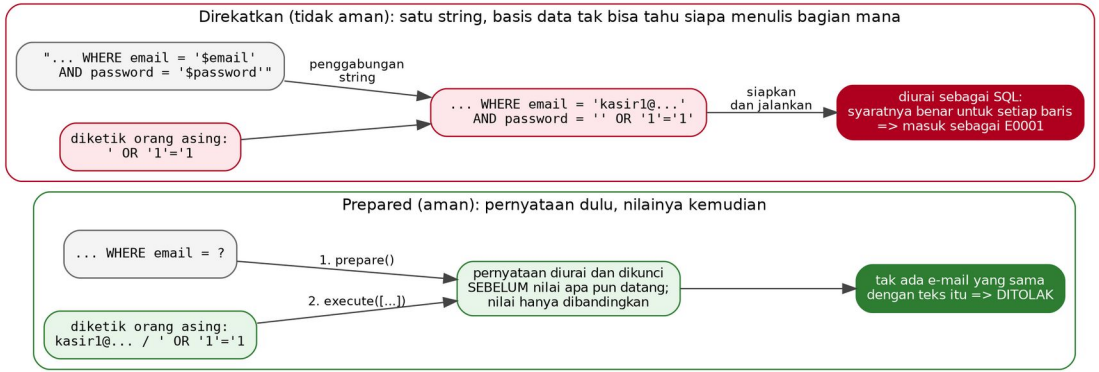
```
=== UNION attack
typed bill id  : x' UNION SELECT name, phone FROM Customer ORDER BY 1 LIMIT 3 --
SQL built      : SELECT id, status FROM Bill WHERE id = 'x' UNION SELECT name, phone FROM Customer ORDER
BY 1 LIMIT 3 -- '
unsafe page    : 3 row(s)
                 Adi Eko Alatas | 0811-0000-1612
                 Adi Saputra | 0811-0000-1509
                 Agus Fauzi | 0856-0000-8730
safe page      : 0 row(s)
```

> **Ini bukan trik cerdik, dan tidak langka.** *Injection* tercantum dalam OWASP Top Ten sejak 2003 dan masih ada pada edisi 2025, sebagai A05; setiap serangan di sini diketik ke kolom formulir biasa.

`UNION` menambahkan baris dari `SELECT` milik orang asing: nama dan nomor telepon pelanggan, yang merupakan data pribadi menurut UU PDP No. 27/2022. Buku: Subbab 5.15.2.

Perbaikannya: *Prepared Statement*

DEFINISI. Prepared statement memiliki *placeholder* (`?` atau `:nama`) di setiap tempat nilai, dan dikirim ke basis data *lebih dulu*. Nilai-nilainya menyusul secara terpisah, sebagai data, sehingga tak ada nilai yang dapat mengubah apa yang dilakukan pernyataan itu.



Gambar 5.5 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.

Gambar 5.5: bila string direkatkan, teks orang asing menjadi bagian pernyataan; bila pernyataan disiapkan lebih dulu, teks yang sama hanya dapat dibandingkan, sebagai nilai. Buku: Subbab 5.15.3.

Fungsi yang Sudah Diperbaiki

Kode 5.24 Dari `php/injection/safe.php`: karyawan dicari berdasarkan surel saja, dan kata sandi diperiksa terhadap *hash*.

```
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);
}
```

*Dengan Kode 5.24 kata sandi yang benar berhasil dan ketiga serangan ditolak; `UNION` menemukan **0 baris**, karena teks serangan dicari sebagai nilai. Buku: Subbab 5.15.3.*

Aturannya

> **WAWASAN KUNCI: ATURANNYA.** Setiap nilai yang datang dari luar program masuk ke SQL sebagai parameter terikat. Setiap nilai, bukan hanya yang tampak berbahaya.

Nama tabel atau kolom tidak dapat menjadi *placeholder*: bandingkan pilihan pengunjung dengan daftar yang Anda tulis, dan sisipkan hanya string milik Anda sendiri.

Wildcard `LIKE`: mengikat `'%' . $text . '%'` itu aman; `%` dalam teks pengunjung hanya memperluas pencariannya sendiri.

`mysqli` juga punya *prepared statement*: `$stmt->bind_param('s', $email)`, satu huruf tipe per *placeholder*.

Escaping dengan `mysqli_real_escape_string()` harus diingat untuk setiap nilai, selamanya, oleh semua orang; prepared statement benar karena konstruksinya. Buku: Subbab 5.15.4.

Aplikasi Web Kecil: Masuk

Kode 5.25 Bagian PHP dari `php/web/login.php`: `loginSafe()`, *session id* baru saat masuk, dan *redirect*.

```php
<?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.';
}
```

*Pada Kode 5.25, `session_regenerate_id(true)` membuat id yang ditanam sebelum masuk menjadi tak berharga. Halaman terlindung menjawab `POST` yang berhasil dengan redirect (Post/Redirect/Get), memberi formulir yang tidak sah status **400**, dan meng-escape semuanya dengan `h()`. Buku: Subbab 5.16.*

Transaksi yang Sama dalam Java: JDBC

Kode 5.26 Akhir `finish` dalam `java/BillDao.java`: `commit()`, `rollback()`, dan *auto-commit* dipulihkan di `finally`.

```
            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);
        }
```

JDBC: *`Connection`, `PreparedStatement`, `ResultSet`; placeholder adalah `?`, bernomor mulai **1**, diikat dengan `setString`, `setInt`, `setLong`. Metodenya dibuka dengan `conn.setAutoCommit(false)`, yaitu `BEGIN`; Kode 5.26 mengakhiri transaksi dengan salah satu dari dua cara. Buku: Subbab 5.18.1 dan 5.18.2.*

Transaksi yang Sama dalam Python

Kode 5.27 Isi `create_bill` dalam `python/bills.py`: DB-API membuka transaksi sendiri; program hanya mengakhirinya.

```
    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
```

Kode 5.27 adalah `createBill()` baris demi baris; `run()` dalam `db.py` mengubah `?` menjadi `%s` untuk PyMySQL, sebuah placeholder, bukan operator `%` Python. `request.args` pada Flask adalah `$_GET`, `request.form` adalah `$_POST`, dan `{{ }}` Jinja meng-escape secara otomatis. Buku: Subbab 5.19.

Satu Cerita, Enam Kali Dijalankan

Tabel 5.6 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

Tabel 5.6: PHP, Java, dan Python, di SQLite dan di MariaDB, mencetak pelanggan baru yang sama, tagihan Rp 70.000 yang sama, dan menolak serangan yang sama. Aturannya tinggal di skema. Buku: Subbab 5.19.3.

Profitina dalam Praktik

Profitina (`app.profitina.com`, Surabaya) mengukur seberapa siap situs web sebuah merek dikutip mesin jawab AI: Python di atas **PostgreSQL**, dengan SQLite di sebagian jalur pengujiannya.

Setiap laporan adalah `GROUP BY` atas tabel yang terus tumbuh: tren visibilitas mingguan adalah laporan bulanan dengan kolom lain.

***Prepared statement*, tanpa kecuali:** alamat dan unggahan yang diketik siapa saja bisa menyerahkan basis data.

Data pribadi adalah hukum: persetujuan, ekspor, dan penghapusan akun secara lengkap menurut UU PDP No. 27/2022; penghapusan adalah pertanyaan `ON DELETE` yang diajukan regulator.

Latihan yang dapat dijalankan selalu memakai Profitina Laundry; yang terbawa adalah bentuknya: skema ternormalisasi, parameter terikat, transaksi, indeks pada kolom yang dicari. Buku: Subbab 5.20.

Kesalahan yang Sering Terjadi

Konfigurasi: menyunting `httpd.conf` atau `php.ini` tanpa menyalakan ulang, atau menyunting `php.ini` yang salah; “port 80 in use”; mengklik ganda berkas `.php`.

***State*:** keluaran sebelum `header()`; tanpa `exit` sesudah *redirect*; berharap *cookie* baru sudah ada di `$_COOKIE`; memercayakan identitas pada *cookie*.

SQL: `bill` untuk `Bill`; `= NULL`; `UPDATE` tanpa `WHERE`; `LIMIT` tanpa `ORDER BY`; `COUNT(*)` sesudah `LEFT JOIN`; uang dalam `FLOAT`.

Kode: merekatkan nilai ke SQL (juga dengan `%` dan *f-string* Python); *placeholder* bernama dipakai dua kali; beberapa operasi tulis tanpa transaksi; terhubung sebagai `root`.

Setiap kesalahan tercantum di buku beserta perbaikannya. Buku: Subbab 5.21.

Ringkasan: Lima Hal untuk Diingat

1. *Web server* adalah tiga program, Apache, PHP, dan MySQL, masing-masing dengan berkas konfigurasinya; *document root* memetakan folder menjadi situs web.

2. HTTP tidak punya ingatan: *cookie* menyimpan data di klien, sesi menyimpannya di peladen dan hanya mengirim id. Identitas tempatnya di sesi.

3. Tabel terhubung hanya dengan mencocokkan nilai; kunci dan *constraint* menjaga kebenaran data; Persamaan (5.1) dan Persamaan (5.2) adalah fakta yang disimpan dan yang diturunkan.

4. `SELECT ... WHERE ... GROUP BY ... HAVING ... ORDER BY ... LIMIT`, dengan *join*; beberapa operasi tulis yang saling terkait masuk dalam satu transaksi.

5. Setiap nilai dari luar adalah parameter terikat, di PHP, Java, dan Python sama saja.

Satu gagasan, tiga bahasa, angka yang sama. Buku: Subbab 5.22.

Pertemuan Berikutnya

Bab 6 — XML dan Data Terstruktur. Dokumen *well-formed* dan *valid*, DTD dan XML Schema, DOM, SAX, dan XPath, XSLT, serta JSON.

Tagihan yang kita kueri hari ini meninggalkan basis data sebagai umpan XML dan JSON, dibangun dalam PHP, Java, dan Python.

Sebelum itu: kerjakan Soal Latihan 5.3, 5.7, 5.9, 5.12, dan 5.15.

Penyelesaian setiap soal latihan ada di buku, beserta keluaran nyata setiap kueri. Buku: Subbab 5.23 dan 5.24.

Kuliah 6 — XML dan Data Terstruktur

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 6

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini manusia meninggalkan halaman: sebuah program menulis data dan program lain membacanya. Buku: Bab 6, Subbab 6.1–6.24.

Cakupan Pertemuan Ini

1. **Mengapa data terstruktur** — dua program yang harus bersepakat; nama yang bepergian bersama nilainya.
2. **XML itu sendiri** — bahasa-meta, *well-formedness*, entitas, elemen atau atribut, pohon, *namespace*.
3. **Tata bahasa** — validitas, DTD, XML Schema, validasi dalam PHP, Java, dan Python.
4. **Memakai dokumen** — DOM, SAX, dan *pull parsing*; XPath; XSLT; menulis XML dengan pustaka.
5. **JSON dan *feed*** — enam tipe, apa yang diperoleh dan yang hilang, *endpoint feed*, dan jebakan entitas.

Setiap contoh memakai satu hari sungguhan: gerai 001 (Rungkut) pada 26 September 2026, empat tagihan. Buku: Subbab 6.1.

Di Mana Kita Berhenti

Bab 5 memberi Profitina Laundry basis data sungguhan, `laundry.db`, dengan tujuh tabel: `Outlet`, `Customer`, `Employee`, `Service`, `Bill`, `Payment`, `StatusLog`.

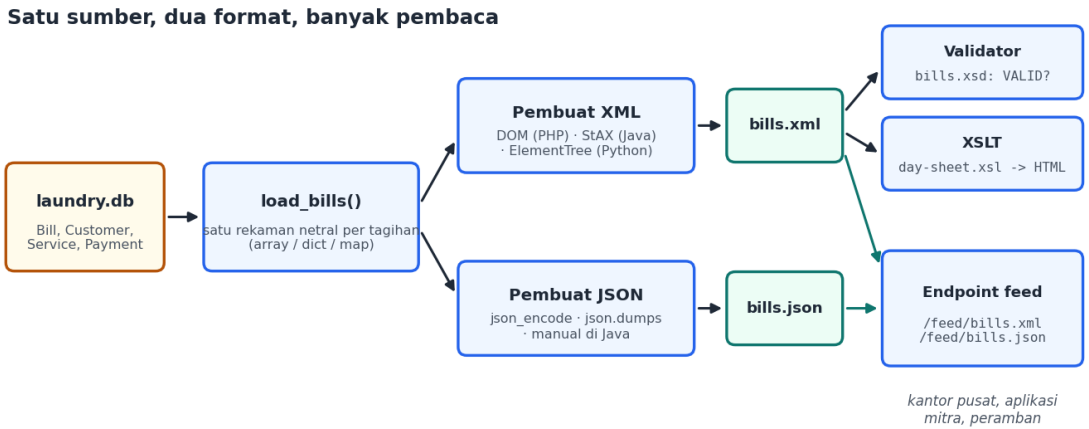
Setiap bab sejauh ini punya **manusia di salah satu ujungnya**: seseorang mengklik, seseorang membaca. `<h1>` berarti *gambarlah ini besar*.

Kini kantor pusat menginginkan tagihan setiap gerai setiap hari, kantor akuntan sebulan sekali, dan *front end* JavaScript meminta daftar harga. **Tidak ada yang melihat datanya.**

Pertanyaannya: dalam bentuk apa data meninggalkan basis data, bagaimana pihak yang menulis dan yang membaca menyepakatinya, dan bagaimana pembaca memeriksa apa yang tiba?

Bab ini menyingkirkan manusia. Buku: Subbab 6.1.

Rencana Bab Ini



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.

***Gambar 6.1**: satu kueri menghasilkan satu rekaman netral per tagihan; dua program penghasil membuat XML dan JSON; dokumennya divalidasi, diubah menjadi lembar harian, dan dilayani oleh endpoint feed. Buku: Subbab 6.1.*

Dua Program yang Harus Bersepakat

> **DEFINISI. Format pertukaran data** adalah cara yang disepakati untuk menuliskan informasi sebagai bita, sehingga program yang tidak menghasilkannya dapat membacanya kembali tanpa bertanya apa pun kepada pembuatnya.

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
```

*Sebelum **Kode 6.1** dapat dipakai, seorang rekan harus diberi tahu: kolom 5 itu kilogram atau potong? Mengapa `42000` muncul dua kali? Mengapa dua kolom kosong? Dan Wash, dry & fold akan menggeser setiap kolom sesudah komanya. Data dan maknanya telah terpisah. Buku: Subbab 6.2.1 dan 6.2.2.*

Nama Bepergian Bersama Nilai

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>
```

> **WAWASAN KUNCI.** Dalam CSV, nama kolom tinggal di luar berkas. Dalam XML, nama setiap nilai ditulis di sampingnya, setiap kali. Harganya pengulangan; manfaatnya berkas yang tetap bermakna sepuluh tahun lagi.

*Pada **Kode 6.2** angka `1` adalah `unit`, tagihannya `cancelled`, yang menjelaskan nilai nol, dan tidak ada `payment` karena tidak ada yang dibayar. Java membuat aplikasi portabel; XML membuat data portabel. Buku: Subbab 6.2.2 dan 6.2.3.*

Bahasa untuk Membuat Bahasa

> **DEFINISI.** XML, **eXtensible Markup Language**, adalah **bahasa-meta**: seperangkat aturan untuk menciptakan bahasa markah. XML tidak memiliki tag sendiri.

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>
```

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>
```

Kode 6.3 *berkata tebal;* ***Kode 6.4*** *berkata ini penting. XML menggambarkan* ***struktur*** *dan* ***makna****, tidak pernah format; satu* ***parser*** *membaca MathML, SVG, dan feed binatu sama saja. Buku: Subbab 6.3.1 dan 6.3.2.*

Lima Hal yang Bukan XML

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 koreksi dalam **Tabel 6.1** mengubah klaim identitas menjadi klaim hubungan. Buka `bills.xml` di peramban dan Anda mendapat pohon tag: tidak ada yang memberi tahu peramban apa arti `<bill>`. Buku: Subbab 6.3.3 dan 6.3.4.*

Dokumen Pertama

```xml
<?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 6.2 Dokumen XML lengkap: catatan singkat dari Naya untuk Yarik. Setiap aturan bahasa ini tampak dalam enam baris tersebut.

*Bacalah **Gambar 6.2** dengan suara keras: ini sebuah catatan; penerimanya Yarik; pengirimnya Naya. Bila kalimatnya berhasil, namanya baik; `value1`, `value2` tidak menggambarkan apa pun. Buku: Subbab 6.4.*

Deklarasi, Elemen, Atribut

Kode 6.5 Deklarasi XML dari **Gambar 6.2**. Diawali `<?` dan diakhiri `?>`, yang menandainya sebagai instruksi pemrosesan, bukan elemen.

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
```

DEFINISI. Elemen adalah tag pembuka, isinya, dan tag penutup yang sepadan. **Atribut** adalah pasangan `nama="nilai"` di dalam tag pembuka. **Elemen akar** memuat semua elemen lain; setiap dokumen memiliki tepat satu.

Gunakan UTF-8. Berkas yang mengaku Latin-1 tetapi memuat karakter di luarnya tidak akan terurai; berkas yang mengaku UTF-8 tetapi disimpan sebagai Latin-1 mungkin terurai lalu memberi Anda huruf yang salah.

Kode 6.5 *adalah instruksi pemrosesan, bukan elemen. Buku: Subbab 6.4.1 dan 6.4.2.*

Feed Tagihan, Bagian 1

Kode 6.6 Awal `data/bills.xml`, diekspor dari `laundry.db`: deklarasi, komentar, akar dengan gerai dan tanggalnya, serta tagihan B2905.

```xml
<?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>
```

*Bacalah **Kode 6.6** dengan suara keras: tagihan B2905, sudah diambil; pelanggan C0022, Vania Wirawan; setrika saja seharga 6.000 rupiah per kilogram, unit 7; ditagih 42.000, dan 42.000 dibayar tunai. Buku: Subbab 6.4.3.*

Feed Tagihan, Bagian 2

Kode 6.7 Akhir `data/bills.xml`: tagihan B2911 yang dibatalkan dan tagihan B2913 yang masih menunggu diambil.

```
  <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>
```

*Pada **Kode 6.7** B2911 tidak memiliki `finished` dan `payment`; B2913 tidak memiliki `departed`. Buku: Subbab 6.4.3.*

Mengapa *Feed* Berbentuk Begini

Pengenal adalah atribut; isi adalah elemen. `id`, `status`, `method` adalah fakta kecil yang atomik; nama, deskripsi, dan jumlah uang adalah isi.

`charged`, bukan `payment`. Kolom basis data `Bill.payment` adalah jumlah yang ditagihkan; setiap elemen `payment` adalah uang yang benar-benar diterima, dan boleh berulang.

Tidak ada berarti tidak ada. Tanpa elemen `finished`, bukan `<finished></finished>` kosong; `cancelled` hanya ditulis bila bernilai benar.

Waktu membawa zonanya: `2026-09-26T09:05:04+07:00`, bentuk `xs:dateTime`. **Hanya yang dibutuhkan pembaca:** tanpa telepon, surel, tanggal lahir, atau *hash* kata sandi.

Feed adalah janji yang diterbitkan, sehingga ia memakai nama yang paling jelas untuk segala hal. Buku: Subbab 6.4.3.

Contoh Terkerjakan 1: Aritmetika Hari Itu

> **CONTOH TERKERJAKAN.** Setiap program dalam bab ini mencetak angka-angka ini, jadi periksalah dengan tangan lebih dulu.

Ditagih: $42000 + 80000 + 0 + 30000 = 152000$ rupiah.

Dibayar: $42000 + 80000 + 30000 = 152000$ rupiah, jadi tidak ada yang terutang.

B2908: unit 10 dengan harga 8.000 per kilogram, $10 \times 8000 = 80000$, tepat `charged`-nya: harga disalin ketika tagihan dibuat.

Ujinya: empat tagihan, satu dibatalkan, Rp 152.000 ditagih, Rp 152.000 dibayar. Program yang mencetak itu tersambung dengan benar.

Perubahan harga di kemudian hari tidak pernah menulis ulang sejarah. Buku: Subbab 6.4.3.

Well-Formed: Enam Aturan

> **DEFINISI.** Sebuah dokumen **well-formed** bila mematuhi sintaksis XML. *Parser* menolak selain itu: tidak memperingatkan, tidak menebak, *menolak*.

1. Satu elemen akar. **2.** Setiap tag pembuka punya tag penutup (atau ditulis `<br/>`). **3.** Huruf besar-kecil yang sama pada keduanya.

4. Elemen bersarang, tidak tumpang-tindih: elemen yang terakhir dibuka ditutup lebih dulu. **5.** Nilai atribut di dalam tanda petik. **6.** Tidak ada atribut yang diulang pada satu elemen.

HTML memaafkan `<p>satu<p>dua`, sebab itu parser HTML sangat besar; XML tidak memaafkan apa pun, sebab itu parser XML kecil. Buku: Subbab 6.5.1–6.5.6.

Dokumen yang Rusak di Empat Tempat

Kode 6.8 `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>
```

*Temukan empat kesalahan dalam **Kode 6.8** sebelum membalik halaman: `date` tanpa petik, `</Customer>` menutup `customer`, `&` telanjang, dan `<unit>` yang tidak pernah ditutup. Buku: Subbab 6.5.7.*

Parser Berhenti pada Kesalahan Pertama

Kode 6.9 Yang dicetak `php/01_wellformed.php` untuk kedua berkas: empat pesan, semuanya tentang baris 2.

```
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
```

Pesan pertama adalah yang sebenarnya: `AttValue: " or ' expected`. Sisanya reruntuhan dari satu kegagalan itu.

Perbaiki satu hal lalu jalankan lagi.

Kode 6.9 *tidak pernah menyebut baris 4 sampai 6. Python berkata* `not well-formed (invalid token)`*; Java berkata* `Open quote is expected`*. Tiga pustaka, tiga kalimat, satu kesalahan. Buku: Subbab 6.5.7.*

Karakter yang Sudah Punya Tugas

DEFINISI. Referensi entitas mewakili sepotong teks: `<`. XML menyediakan lima secara bawaan. **Referensi karakter** menyebut sebuah titik kode: `<` atau `<`.

Tabel 6.2 Lima referensi entitas bawaan XML, beserta referensi karakternya.

Tulis ini	Untuk	Karakter	Juga sah
`<`	lebih kecil	`<`	`<` atau `<`
`>`	lebih besar	`>`	`>` atau `>`
`&`	ampersan	`&`	`&` atau `&`
`"`	tanda petik ganda	`"`	`"` atau `"`
`'`	apostrof	`'`	`'` atau `'`

*Dari **Tabel 6.2**, `&` selalu wajib, sehingga Wash, dry & fold ditulis `Wash, dry & fold`; Ma'ruf tidak perlu di-escape sebagai isi. CDATA dan komentar untuk pengetik dan pembaca manusia, bukan untuk data. Buku: Subbab 6.6.*

Elemen atau Atribut?

```
<population animal="lion">80</population>

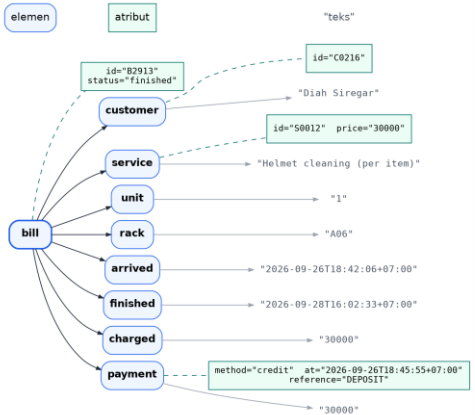
<population>
    <animal>lion</animal>
    <quantity>80</quantity>
</population>
```

Gambar 6.3 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.

***Gambar 6.3**: fakta yang sama dengan dua cara. Elemen dapat berulang, punya bagian, tumbuh kemudian, dan urutannya terjaga. Pakai atribut untuk fakta kecil yang atomik tentang bendanya; bila ragu, pilih elemen. Buku: Subbab 6.7.*

Setiap Dokumen Adalah Pohon

DEFINISI. Jika *B* ditulis langsung di dalam *A*, maka *A* adalah **induk** *B* dan *B* adalah **anak** *A*. Elemen dengan induk yang sama adalah **saudara**; yang berada di bawah *A* pada kedalaman berapa pun adalah **keturunannya**. Setiap elemen kecuali akar memiliki tepat satu induk.



Gambar 6.4 Tagihan B2913 sebagai pohon. Elemen-elemen adalah anak dari `bill`; atribut tergantung pada elemen yang membawanya; setiap elemen daun memiliki satu anak teks.

***Gambar 6.4**: atribut bergantung pada elemennya; setiap elemen daun memiliki satu anak teks. Buku: Subbab 6.8.1.*

Contoh Terkerjakan 2: Menghitung Simpul

Kode 6.10 Sensus yang dicetak `python/04_minidom_nodes.py` untuk `bills.xml`, satu baris per jenis simpul.

```
attribute              31
comment                 1
document                1
element                38
text                   33
text, whitespace only  42
```

Elemen: 1 `bills` + 4 `bill` + 9 + 9 + 7 + 8 = 33 anak, jadi 1 + 4 + 33 = 38.

Teks spasi putih saja: 4 + 1 = 5 di dalam `bills`, dan di dalam setiap tagihan satu lebih banyak daripada jumlah anaknya, 10 + 10 + 8 + 9 = 37: seluruhnya 5 + 37 = 42.

*Pada **Kode 6.10** indentasi yang membuat berkas mudah dibaca adalah isi: `firstChild` dari `<bill>` adalah simpul teks. Mintalah anak berdasarkan nama, atau pasang `$doc->preserveWhiteSpace = false`. Buku: Subbab 6.8.2.*

Namespace

> **DEFINISI. Namespace** (ruang nama) XML adalah nama bagi sebuah kosakata, ditulis sebagai URI. Nama lengkap sebuah elemen adalah pasangan (URI *namespace*, nama lokal).

Kode 6.11 Dari `data/bills-ns.xml`: dua kosakata, dua prefiks, dan dua elemen bernama `customer` yang tidak lagi bertabrakan.

```
<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>
```

*Pada **Kode 6.11** prefiks hanyalah nama panggilan: cocokkan berdasarkan URI. URI tidak pernah diambil. Atribut tanpa prefiks tidak berada di namespace mana pun. Buku: Subbab 6.9.1.*

Jebakan yang Terkenal

Kode 6.12 Keluaran `php/09_namespaces.php`, yang mendaftarkan kedua URI dengan prefiksnya sendiri, `x` dan `k`.

```
//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` tidak menemukan simpul apa pun: nama tanpa prefiks dalam XPath berarti *tanpa namespace*. Bila ekspresi yang jelas benar tidak mengembalikan apa-apa, curigai *namespace* lebih dulu.

*Pada **Kode 6.12**, `$bill['id']` kosong karena sesudah `children(BILL)` SimpleXML mencari `id` di dalam namespace itu; `attributes()` menemukan `B2913`. Buku: Subbab 6.9.2.*

Well-Formed Tidak Sama dengan *Valid*

Kode 6.13 Tagihan pertama dalam `data/bills-invalid.xml`: setiap tag cocok, setiap atribut berpetik, dan tetap salah.

```xml
<?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>
```

*Kesalahan dalam **Kode 6.13**: tanggal `26/09/2026`, id `2905`, status `washing`, unit `0`, waktu tanpa `T` dan zona, metode `ovo`; dan tagihan kedua menaruh `service` sebelum `customer`. Parser menerima ketujuhnya. Buku: Subbab 6.12.1.*

Validitas dan Tata Bahasa

> **DEFINISI.** Sebuah dokumen **valid** bila, selain *well-formed*, ia mematuhi sebuah **tata bahasa** yang dinyatakan: elemen apa yang boleh muncul, dalam urutan apa, berapa kali, dengan atribut apa, dan, dalam XML Schema, nilai seperti apa. Bahasa tata bahasanya: **DTD** dan **XML Schema**.

Tata bahasa adalah pemeriksaan yang ditulis **sekali**, **dokumentasi yang dapat dijalankan** yang tidak bisa usang, dan kegagalan yang dipindahkan **ke perbatasan**, lengkap dengan nomor baris.

Validasi adalah soal komunikasi: dua sistem yang bertukar dokumen harus berbicara bahasa yang sama, yang dituliskan dalam bentuk yang dapat diperiksa mesin. Buku: Subbab 6.12.1.

DTD: Urutan, Pilihan, Berapa Kali

Tabel 6.3 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 memerlukan pelanggan, layanan, unit, rak, dan waktu tiba masing-masing sekali, lalu mungkin waktu selesai, mungkin waktu diambil, jumlah tagihan, dan pembayaran berapa pun: `<!ELEMENT bill (customer, service, unit, rack, arrived, finished?, departed?, charged, payment*)>`.

*Koma berarti urutan, `|` pilihan, dan sufiks pada **Tabel 6.3** menghitung. B2911 valid karena `?` dan `*` membolehkan nol. Buku: Subbab 6.12.2 dan 6.12.3.*

DTD untuk *Feed*

Kode 6.14 Dari `data/bills.dtd`: akar, model isi `bill`, dan ketiga atributnya.

```
<!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">
```

> **Yang tidak dapat dinyatakan DTD:** bahwa `unit` adalah bilangan bulat 1 sampai 999, bahwa `arrived` adalah tanggal dan waktu yang sungguhan, bahwa id adalah `B` dan empat angka. Dari tujuh kesalahan, DTD menangkap empat.

*Pada **Kode 6.14**, `ID` harus unik dan berupa nama XML yang sah (sehingga `2905` gagal); `status` dan `cancelled` adalah enumerasi; `cancelled` berbawaan `"false"`. Buku: Subbab 6.12.4 dan 6.12.5.*

XML Schema: Nilai, Bukan Hanya Bentuk

Tabel 6.4 *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)

*XSD sendiri adalah XML. Sekitar empat puluh tipe bawaan (`xs:integer`, `xs:date`, `xs:dateTime`, …) dipersempit dengan facet pada **Tabel 6.4**; `minOccurs` dan `maxOccurs` menggantikan sufiks DTD. Buku: Subbab 6.13.1 dan 6.13.2.*

Dua Tipe Bernama dari `bills.xsd`

Kode 6.15 Dari `data/bills.xsd`: kuantitas adalah bilangan bulat positif paling besar 999, dan rak adalah huruf A sampai C diikuti dua angka.

```
  <xs:simpleType name="quantity">        <!-- kg, items or pairs -->
    <xs:restriction base="xs:positiveInteger">
      <xs:maxInclusive value="999"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="rack">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-C][0-9]{2}"/>
    </xs:restriction>
  </xs:simpleType>
```

*Tipe sederhana bernama pada **Kode 6.15** didefinisikan sekali dan dipakai lewat namanya. `xs:simpleContent` dengan `xs:extension` berarti teks ditambah atribut; `xs:unique` melarang dua tagihan ber-id sama. Buku: Subbab 6.13.2.*

Contoh Terkerjakan 3: Tujuh Kesalahan Ditemukan

Kode 6.16 Yang dicetak `php/04_validate.php` untuk `bills-invalid.xml` terhadap `bills.xsd`: setiap kesalahan, dengan barisnya.

```
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 ).
```

> **WAWASAN KUNCI.** Skema memeriksa tata bahasa dan tipe data, bukan dunia nyata. Tagihan yang valid bisa saja dibatalkan *sekaligus* ditagih; aturan bisnis diperiksa program, tepat sesudah validasi.

*Dokumennya terurai, sehingga validator pada **Kode 6.16** terus berjalan dan melaporkan semuanya dalam sekali jalan; DTD menemukan empat. "No precomputed value" mengulang keluhan tentang id. Validasi di perbatasan, sekali. Buku: Subbab 6.13.3.*

DOM: Membangun Seluruh Pohon

Kode 6.17 Dari `main` dalam `java/BillsDom.java`: *factory* yang menolak setiap `DOCTYPE`, dan seluruh dokumen diurai menjadi pohon.

```
        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();
```

Kode 6.18 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
```

*Sesudah **Kode 6.17**, perulangan atas `getElementsByTagName("bill")` mencetak **Kode 6.18**; PHP dan Python mencetak teks yang persis sama. `getAttribute` mengembalikan `""` untuk atribut yang tidak ada. Buku: Subbab 6.14.1.*

SAX: Peristiwa, Satu Lintasan, Tanpa Pohon

Kode 6.19 `endElement` dalam `java/BillsSax.java`: program menyimpan jumlah berjalannya sendiri, karena SAX tidak mengingat apa pun.

```
    @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 -> { }
        }
    }
```

*Pada **Kode 6.19** parser memanggil `startElement`, `characters`, dan `endElement`; teks dikumpulkan dalam buffer karena bisa tiba sepotong-sepotong. Jumlahnya sama dengan ringkasan harian. Buku: Subbab 6.14.2.*

Pull Parsing: Kursor yang Anda Kendalikan

Kode 6.20 Perulangan dalam `java/BillsStax.java`: program meminta peristiwa berikutnya, dan dapat `break`.

```
            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;
                }
            }
```

Kode 6.20 *berhenti sesudah* `</bill>` *kedua; sisa berkas tidak pernah diurai. Pada empat tagihan ini tidak menghemat apa-apa; pada empat juta tagihan bedanya antara satu detik dan satu jam. Buku: Subbab 6.14.3.*

Mengukur Perbedaannya

Kode 6.21 `total_streaming` dalam `python/09_stream_memory.py`: `iterparse`, dan `el.clear()` untuk membebaskan setiap tagihan yang selesai.

```python
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 6.22 Keluaran **Kode 6.21**. 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
```

***Kode 6.22**: tagihan setahun, 1,5 MB; pohonnya memakan 13,4 MB, streaming 1,0 MB, dengan jumlah yang sama sampai ke rupiahnya. Hapus `el.clear()` dan `iterparse` diam-diam tetap membangun seluruh pohon. Buku: Subbab 6.14.4.*

Memilih Strategi Membaca

Tabel 6.5 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

***Tabel 6.5**: bila berkas muat dengan lega di memori, pakailah DOM, yang lebih sederhana dan lebih sulit disalahgunakan. Anggarkan sekitar sepuluh kali ukuran berkas untuk sebuah pohon. Buku: Subbab 6.14.4.*

XPath: Bahasa Kueri untuk Pohon

DEFINISI. Ekspresi XPath adalah deretan **langkah** yang dipisahkan `/`, masing-masing memilih simpul relatif terhadap langkah sebelumnya, dan dapat dipersempit dengan **predikat** di dalam `[ ]`. Ekspresi juga dapat menghitung bilangan, string, atau boolean.

Tabel 6.6 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

*XPath bagi XML kira-kira seperti SQL bagi tabel, dan karena ia standar tersendiri, **Tabel 6.6** sama saja di PHP, Java, Python, JavaScript, dan C#. Buku: Subbab 6.15.1.*

Dua Belas Pertanyaan

Kode 6.23 Dua belas ekspresi dalam `php/05_xpath.php`, masing-masing dengan pertanyaan yang dijawabnya.

```
$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',
];
```

Kode 6.23 *memilih tagihan berdasarkan sifat anaknya (`unit > 5`), berdasarkan ketiadaan (`not(payment)`), dan mengajukan pertanyaan pagi di meja kasir: `finished and not(departed)`. Buku: Subbab 6.15.2.*

Dua Belas Jawaban

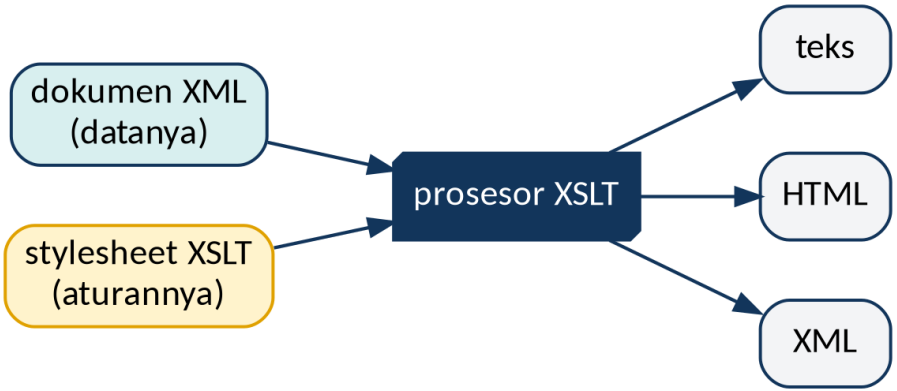
Kode 6.24 Sebagian keluaran `php/05_xpath.php`: sebuah ketiadaan, pertanyaan pagi, dan sebuah perbandingan.

```
//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
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
```

*`count()` mengembalikan `4.0` karena XPath 1.0 hanya punya satu tipe bilangan, double; `div` membagi karena `/` sudah terpakai. Java mencetak jawaban yang sama (**Kode 6.24**); ElementTree Python hanya mengenal sebagian, jadi pakailah lxml. Buku: Subbab 6.15.2 dan 6.15.3.*

XSLT: Satu Dokumen Menjadi Dokumen Lain

DEFINISI. XSLT mengubah dokumen XML menjadi HTML, teks, atau XML lain. ***Stylesheet*** sendiri adalah XML, dan dijalankan oleh **prosesor XSLT**.



Gambar 6.5 Transformasi XSL. Dua dokumen masuk, data dan *stylesheet*; satu prosesor mengerjakannya; keluarannya dapat berupa teks, HTML, atau XML.

Gambar 6.5*: dua masukan, satu prosesor, tiga jenis keluaran. Bila kasir menginginkan kolom telepon, datanya tidak berubah; hanya stylesheet-nya. Buku: Subbab 6.16.1.*

Aturan Templat

Tabel 6.7 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 bersifat fungsional: `xsl:variable` tidak pernah dapat diberi nilai ulang. Jumlah didapat dari `sum()`; sisanya dari templat yang memanggil dirinya sendiri.

*Stylesheet bukan program yang punya awal dan akhir: prosesor menelusuri pohon dan menerapkan aturan yang `match`-nya cocok dengan setiap simpul (**Tabel 6.7**). Buku: Subbab 6.16.2.*

Lembar Harian: Satu Baris per Tagihan

Kode 6.25 Sebagian templat `bill` dalam `data/day-sheet.xsl`: `current()` menggabungkan nomor telepon dari dokumen pelanggan.

```
  <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>
```

*Pada **Kode 6.25**, `document('customers.xml')` memuat pohon kedua sebagai `$people`; di dalam `[ ]` konteksnya adalah rekaman pelanggan, sehingga `current()` diperlukan. Baris diurutkan dengan `data-type="number"`, kalau tidak `"9000"` jatuh sesudah `"80000"`. Buku: Subbab 6.16.3.*

Menjalankannya Tiga Kali

Kode 6.26 Satu baris HTML yang dihasilkan `XSLTProcessor` PHP: B2908, tagihan terbesar, lebih dulu, dengan nomor telepon dari dokumen kedua.

```
<tr>
<td>B2908</td>
<td>Laras Chandra</td>
<td>0877-0000-6820</td>
<td>Wash, dry &amp;amp; fold (per kg) x 10</td>
<td>departed</td>
<td>80.000</td>
<td>80.000</td>
<td>qris</td>
</tr>
```

*Keluaran **Kode 6.26** dari lxml Python identik sampai ke bitanya (keduanya memakai libxslt); prosesor Java menatanya berbeda, tetapi sel, urutan, dan totalnya sama. Peramban sedang menghapus XSLT (Chrome berhenti menjalankannya pada 17 November 2026): lakukan transformasi di peladen, yang juga menjaga data pelanggan tetap privat. Buku: Subbab 6.16.4 dan 6.16.5.*

XML di Peramban

Kode 6.27 Instruksi pemrosesan yang memasang *stylesheet* pada dokumen XML.

```
<?xml-stylesheet type="text/xsl" href="day-sheet.xsl"?>
```

Kode 6.28 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 6.27** peramban yang mengubah; dengan **Kode 6.28** skrip Anda yang mengubah. Huruf "X" pada AJAX dan `XMLHttpRequest` adalah XML: sekitar 2005 data yang diambil berupa XML, lalu JSON mengambil alih tugas itu. Buku: Subbab 6.17.*

Menulis XML: Jangan dengan Menyambung String

> `"<service>" . $description . "</service>"` **rusak pada baris sungguhan pertama,** *Wash, dry & fold*. Ini kesalahan *SQL injection* Bab 5 dalam bahasa lain, dengan jawaban yang sama: pakai pustaka.

Kode 6.29 `addText` dalam `php/laundry_feed.php`: teks masuk lewat `createTextNode`, yang meng-*escape* `&` dan `<`.

```
// 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;
}
```

*Pada **Kode 6.29**, `createElement('service', $text)` **tidak** meng-escape argumen keduanya. Python memakai ElementTree, Java `XMLStreamWriter` dari StAX; tak satu pun program penghasil memuat kurung sudut sebagai markah. Buku: Subbab 6.18.*

Tiga Bahasa, Dua Belas Berkas Identik

Kode 6.30 Yang dicetak `compare_exports.py` sesudah program ekspor PHP, Java, dan Python dijalankan.

```
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
```

***Kode 6.30**: tiga pustaka XML dan satu penyusun JSON buatan tangan sepakat sampai ke bitanya. Secara umum, bandingkan bentuk kanonis (Canonical XML, `c14n`), bukan bita mentahnya. Ekspor PHP juga memvalidasi keluarannya sendiri sebelum diserahkan. Buku: Subbab 6.18.2.*

JSON dari Nol

Tabel 6.8 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`

*JSON, yang dibakukan sebagai ECMA-404 dan RFC 8259, muat dalam satu halaman dan memiliki tepat enam tipe pada **Tabel 6.8**. Buku: Subbab 6.19.1.*

Tagihan yang Dibatalkan dalam JSON

Kode 6.31 Tagihan B2911 dalam `data/bills.json`: medan yang sama dengan XML, dengan bilangan, boolean, dan `null`.

```
            "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": []
```

***Yang diperoleh** pada **Kode 6.31**: bilangan dan boolean di dalam data, serta kecocokan langsung dengan objek setiap bahasa. **Yang hilang:** komentar, pembedaan atribut dan elemen, dan tanggal, yang hanya string menurut kesepakatan. Buku: Subbab 6.19.1.*

Tipe, Ketiadaan, Ukuran

Kode 6.32 Sebagian keluaran `php/07_json.php`: nilai yang sama dibaca dari kedua format, dan ukuran empat tagihan yang sama.

```
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
```

*Pada **Kode 6.32**, dari XML `charged` adalah string, dari JSON bilangan bulat. Tidak ada (XML) dan `null` (JSON) adalah dua dokumen berbeda: feed ini selalu menulis setiap kunci. Setelah dipadatkan dan di-gzip, kedua format selisih beberapa bita saja: ukuran bukan alasannya. Buku: Subbab 6.19.2 dan 6.19.3.*

XML Dibandingkan JSON

Tabel 6.9 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()`

*JSON menguasai web karena ia sintaksis JavaScript sendiri, cocok dengan rekaman, membawa tipe, mudah diimplementasikan, dan datang bersama REST. XML bertahan untuk teks bermarkah, tata bahasa yang disepakati beberapa pihak, dan transformasi standar (**Tabel 6.9**). Buku: Subbab 6.19.4.*

Endpoint Feed

Alamat: `/feed/bills.xml` dan `/feed/bills.json` dengan `?outlet=O01&date=2026-09-26`; tanpa ekstensi, header `Accept` yang memilih, dan jawabannya menyertakan `Vary: Accept`.

Tipe media: `application/xml` dan `application/json`, keduanya `charset=utf-8`. ***Caching*:** `private, max-age=60`, karena *feed* memuat nama pelanggan.

Kode 6.33 Dari `php/feed.php`: kedua parameter kueri diperiksa sebelum basis data disentuh.

```
// 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.');
}
```

*Pada **Kode 6.33** tanggal `2026-02-30` gagal pada pemeriksaan bolak-balik dan mendapat `400 Bad Request` dengan badan galat API binatu. Flask dan `HttpServer` JDK menerapkan rancangan yang sama dan mengembalikan badan yang identik. Buku: Subbab 6.20.*

Bom Entitas

Kode 6.34 `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 6.35 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
```

*libxml pada PHP menolak **Kode 6.34**; Python mengembangkannya; **Kode 6.35** menunjukkan JDK mengembangkannya sampai factory menolak setiap `DOCTYPE`. Ukurlah bawaan perangkat Anda sendiri, jangan menganggapnya. Buku: Subbab 6.21.1 dan 6.21.2.*

XXE: Dokumen yang Membaca Berkas Anda

Kode 6.36 `data/xxe.xml`: `SYSTEM` meminta *parser* mengambil sebuah berkas di peladen.

```
<?xml version="1.0"?>
<!DOCTYPE bills [
  <!ENTITY leak SYSTEM "secret.txt">
]>
<bills><note>&leak;</note></bills>
```

Kode 6.37 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
```

> **PHP:** jangan pernah memberikan `LIBXML_NOENT`. **Java:** `disallow-doctype-decl` pada setiap *factory*. **Python:** biarkan `resolve_entities` lxml mati. **Di mana saja:** batas ukuran dan batas waktu.

*Dengan **Kode 6.36**, bawaan PHP tidak membaca apa pun; **Kode 6.37** menunjukkan factory bawaan JDK membaca `secret.txt`. Feed tidak pernah memerlukan `DOCTYPE`. Buku: Subbab 6.21.3.*

Profitina dalam Praktik

Data terstruktur adalah yang dapat dikutip mesin. Profitina (`app.profitina.com`) melaporkan fakta schema.org yang disematkan sebagai **JSON-LD**: gagasan *namespace* Subbab 6.9 berbaju JSON.

Berkas sederhana untuk mesin jawab: `/llms.txt`, Markdown pendek yang memberi tahu model bahasa apa isi sebuah situs.

Penemuan berjalan di atas XML: `sitemap.xml`, satu `<url>` per halaman; membacanya adalah perulangan DOM dengan nama lain.

Validasi di perbatasan: pemindai membaca dokumen tulisan orang lain, dengan batas ukuran dan waktu serta entitas eksternal dimatikan.

Aplikasi memberi saran; pengguna yang memutuskan. Latihan yang dapat dijalankan selalu Profitina Laundry. Buku: Subbab 6.22.

Kesalahan yang Sering Terjadi

Menulis: menyambung string; `createElement` dengan teks di PHP; elemen kosong untuk nilai yang tidak ada; format waktu basis data di dalam `xs:dateTime`.

Membaca: hanya pesan galat terakhir; mengira `firstChild` adalah elemen; lupa melakukan *cast* di PHP, atau bahwa `.text` bisa `None`.

Mengueri: dokumen ber-*namespace* tanpa mendaftarkan *namespace*-nya; bilangan diurutkan sebagai teks di XSLT; `current()` yang terlupa.

Keamanan dan skala: DOM untuk berkas yang tidak muat, `iterparse` tanpa `clear()`; bawaan entitas *parser* yang tidak diperiksa; tidak ada dikacaukan dengan `null`.

Setiap kesalahan tercantum di buku beserta perbaikannya. Buku: Subbab 6.23.

Ringkasan: Lima Hal untuk Diingat

1. XML memungkinkan program orang asing bertukar data karena nama bepergian bersama nilai; ia bahasa-meta dengan enam aturan *well-formedness*.

2. Dokumen adalah pohon simpul, termasuk spasi putih; *namespace* menjadikan nama sebagai pasangan (URI, nama lokal).

3. *Well-formed* bukan *valid*: DTD menyatakan bentuk, XML Schema bentuk dan nilai. Validasi di perbatasan, lalu periksa aturan bisnis.

4. Baca dengan DOM, SAX, atau *pull parser*; bertanya dengan XPath; ubah dengan XSLT; tulis hanya lewat pustaka.

5. JSON punya enam tipe dan menguasai web; *feed* melayani keduanya, memvalidasi parameternya, dan memperlakukan setiap dokumen dari orang asing sebagai masukan yang tidak tepercaya.

Dua belas berkas hasil ekspor sepakat sampai ke bitanya dalam PHP, Java, dan Python. Buku: Subbab 6.24.

Pertemuan Berikutnya

Bab 7 — Java Server Pages. *Servlet* dan kontainer, daur hidup JSP, *scope*, JavaBeans, EL dan JSTL, *filter*, serta JDBC dalam aplikasi web.

Bab 7 menyunting `web.xml` dengan tangan: XML hari ini mengonfigurasi peladen pertemuan berikutnya.

Sebelum itu: kerjakan Soal Latihan 6.2, 6.5, 6.8, 6.13, dan 6.14.

Penyelesaian setiap soal latihan ada di buku. Buku: Subbab 6.25 dan 6.26.

Kuliah 7 — Java Server Pages

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 7

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini kotak di tengah berbicara Java: sebuah container, servlet, dan halaman yang berubah menjadi servlet. Buku: Bab 7, Subbab 7.1–7.26.

Cakupan Pertemuan Ini

1. ***Container*** — mengapa Java di sisi server ada, *servlet*, `javax` dibandingkan `jakarta`, menjalankan Tomcat.
2. **Apa yang terjadi pada halaman JSP** — penerjemahan, kompilasi, kelas yang dihasilkan, siklus hidup.
3. **Menaruh Java di halaman** — direktif, elemen *scripting*, sembilan objek implisit, empat *scope*.
4. **Mengeluarkan Java lagi** — JavaBeans, Expression Language, JSTL, dan `<c:out>`.
5. **Halaman tagihan binatu** — MVC, JDBC lewat DAO, *login* dan sesi, *filter*, galat; halaman yang sama pada peladen bawaan JDK dan di Flask.

Segala yang ditemukan sesudah JSP pertama ada untuk mengeluarkan kembali Java dari halaman; urutan hari ini mengikuti kisah itu. Buku: Subbab 7.1.

Di Mana Kita Berhenti

Peramban bertanya, peladen menjawab (Bab 1); HTML dan CSS mengisi jawaban (Bab 2); PHP menghitungnya (Bab 3); MySQL mengingat (Bab 5); XML membawa data antarprogram (Bab 6).

Satu hal berubah hari ini: bahasa di kotak tengah. “Penerjemah PHP” menjadi “*servlet container* Java”.

Tiga pertanyaan: mengapa ini ada? Apa yang sebenarnya terjadi pada halaman JSP? Bagaimana menjauhkan Java dari halaman?

Kosakatanya berat (servlet, container, WAR, filter, bean, EL, pustaka tag) dan menyembunyikan betapa sedikit yang baru: permintaan datang, kode berjalan, HTML keluar. Buku: Subbab 7.1.

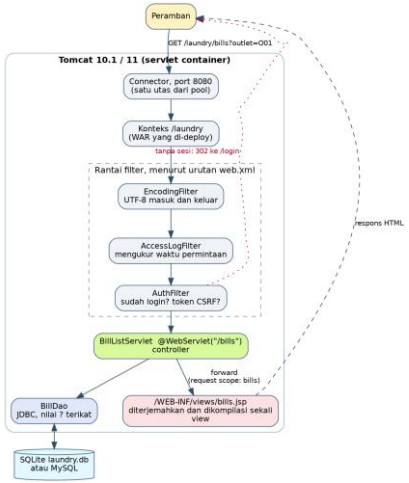
Mengapa Java di Sisi Server, dan Apa Itu *Container*

WAWASAN KUNCI. Java di sisi server menang karena logika bisnis (penggajian, persediaan, akuntansi) sudah ditulis dalam Java, dan jalan terpendek dari peramban ke logika itu adalah program Java.

DEFINISI. *Container* menjalankan kode Anda dan menyediakan segala sesuatu di sekitarnya: soket, *thread*, penguraian HTTP, siklus hidup objek Anda, *pooling*, dan penamaan. Apache **Tomcat** adalah *container* bab ini.

Di PHP Anda menulis berkas dan berkas itu berjalan; di Java Anda menulis kelas lalu menyerahkannya, dan container yang memutuskan kapan ia berjalan. ***Pembalikan kendali*** *inilah sebabnya Java memerlukan konfigurasi. Buku: Subbab 7.2.1 dan 7.2.2.*

Satu Permintaan Melewati *Container*



Gambar 7.1 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.

***Gambar 7.1**: connector di port 8080, aplikasi `/laundry`, tiga filter, servlet yang memutuskan, dan halaman JSP yang menggambar. Buku: Subbab 7.2.2.*

Sebuah *Servlet*

Kode 7.1 Kelas dalam `HelloServlet.java`: `@WebServlet("/hello")`, `extends HttpServlet`, dan `doGet` yang mencetak halaman baris demi baris.

```java
@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>");
    }
```

Servlet *adalah kelas Java yang dipanggil container untuk sebuah alamat; ia menerima objek permintaan dan objek respons. Pada* ***Kode 7.1*** *setiap karakter HTML adalah string Java: tak terlihat oleh editor, penuh tanda petik ber-escape, dan setiap perubahan perlu dikompilasi dan dipasang ulang. Buku: Subbab 7.2.3.*

Membalik Halaman

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 di samping *servlet*. JSP adalah cara *menulis servlet* dengan HTML yang ditulis sebagai HTML.

*Baris ketiga **Tabel 7.1**: halaman JSP tidak menggantikan servlet, ia menjadi servlet. Nasihat bakunya: minimalkan elemen scripting di halaman. Buku: Subbab 7.2.4.*

`javax` dan `jakarta`

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)

> **WAR `javax` di *container* `jakarta` tidak berguna.** Tomcat 10 dan 11 tidak menemukan kelas yang memperluas `jakarta.servlet.http.HttpServlet` dan menjawab setiap alamat dengan 404.

*Aturan dari **Tabel 7.2**: pakai Tomcat 10.1 atau 11; import `jakarta.servlet…`; perbaiki contoh lama dengan mengganti awalan paketnya. `java.sql`, `javax.sql`, dan `javax.naming` milik Java sendiri dan namanya tetap. Buku: Subbab 7.2.5.*

Menjalankan Tomcat

JDK, bukan sekadar *runtime*: `java -version` dan `javac -version` harus sama-sama menjawab; pasang `JAVA_HOME`.

Membongkar arsip berarti memasang: unduh Tomcat 11 (atau 10.1) dari `tomcat.apache.org`, ekstrak ke `CATALINA_HOME`, jalankan `catalina.bat run` atau `bin/catalina.sh run`; di macOS `brew install tomcat`.

Buktikan: `http://localhost:8080/`. *Port* harus ditulis; Apache dan Tomcat adalah dua peladen yang berbeda.

Di VS Code: *Community Server Connectors* membuat, menyalakan, memasang aplikasi ke, dan mematikan Tomcat dari bilah samping.

> **Jangan pernah mengirim `admin`/`admin` di `tomcat-users.xml`.** *Manager* menerima unggahan `.war`, dan `.war` adalah program.

XAMPP 8.2.12 membawa Tomcat 8.5.96, container `javax` yang telah melewati akhir masa pakainya: cukup untuk halaman pertama, tidak untuk servlet bab ini. Buku: Subbab 7.3.1–7.3.6.

Halaman JSP Pertama

Kode 7.2 `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>
```

Kode 7.3 Respons sebenarnya untuk **Kode 7.2**, 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>
```

***Template text** pada **Kode 7.2** disalin tanpa perubahan; `<%= … %>` adalah lubang yang diisi saat permintaan. Pada **Kode 7.3** tag itu lenyap, spasinya bertahan, dan tanggalnya milik peladen. MiniJSP, mesin kecil yang hanya memerlukan JDK, yang melayaninya. Buku: Subbab 7.4.*

Contoh Terkerjakan 1: Tebak Keluarannya

CONTOH TERKERJAKAN. Tebaklah keluaran `Your bill is <%= 4 * 12000 %> rupiah.`

Langkah 1. `Your bill is` adalah *template text*: disalin.

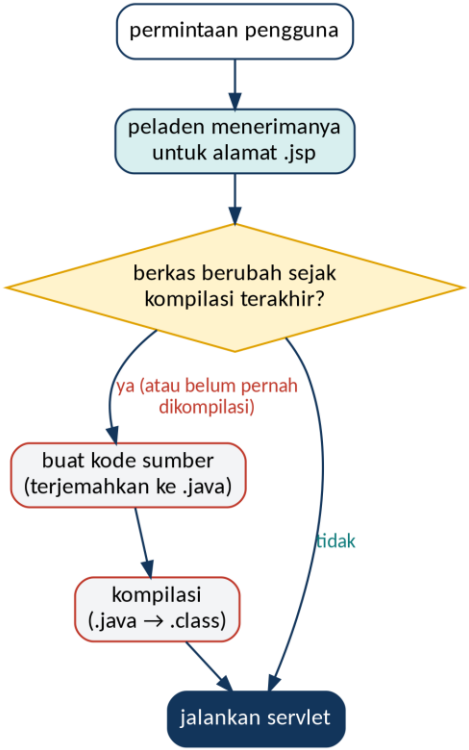
Langkah 2. `4 * 12000` adalah aritmetika bilangan bulat Java: $4 \times 12000 = 48000$, diubah menjadi teks `48000`.

Langkah 3. `rupiah.` disalin. Peramban menerima `Your bill is 48000 rupiah.`

Yang tidak ada: tanpa pemisah ribuan, tanpa `Rp`. `getPaymentRupiah()` milik sebuah *bean* mengembalikan `Rp 48.000`.

Bagaimana dengan `<%= 4 * 12000; %>`? Baris yang dihasilkan adalah `out.print(4 * 12000);`, yang tidak dapat dikompilasi. Buku: Subbab 7.4.3 dan 7.5.3.

Halaman JSP Adalah *Servlet* yang Menyamar



Gambar 7.2 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.

***Gambar 7.2**: halaman **diterjemahkan** menjadi Java, **dikompilasi**, dimuat, dan dibuat instansnya sekali, hanya bila berkasnya berubah. Permintaan pertama lambat, menyunting halaman yang sedang berjalan langsung berlaku, dan container memerlukan kompilator saat berjalan. Buku: Subbab 7.5.1 dan 7.5.2.*

Servlet yang Dihasilkan

Kode 7.4 Isi `_jspService` pada kelas yang dihasilkan MiniJSP dari `hello.jsp`: objek implisit dideklarasikan, `out.write` untuk *template text*, `out.print` untuk ekspresi.

```
        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");
```

*Pada **Kode 7.4** kelasnya `hello_jsp extends JspPage`; `_jspService` berjalan sekali per permintaan; variabel di bagian atas adalah objek implisit. Galat kompilasi menyebut berkas hasil terjemahan: bacalah baris "in the jsp file". Buku: Subbab 7.5.3 dan 7.5.4.*

Riwayat Satu Permintaan

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 7.2**)
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 7 dan 8 pada **Tabel 7.3** terjadi sekali: **satu instans melayani setiap pengunjung**. Buku: Subbab 7.5.5.*

Halaman yang Menghitung

> **DEFINISI. Siklus hidup JSP**: `jspInit()` sekali, `_jspService()` sekali per permintaan, `jspDestroy()` sekali. Anda boleh menulis yang pertama dan terakhir, tidak pernah yang tengah.

Kode 7.5 Deklarasi dalam `counter.jsp`: sebuah *field*, metode `synchronized` yang menaikkannya, dan dua metode siklus hidup.

```
<%!
    // 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()"); }
```

*Semua yang ada pada **Kode 7.5** menjadi anggota kelas yang dihasilkan, di luar setiap metode. Lebih ke bawah, halaman menulis `<% int local = 0; local++; %>`, variabel lokal `_jspService`. Buku: Subbab 7.5.6.*

Field atau Variabel Lokal?

Kode 7.6 Keluaran sebenarnya dari dua permintaan berturut-turut ke **Kode 7.5**. *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>
```

> ***Bug* paling keji di bab ini.** *Field* yang dideklarasikan dalam `<%! … %>` dipakai bersama oleh setiap pengunjung sekaligus. `<%! private String customerName; %>` yang diisi dari permintaan membuat dua kasir melihat pelanggan satu sama lain.

*Pada **Kode 7.6** field menghitung 1, 2; variabel lokal selalu 1; `jspInit()` tercetak sekali di konsol. Jangan pernah menaruh data permintaan dalam deklarasi. Buku: Subbab 7.5.6.*

Servlet Selengkapnya

Siklus hidup: satu instans, `init()` sekali, `service()` per permintaan di banyak *thread*, `destroy()` di akhir; `doGet`, `doPost`; metode yang tidak diimplementasikan menjawab 405.

Permintaan: `getParameter`, `getParameterValues`, `getHeader`, `getSession`, `setAttribute`, `getRequestDispatcher`. **Respons:** `setContentType`, `getWriter`, `setStatus`, `sendError`, `sendRedirect`, `addCookie`.

Pemetaan: `@WebServlet("/bills")` atau `<servlet-mapping>` di `web.xml`; pola tepat (`/bills`), awalan (`/bill/*`), ekstensi (`*.jsp`). *Descriptor* yang menang.

Field sebuah servlet hanya boleh menyimpan yang sama untuk setiap permintaan; servlet binatu tidak punya field sama sekali. Buku: Subbab 7.6.1–7.6.4.

Aplikasi Web: `WEB-INF` dan WAR

DEFINISI. `WEB-INF` tidak pernah dilayankan ke peramban: kelas, jar, `web.xml`, dan *view* tinggal di sana. **WAR** adalah ZIP aplikasi dengan tata letak itu, satuan pemasangan. ***Deployment descriptor*** `WEB-INF/web.xml` memuat pengaturan, pemetaan, halaman galat, dan aturan sesi, tanpa logika.

Enam langkah: kompilasi; sumber daya statis; `web.xml`; bangun WAR dengan JDK saja; taruh di `webapps/` (atau lewat *manager*, atau VS Code); minta halamannya.

Jangan pernah menaruh Servlet API di dalam WAR. *Container* sudah menyediakannya; salinan kedua menghasilkan kelas yang tidak dapat di-*cast* ke dirinya sendiri. JSTL, sebaliknya, harus dibawa di `WEB-INF/lib`.

View binatu tinggal di `WEB-INF/views/`, hanya dapat dicapai lewat forward dari servlet, sehingga tidak ada pintu samping yang melewati pemeriksaan login. Buku: Subbab 7.7.1–7.7.4.

Filter di `web.xml`, Menurut Urutan

Kode 7.7 Pemetaan *filter* dalam `WEB-INF/web.xml`: pengodean dan log akses untuk setiap alamat, pemeriksaan *login* hanya untuk halaman tagihan.

```
  <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>
```

*Hanya `web.xml` yang menetapkan **urutan** filter, sebab itu **Kode 7.7** mendeklarasikannya di sana. Berkas yang sama memasang `shopName`, URL basis data, halaman galat, dan sesi `HttpOnly` 30 menit. Buku: Subbab 7.7.2.*

Contoh Terkerjakan 2: Alamat yang Mana?

> **CONTOH TERKERJAKAN.** WAR-nya `laundry.war`, `BillListServlet` membawa `@WebServlet("/bills")`, Tomcat berjalan di port 8080. Alamat apa yang mencapai *servlet* itu, dan apa yang salah dengan `<a href="/bills">`?

Peladen `http://localhost:8080` + ***context path*** `/laundry` (dari nama berkas) + **pola** `/bills` = `http://localhost:8080/laundry/bills`.

`<a href="/bills">` menunjuk ke `http://localhost:8080/bills`, tanpa *context path*: **404**.

Setiap tautan binatu ditulis `${ctx}/bills`, dengan `ctx` ditetapkan sekali sebagai `${pageContext.request.contextPath}`; ganti nama WAR dan tautannya tetap bekerja.

Servlet membaca nilai kueri sebagai `request.getParameter("username")`, halaman sebagai `${param.username}`. Buku: Subbab 7.7.5.

Direktif

> **DEFINISI. Direktif**, `<%@ … %>`, adalah pesan kepada penerjemah tentang *servlet* yang akan dihasilkannya. Direktif tidak menghasilkan keluaran dan dipatuhi sekali, pada fase penerjemahan.

`page`: `import`, `contentType="text/html; charset=UTF-8"`, `errorPage`, `isErrorPage`, `session`. `isThreadSafe="true"` adalah janji *Anda*, bukan layanan.

`include`: `<%@ include file="/WEB-INF/jspf/header.jspf" %>` menempelkan *teks* sebelum penerjemahan: sesudahnya hanya ada satu halaman, dengan variabel yang sama.

`taglib`: `<%@ taglib prefix="c" uri="jakarta.tags.core" %>`; sesudah itu `<c:forEach>` adalah tag `forEach` dari pustaka yang dinamai `c`.

Setiap view binatu menyertakan `header.jspf`, yang mendeklarasikan pustaka tag dan menetapkan `ctx`. Buku: Subbab 7.8.1–7.8.5.

Elemen *Scripting*

Tabel 7.4 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	—

> `<%= request.getParameter("x") %>` **adalah lubang *cross-site scripting* Bab 3, dalam Java.** Obatnya tetap: *escape* saat keluaran, di JSP dengan `<c:out>`.

*Kolom ketiga **Tabel 7.4** menjelaskan semuanya: ekspresi menjadi `out.print(expr);`, scriptlet ditempelkan ke dalam metode, deklarasi mendarat di kelas dan tidak dapat melihat `request`. Buku: Subbab 7.9.1–7.9.5.*

Scriptlet yang Membentang di atas HTML

Kode 7.8 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
    }
%>
```

*Pada **Kode 7.8** `while` dibuka di satu scriptlet dan ditutup di scriptlet lain. Dengan dua barang di keranjang, `hasNext()` benar dua kali dan dua baris ditulis; harganya* $39.95 + 39.99 = 79.94$*. Buku: Subbab 7.9.3.*

Menjadi Apa Perulangan Itu

Kode 7.9 Garis besar hasil terjemahan **Kode 7.8**. 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
}
```

> **Kurung kurawal yang tidak ditutup.** Hapus *scriptlet* terakhir dan kompilator melaporkan "insert `}` to complete Block" di akhir berkas hasil terjemahan. `<c:forEach>` tidak dapat dibiarkan terbuka tanpa ketahuan.

***Kode 7.9**: HTML di antara scriptlet menjadi panggilan `out.write` di dalam badan `while`. Tidak pernah ada sesuatu yang cerdik di sana. Buku: Subbab 7.9.3.*

Objek Implisit

Tabel 7.5 Tujuh dari sembilan objek implisit; dua lainnya adalah `config`, pengaturan milik halaman ini, dan `page`, yaitu `this`.

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`
`pageContext`	`PageContext`	permintaan ini	menjangkau keempat *scope* sekaligus
`exception`	`Throwable`	permintaan ini	hanya di halaman dengan `isErrorPage="true"`

***Tabel 7.5**: variabel lokal yang dideklarasikan penerjemah. `getParameter` mengembalikan `null` untuk kolom yang tidak dikirim; binatu membaca parameter lewat `Validator.field`. Buku: Subbab 7.10.*

Empat *Scope*

DEFINISI. ***Scope*** (cakupan) adalah peta bernama dari string ke objek, dengan masa hidup: **page**, **request** (termasuk *forward*), **session** (peramban ini), dan **application** (semua orang, sampai aplikasi berhenti).

Tabel 7.6 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`

Tabel 7.6 *menjawab "berapa lama nilai ini harus hidup?". `${name}` mencari di page, request, session, application, dan yang tersempit menang; sebutkan scope-nya bila penting: `${sessionScope.user}`. Buku: Subbab 7.11.1 dan 7.11.3.*

Menyaksikan *Scope* Bergerak

Kode 7.10 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
```

> ***Scope* application dipakai bersama oleh semua orang, pada saat yang sama.** Dua permintaan bisa sama-sama membaca 7 lalu menulis 8. Simpan di sana hal yang sebagian besar hanya dibaca; taruh penghitung di basis data.

*Pada **Kode 7.10** page tidak pernah bergerak, request selalu baru, session dan application naik bersama; dari peramban kedua sesi mulai lagi dari 1 sementara application berlanjut. Buku: Subbab 7.11.2.*

JavaBeans: Halaman Tanpa Java

DEFINISI. JavaBean memiliki konstruktor publik tanpa argumen, *field* privat, dan satu properti untuk setiap `getX()` (atau `isX()`) dan `setX(value)` yang publik. `getPaymentRupiah()` menjadikan `paymentRupiah` sebuah properti tanpa *field* bernama itu.

Kode 7.11 Perulangan dalam `bills.jsp` milik MiniJSP: EL membaca properti *bean*, JSTL mengulang, `<c:out>` meng-*escape*.

```
      <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>
```

*Di atas **Kode 7.11**, `<jsp:useBean id="book" class="laundry.BillBook"/>` menemukan atau membuat bean dan `<jsp:setProperty … param="outlet"/>` mengisi satu properti. Jangan pernah `property="*"` pada bean yang memuat harga: pengunjunglah yang akan memilihnya. Buku: Subbab 7.12.1–7.12.3.*

Contoh Terkerjakan 3: Memeriksa Antrean

Kode 7.12 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 TERKERJAKAN. B2949: cuci, kering, dan lipat, Rp 8.000 per kg, 7 kg: $7 \times 8000 = 56000$, tercetak `Rp 56.000`. B2948: setrika, Rp 6.000 per kg, 6 kg: $6 \times 6000 = 36000$, tercetak `Rp 36.000`.

*Bean tidak menghitung apa pun pada **Kode 7.12**: jumlahnya ditetapkan ketika setiap tagihan dibuat, sehingga perubahan harga kemudian tidak menulis ulang sejarah. Buku: Subbab 7.12.3.*

Expression Language

Kode 7.13 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}
```

`${ … }` membaca properti, masuk ke peta dan *list*, membandingkan, dan menguji kekosongan, dan dengan sengaja tidak dapat mengueri basis data atau memberi nilai.

Titik masuk ke dalam: `${bill.customerName}` memanggil `getCustomerName()`. **Kurung siku** mengizinkan kunci lain: `${prices['wash-fold']}`. **Nama yang tidak ada mencetak kosong.**

Nama milik EL sendiri: `param`, `paramValues`, `header`, `cookie`, `initParam`, keempat peta `…Scope`, dan `pageContext`.

*Baris pertama **Kode 7.13** memerlukan cast, nama kelas, dan panggilan metode; baris kedua tidak memerlukan satu pun dan tetap bekerja bila paketnya pindah. Buku: Subbab 7.13.1, 7.13.2, dan 7.13.4.*

Setiap Operator, dengan Jawabannya

Kode 7.14 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
```

*Pada **Kode 7.14** bilangan bulat tetap bulat di bawah `+`, `-`, `*`, `mod`; `/` dan `div` menghasilkan pecahan; `${prices.wash-fold}` bernilai `0` karena tanda hubung adalah minus. `empty` benar untuk `null`, `""`, dan koleksi kosong. Buku: Subbab 7.13.3.*

JSTL: Sisa Java Terakhir Keluar

Tabel 7.7 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*

JSTL (Tabel 7.7) menghapus Java yang memutuskan dan mengulang. Buku: Subbab 7.14.1, 7.14.2, dan 7.14.4.

`<c:out>` dan Serangan yang Dihentikannya

Kode 7.15 Nilai yang sama dicetak dengan dua cara. Yang pertama adalah lubang; yang kedua bukan.

```
Customer: <%= request.getParameter("name") %>
Customer: <c:out value="${param.name}"/>
```

> **`${x}` telanjang di *template text* tidak meng-*escape*.** Aman untuk id tagihan atau jumlah yang sudah diformat; berlubang untuk apa pun yang diketik orang: nama, telepon, kotak *filter*.

*Dengan nama `<script>alert('x')</script>`, baris pertama **Kode 7.15** menjalankan skrip di setiap peramban; baris kedua menggambarnya sebagai teks. Validasi saat masukan, escape saat keluaran. Buku: Subbab 7.14.3.*

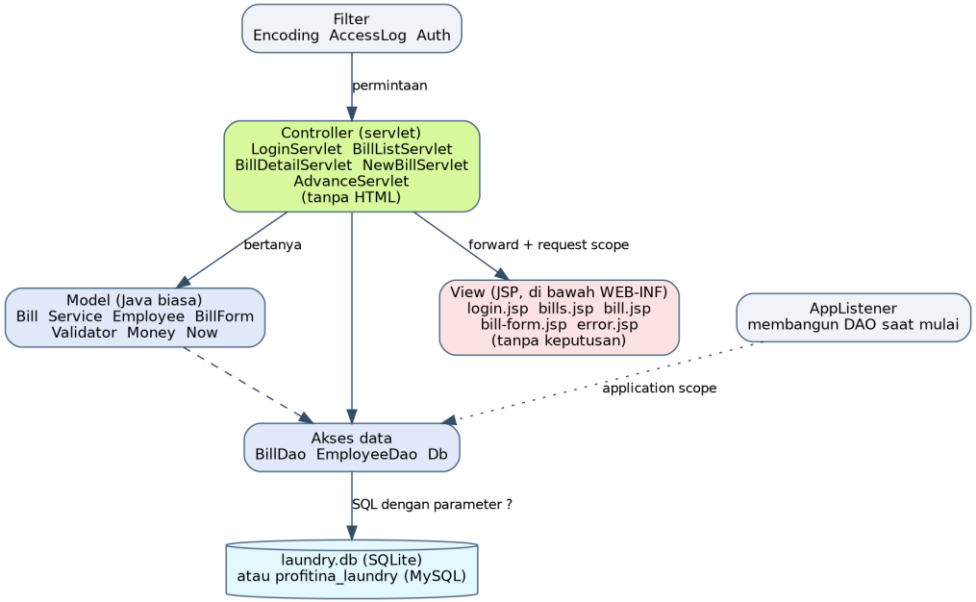
Model–View–Controller

DEFINISI. Model adalah data dan aturannya, tanpa tahu apa pun tentang web; **view** menampilkan; **controller** menerima permintaan, meminta model bekerja, dan memilih *view*. Di sini: sebuah *servlet*, sebuah halaman JSP, dan kelas-kelas Java biasa.

WAWASAN KUNCI. *Controller* tidak berisi HTML dan *view* tidak berisi keputusan. Hampir setiap kesulitan kemudian berasal dari pelanggaran salah satu dari dua batas itu.

Kekuatan servlet adalah mengendalikan dan meneruskan; kekuatan halaman adalah menampilkan. Buku: Subbab 7.15.1.

Halaman Binatu sebagai MVC



Gambar 7.3 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.

***Gambar 7.3**: lima servlet, lima view di bawah `WEB-INF/views`, model berupa kelas biasa dengan `BillDao` dan `EmployeeDao`, tiga filter di depan, dan listener yang membangun DAO saat aplikasi mulai. Buku: Subbab 7.15.2.*

Controller untuk `GET /bills`

Kode 7.16 `doGet` pada `BillListServlet`: membaca dua *filter*, meminta model, menaruh hasilnya di *scope* request, lalu *forward*. Tanpa HTML.

```
    @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);
    }
```

*Pada **Kode 7.16** parameter yang tidak ada menjadi `""`, status yang tidak dikenal berarti "semua", DAO diambil dari scope application, dan view hanya dapat dicapai lewat forward ini. Buku: Subbab 7.15.3.*

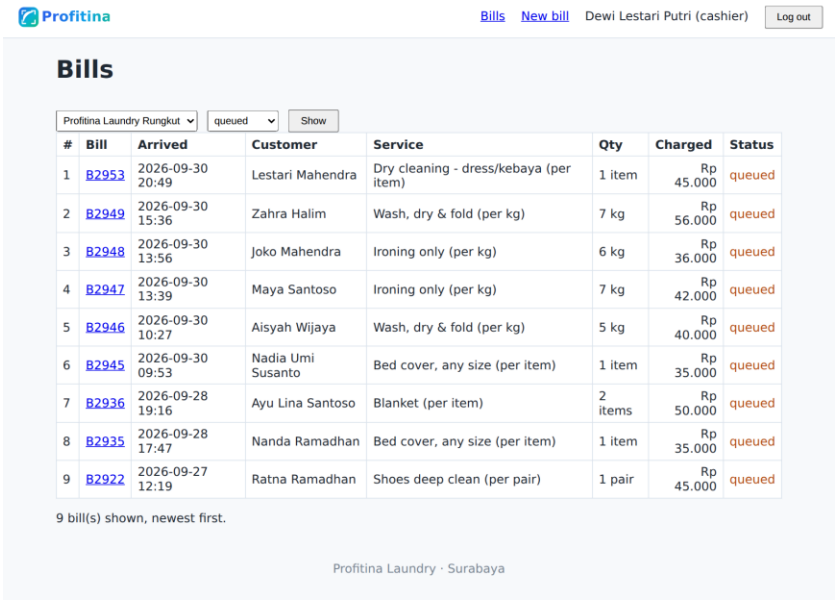
View: Hanya EL dan JSTL

Kode 7.17 Perulangan dalam `bills.jsp`: nama dan deskripsi melewati `<c:out>`; id, jumlah uang, dan kata status milik aplikasi sendiri.

```
      <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>
```

Kode 7.17 *dimulai dengan menyertakan* `header.jspf`*, yang menetapkan* `ctx`*, mendeklarasikan pustaka tag, menampilkan siapa yang login, dan mencetak pesan flash sekali pakai. Buku: Subbab 7.15.3.*

Daftar Tagihan di Mata Kasir



Gambar 7.4 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`.

Gambar 7.4 *berasal dari jalur HttpServer JDK, yang menghasilkan markah yang sama dengan* `bills.jsp` *dan benar-benar dijalankan. Buku: Subbab 7.15.3.*

Formulir: Validasi, Tampilkan Ulang, atau Simpan lalu *Redirect*

Kode 7.18 Dari `doPost` pada `NewBillServlet`: formulir yang buruk kembali dengan status 400; formulir yang baik disimpan dengan harga yang dihitung di sini, lalu peramban di-*redirect*.

```
        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);
```

*Sebelum **Kode 7.18**, peran diperiksa di peladen dan formulir disalin kolom demi kolom. **Post/Redirect/Get:** Reload mengulang GET `/bill?id=B2954` yang tidak berbahaya, tidak pernah POST-nya. Pesan flash bertahan melewati redirect di scope session. Buku: Subbab 7.15.4.*

Contoh Terkerjakan 4: Tagihan Baru

Kode 7.19 `Validator.units`: kilogram dibulatkan ke atas menjadi bilangan bulat dengan `BigDecimal`; potong harus sudah bulat.

```
    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;
        }
    }
```

Pelanggan `c0001`, S0002 (kilat, Rp 12.000 per kg), berat `3.2`, rak `a02`: id dijadikan huruf besar `C0001`, `A02`; 3,2 kg dibulatkan **ke atas** menjadi 4 unit; $4 \times 12000 = 48000$.

*Id tertinggi B2953, jadi tagihan baru B2954, disimpan `queued` bersama baris `StatusLog`-nya: "Bill B2954 created: 4 kg x Rp 12.000 = Rp 48.000." (**Kode 7.19**). Ketiga jalur mencetak persis ini. Buku: Subbab 7.15.4.*

JDBC Lewat DAO

DEFINISI. Data Access Object memuat setiap pernyataan yang menyentuh penyimpanan dan menyediakan metode dalam bahasa aplikasi: `find`, `findById`, `insert`, `advance`.

Kode 7.20 Awal `BillDao.find`: satu teks SQL yang tetap, lima *placeholder* diikat satu per satu.

```
    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);
```

*Pada **Kode 7.20** `(? = '' OR b.Outlet_id = ?)` berarti "gerai mana saja bila filter kosong"; "gerai" berisi `' OR '1'='1` tidak cocok dengan apa pun. `insert` membungkus tagihan dan baris log-nya dengan `setAutoCommit(false)`, `commit()`, `rollback()`. URL JDBC, atau pool JNDI, memilih SQLite atau MySQL. Buku: Subbab 7.16.*

Login

Kode 7.21 Akhir `LoginServlet.doPost`: satu jawaban untuk kedua kesalahan; bila berhasil, *session id* baru, karyawan dan token CSRF di sesi, lalu *redirect*.

```
        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;
        }
        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");
```

Kode 7.21*: satu pesan sehingga tak ada yang tahu surel mana yang terdaftar; kata sandi diperiksa terhadap hash PBKDF2 dalam waktu konstan;* `changeSessionId()` *menggagalkan session fixation; token CSRF memungkinkan filter menolak POST palsu. Jangan pernah mencetak session id. Buku: Subbab 7.17.*

Filter

DEFINISI. ***Filter*** berjalan sebelum *servlet* untuk setiap alamat yang cocok; `chain.doFilter(request, response)` meneruskan permintaan, dan *filter* yang tidak memanggilnya menghentikan permintaan di situ.

Kode 7.22 Isi `AuthFilter.doFilter`: tidak ada karyawan di sesi, ke halaman *login*; POST tanpa token yang benar, 403; selain itu teruskan.

```
        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 **Kode 7.22** tidak ada servlet tagihan yang memuat pemeriksaan login, dan tak satu pun dapat melupakannya. Filter memuat aturan yang melintang: middleware di ASP.NET Core, `before_request` di Flask. Buku: Subbab 7.18.*

Galat, *Include*, *Forward*

Tabel 7.8 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

> **WAWASAN KUNCI: *forward* bukan *redirect*.** *Forward* tetap di dalam peladen: satu permintaan, bilah alamat tetap, *scope* request utuh. *Redirect* mengirim 302 atau 303 dan peramban bertanya lagi. *Forward* untuk menampilkan *view*; *redirect* sesudah POST yang berhasil.

*Dari **Tabel 7.8**: direktif untuk isi yang jarang berubah, `<jsp:include>` untuk isi yang dipilih per permintaan. `error.jsp` menampilkan satu kalimat dan mencatat eksepsinya: jangan pernah menampilkan stack trace. Buku: Subbab 7.19.*

Satu Aplikasi, Tiga Tumpukan

Tabel 7.9 Sebagian tabel terjemahan antara ketiga tumpukan. 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`
Sesi	`HttpSession` (di server)	`Sessions` (ditulis sendiri)	`flask.session` (*cookie* bertanda tangan)
Cetak dengan *escape*	`<c:out value="${x}"/>`	`{{x}}`	`{{ x }}`
Perulangan	`<c:forEach>`	perulangan Java atas templat baris	`{% for b in bills %}`
Forward ke *view*	`RequestDispatcher.forward`	`Template.render`	`return render_template(…)`

***Tabel 7.9**: objek pengganti container (16 pemeriksaan), peladen JDK lewat HTTP sungguhan (14), dan test client Flask (11) semuanya mencetak "Bill B2954 created: 4 kg x Rp 12.000 = Rp 48.000." Jinja2 meng-escape secara bawaan. Buku: Subbab 7.20–7.22.*

JSP di Samping PHP

Tabel 7.10 PHP dan *servlet*/JSP berdampingan. Dua baris terakhir adalah satu-satunya perbedaan yang mendasar.

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")`, `${param.q}`
Escape untuk HTML	`htmlspecialchars($x)`	`<c:out value="${x}"/>`
Sesi	`$_SESSION['k']` sesudah `session_start()`	`session.setAttribute("k", v)`
Prepared statement	PDO dengan penampung `?`	`PreparedStatement` dengan `?`
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 **Tabel 7.10** adalah yang harus dilupakan pengembang PHP: field Java dipakai bersama oleh setiap pengunjung, dan URL memetakan ke komponen, bukan ke berkas. Buku: Subbab 7.23.*

Profitina dalam Praktik

Profitina (`app.profitina.com`) adalah Python di atas FastAPI dan PostgreSQL, tanpa Java, dan setiap gagasan hari ini menopangnya.

***Controller* tanpa HTML, *view* tanpa keputusan:** *dashboard* menampilkan skor dan tidak menghitung apa pun.

***Filter*:** autentikasi, pembatas laju *login*, dan pemeriksaan persetujuan berjalan sebelum *handler* mana pun. ***Escaping* tanpa kecuali** di umpan komunitas.

***Scope* dan rahasia:** pemindaian per permintaan, pengguna per sesi; *hash* Argon2id; kewajiban UU PDP sebagai fitur.

Studi kasus yang dapat dijalankan selalu Profitina Laundry; bentuknya terbawa lintas bahasa. Buku: Subbab 7.24.

Kesalahan yang Sering Terjadi

Pemasangan: *import* `javax` di Tomcat 10 atau 11; jar Servlet API di dalam WAR; tanpa jar JSTL; `admin`/`admin`.

Halaman: titik koma di dalam `<%= … %>`; data permintaan dalam deklarasi atau *field servlet*; `${x}` telanjang untuk nilai ketikan; `${prices.wash-fold}`; `<a href="/bills">`.

Permintaan: lupa bahwa `getParameter` mengembalikan `null`; memasang pengodean sesudah `getParameter` pertama; `property="*"` pada *bean* yang memuat harga.

Rancangan: *forward* sesudah POST yang berhasil; SQL yang dibangun dengan menyambung string; *stack trace* diperlihatkan kepada pengunjung; `HttpExchange.setAttribute` dikira per permintaan.

Setiap kesalahan tercantum di buku beserta perbaikannya. Buku: Subbab 7.25.

Ringkasan: Lima Hal untuk Diingat

1. *Container* menjalankan kelas Anda: *connector*, *filter*, *servlet*, halaman JSP. Pakai Tomcat 10.1 atau 11 dan `jakarta.*`.
2. Halaman JSP diterjemahkan menjadi *servlet* sekali; setiap permintaan adalah satu panggilan `_jspService` pada satu objek bersama.
3. Sembilan objek implisit dan empat *scope*; deklarasi dipakai bersama oleh semua orang.
4. *Bean*, EL, dan JSTL mengeluarkan Java dari halaman; `<c:out>` meng-*escape*, dan bawaan itu adalah kendali keamanan.
5. *Servlet* dan JSP bersama adalah MVC; DAO mengikat setiap nilai; *filter* menjaga; *redirect* sesudah POST yang berhasil.

Aplikasi yang sama berjalan dengan API Tomcat, dengan peladen bawaan JDK, dan di Flask: hanya namanya yang berubah. Buku: Subbab 7.26.

Pertemuan Berikutnya

Bab 8 — Evaluasi dan Pembahasan II. Bab 5 sampai 7 dikerjakan tuntas: situs web CRUD dengan autentikasi, dan soal-soal baru tentang PHP dan MySQL, XML dan JSON, serta *servlet* dan JSP.

Setiap soal diselesaikan dalam PHP, Java, dan Python, dan diperiksa oleh sebuah program.

Sebelum itu: kerjakan Soal Latihan 7.2, 7.5, 7.6, 7.8, dan 7.9, dan baca ulang Bab 5 dan 6.

Penyelesaian setiap soal latihan ada di buku. Buku: Subbab 7.27 dan 7.28.

Kuliah 8 — Evaluasi dan Pembahasan II: Aplikasi Web Berbasis Basis Data, dari Rancangan sampai Penerapan

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 8

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini tidak ada teknologi baru: dua belas soal, masing-masing diselesaikan dari halaman kosong dalam PHP, SQL, XML, JSON, atau JSP, lalu sekali lagi dalam Java dan Python, dengan setiap keluaran dicetak oleh programnya. Buku: Bab 8, Subbab 8.1–8.11.

Cakupan Pertemuan Ini

1. **Sebuah metode** — delapan langkah dari satu paragraf kebutuhan sampai aplikasi web yang teruji.
2. **Proyek, Soal 8.1–8.4** — rancangan basis data, registrasi dan *login*, delapan fitur, serta bukti dan penyerahan.
3. **PHP dan MySQL, Soal 8.5–8.7** — laporan dengan `GROUP BY` dan `HAVING`, pemeriksaan yang tak dapat dikelabui *injection*, *cookie* yang bukan sesi.
4. **XML dan JSON, Soal 8.8–8.10** — ekspor yang tetap *well-formed*, skema yang menangkap data ngawur, harga yang benar-benar bilangan.
5. ***Servlet* dan JSP, Soal 8.11–8.12** — *model*, *view*, dan *controller*; empat tempat menyimpan sebuah bilangan.

Setiap soal menempuh jalan yang sama: soal, dengan bahasa kita sendiri, rancangan, penyelesaian, baris demi baris, penyelesaian keliru yang menggoda, dan uji diri. Buku: Subbab 8.1.

Titik Tolak Kita: Kotak Perkakas

Bab 5: peladen yang berjalan, *redirect*, *cookie* dan sesi, model relasional dan normalisasi, SQL dari `CREATE TABLE` sampai `HAVING`, `mysqli` dan PDO, *prepared statement*, transaksi.

Bab 6: *well-formedness*, entitas, pohon, *namespace*, *well-formed* dibandingkan *valid*, DTD dan XML Schema, DOM, SAX, XPath, XSLT, dan JSON.

Bab 7: *container*, *servlet* dan siklus hidupnya, JSP, objek implisit, empat *scope*, *bean*, EL, JSTL, *forward*, dan JDBC lewat DAO.

> **CARA MEMAKAI PERTEMUAN INI.** Bacalah pernyataan soal, tutup sisanya, dan pikirkan sepuluh menit: sketsakan tabelnya, tulis URL setiap halaman, tulis satu pernyataan SQL dengan tangan. Lalu bandingkan.

Penyelesaian yang sudah Anda gulati sendiri dipahami dengan cara yang tidak pernah dicapai oleh penyelesaian yang hanya dibaca. Buku: Subbab 8.1.

Metodenya, Langkah 1 sampai 4

> **DEFINISI.** *Kebutuhan* adalah satu kalimat yang menyatakan apa yang harus dimungkinkan sistem bagi seseorang, atau yang tidak boleh terjadi. *Fitur* adalah sekelompok halaman dan tindakan yang memenuhi satu atau beberapa kebutuhan bagi satu jenis pengguna.

1. **Orang dan kata kerja:** pelanggan, kasir, pengunjung; *mendaftar, masuk, mencari, menambah, membayar, mengambil*.

2. **Kata benda menjadi tabel;** "memiliki" menjadi *foreign key* di sisi "banyak". **3. Setiap fakta di satu tempat**, kecuali salinan yang mencatat sejarah, seperti jumlah yang ditagihkan.

4. **Daftar URL:** membaca dengan `GET`, mengubah dengan `POST`, dan *redirect* sesudah setiap `POST` yang berhasil (Post/Redirect/Get).

Pemula mulai dengan menulis `<html>`: satu-satunya tempat yang tidak boleh dijadikan awal, karena setiap halaman bergantung pada data yang belum diputuskan. Buku: Subbab 8.2.

Metodenya, Langkah 5 sampai 8

5. Siapa boleh apa, untuk setiap URL, diperiksa di peladen pada setiap permintaan. Menyembunyikan tautan adalah kenyamanan, bukan perlindungan.

6. Tulis pemeriksaan sebelum kode, masing-masing dengan kasus tepi: pencarian kosong, kuantitas nol, pelanggan yang masih punya tagihan.

7. Satu permintaan pada satu waktu, dijalankan sesudah setiap rute. **8. Rahasia di luar kode,** dan laporan singkat yang ditulis sambil jalan.

> **URUTANNYA PENTING.** Data, lalu alamat, lalu izin, lalu pengujian, baru kemudian kode. Tim yang mulai dari Langkah 7 merancang ulang tabelnya pada malam terakhir.

Pada soal kecil, empat langkah pertama memakan dua puluh menit; pada soal besar, langkah-langkah itulah pembeda antara rancangan dan tumpukan halaman. Buku: Subbab 8.2.

Satu Basis Data, Satu Folder Kode

Tabel 8.1 Folder-folder utama kode bab ini. Setiap program menyebut, di 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
`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

Basis data Profitina Laundry: tiga gerai, empat belas karyawan, dua belas layanan, 256 pelanggan, 2.953 tagihan, 3.319 pembayaran, satu tahun yang berakhir 30 September 2026 pukul 21.00. SQLite secara bawaan; MySQL dengan satu variabel lingkungan (Tabel 8.1). Buku: Subbab 8.3.

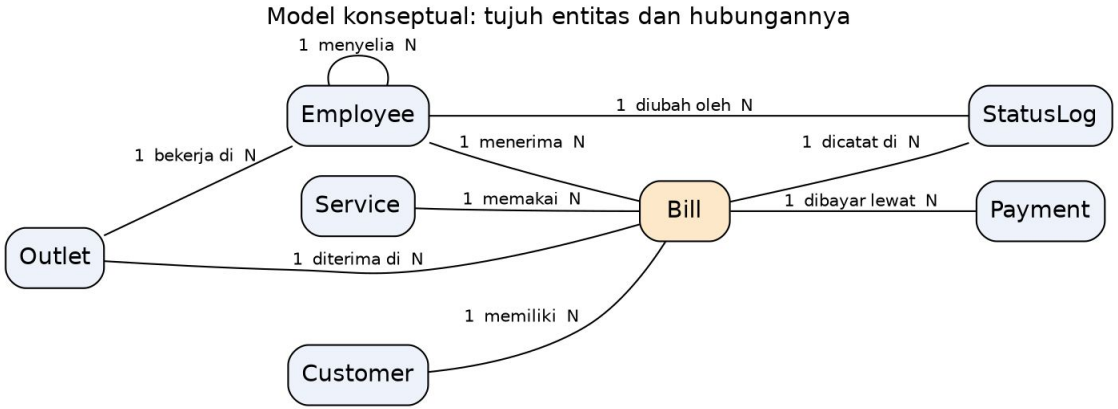
Proyek: Situs Web CRUD dengan Autentikasi

> **SOAL.** Bangunlah situs web CRUD dengan Laravel atau *framework* lain: sedikitnya delapan fitur di luar autentikasi; *login*, registrasi, dan *logout*; skema dengan sedikitnya satu *foreign key*. Tulis laporan berisi model konseptual dan fisik, fitur beserta tangkapan layarnya, dan bagian kerja setiap anggota; terapkan di hosting gratis dan terbitkan kode sumbernya.

Dengan bahasa kita sendiri: empat soal, diselesaikan menurut urutan ketergantungannya. **8.1** basis data; **8.2** registrasi, *login*, *logout*; **8.3** delapan fitur; **8.4** bukti dan penyerahan.

Laundry Desk *dibangun tiga kali tanpa framework (PHP dengan PDO, Java di peladen HTTP bawaan JDK, Python dengan Flask), masing-masing dengan cara MVC, sehingga pindah ke Laravel kemudian hanyalah terjemahan, bukan misteri. Buku: Subbab 8.4.*

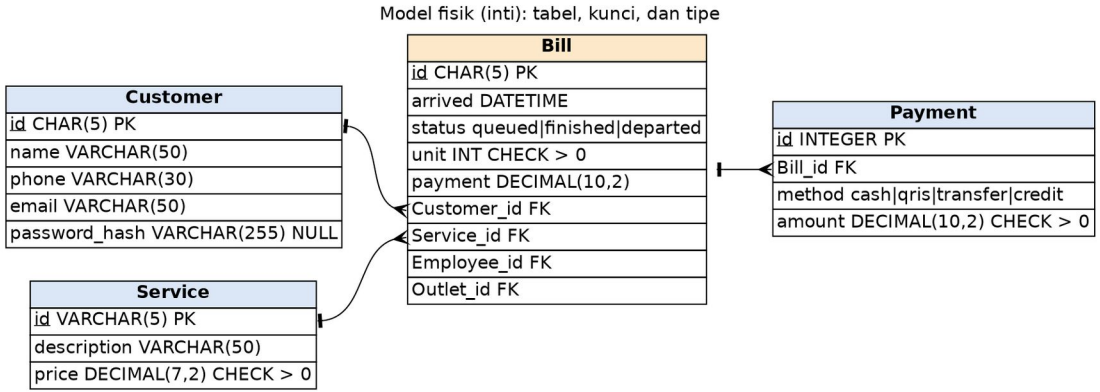
Soal 8.1 — Model Konseptual



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".

Model konseptual mendaftar entitas dan relasinya beserta kardinalitasnya. Bacalah setiap garis Gambar 8.1 dua arah: "satu pelanggan memiliki banyak tagihan; satu tagihan milik tepat satu pelanggan". Pemilik, yang tidak punya atasan, adalah alasan kolom atasan boleh kosong. Buku: Subbab 8.4.1.

Soal 8.1 — Model Fisik



Gambar 8.2 Inti model fisik: kunci primer bergaris bawah, setiap *foreign key* digambar dari kunci yang dirujuknya. Kaki gagak menandai ujung "banyak".

Gambar 8.2: kode yang mudah dibaca (`C0001`, `B2953`) karena dicetak di nota; uang sebagai `DECIMAL(10,2)`; `Bill.payment` adalah jumlah yang ditagihkan, disalin saat dibuat; `password_hash` boleh `NULL` bagi pelanggan tanpa akun. Buku: Subbab 8.4.1.

Soal 8.1 — Tabel `Bill` di MySQL

Kode 8.1 Tabel `Bill` dalam `p81_design.mysql.sql`: *constraint* bernama, sebuah `CHECK`, dan dua *foreign key* dengan `RESTRICT` dan `CASCADE`.

```
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;
```

Constraint pada Kode 8.1 diberi nama, sehingga galat dapat menyebut aturan mana yang dilanggar. `ENGINE = InnoDB` menegakkan foreign key; mesin MyISAM yang lama menerima klausanya lalu diam-diam mengabaikannya. Buku: Subbab 8.4.1.

Soal 8.1 — Menguji Kunci-Kuncinya

Kode 8.2 Empat pernyataan uji dalam `p81_design.mysql.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';
```

```
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)
```

Pada Kode 8.2 pernyataan 1 gagal dengan 1452, baris anak yang menunjuk ke ketiadaan; pernyataan 2 gagal dengan 1451, `C0001` masih punya `B0001`; pernyataan 3 berhasil; pernyataan 4 membawa `C0100` ke `Bill` lewat `ON UPDATE CASCADE`. Jalankan dengan `mysql -u root --force -t`. Buku: Subbab 8.4.1.

Soal 8.1 — Penyelesaian Keliru yang Menggoda

Hilangkan `PRAGMA foreign_keys = ON` di SQLite, dan setiap pernyataan "berhasil":

```
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
```

Tagihan untuk `C9999` tersimpan, `C0001` terhapus padahal tagihannya masih ada, dan tersisa dua tagihan yatim. SQLite hanya memeriksa foreign key bila setiap koneksi memintanya, sebab itu setiap program menyalakannya pada baris kedua kode basis datanya. Buku: Subbab 8.4.1.

Soal 8.1 — Uji Diri

> **UJI DIRI.** Tabel `Pickup(id, Bill_id, courier, pickup_at)` mencatat kurir yang menjemput cucian; setiap penjemputan milik tepat satu tagihan, dan satu tagihan dijemput paling banyak sekali. *Constraint* mana yang menyatakan "paling banyak sekali"?

Foreign key `Pickup_Bill FOREIGN KEY (Bill_id) REFERENCES Bill (id)` menyatakan "tagihan yang sungguh ada".

`UNIQUE (Bill_id)` menyatakan "paling banyak sekali": relasinya menjadi satu-ke-satu.

`CREATE TABLE` lengkapnya, dengan `ON DELETE RESTRICT ON UPDATE CASCADE` dan `ENGINE = InnoDB`, ada di buku. Buku: Subbab 8.4.1.

Soal 8.2 — Registrasi, *Login*, *Logout*

> **SOAL.** Situs web harus memiliki fitur *login*, registrasi, dan *logout*.

Registrasi untuk pelanggan: baris `Customer` baru dengan *hash* kata sandi. ***Login*** mencari di `Employee`, lalu di `Customer`, dan mengingat *staf* atau *pelanggan*.

Aturan 1. Jangan pernah menyimpan kata sandi: simpan *hash* yang lambat dan bergaram, PBKDF2 dengan HMAC-SHA256 dan 600.000 putaran.

Aturan 2. Simpan identitas di peladen. **Aturan 3.** *Session id* baru saat *login* (melawan *session fixation*); musnahkan sesi saat *logout*.

Bentuk tersimpannya `pbkdf2_sha256$600000$<salt>$<hash>`*, 90 karakter. Buku: Subbab 8.4.2.*

Soal 8.2 — Membuat dan Memeriksa *Hash*

Kode 8.3 `hash_password` dalam `lib.php`: garam 16 bita yang baru dan 600.000 putaran PBKDF2.

```
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);
}
```

Kode 8.4 Inti `verify_password` dalam `lib.php`: hitung ulang dengan garam dan jumlah putaran yang tersimpan, lalu bandingkan dalam waktu konstan.

```
    $salt = base64_decode($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
```

Pada Kode 8.3 garamnya baru untuk setiap kata sandi; Kode 8.4 lebih dulu menolak bentuk tersimpan yang tidak dikenalnya, lalu membandingkan dengan `hash_equals`. Java memakai `SecretKeyFactory` dan `MessageDigest.isEqual`; Python `hashlib.pbkdf2_hmac` dan `hmac.compare_digest`. Buku: Subbab 8.4.2.

Soal 8.2 — Rute *Login*

Kode 8.5 Cabang POST dari `/login` dalam `public/index.php`: cari di `Employee`, lalu di `Customer`; bila berhasil, *session id* baru, lalu penggunanya.

```
    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');
        }
```

Pada Kode 8.5 pesannya sama untuk alamat yang tidak dikenal dan kata sandi yang salah, sehingga formulir tidak dapat membocorkan siapa yang punya akun. Sesi dimulai dengan `cookie_httponly` dan `cookie_samesite => 'Lax'`. Buku: Subbab 8.4.2.

Soal 8.2 — Penyelesaian Keliru yang Menggoda

"Kata sandi cukup diacak": `hash('sha256', $password)`, tanpa garam, tanpa putaran. Dua pelanggan memilih kata sandi yang sama:

```
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, dan SHA-256 itu cepat: miliaran tebakan per detik pada kartu grafis. Dengan PBKDF2 dan garam baru, bentuk tersimpannya berbeda dan setiap tebakan menelan 600.000 putaran. Java dan Python mencetak tujuh baris yang sama. Buku: Subbab 8.4.2.

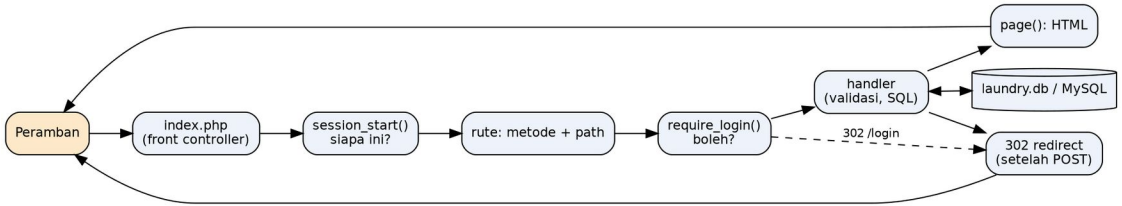
Soal 8.3 — Sembilan Fitur

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 huruf CRUD muncul pada Tabel 8.2, dan setiap fitur adalah URL yang dapat dipanggil sebuah pengujian. F9 memberi registrasi Soal 8.2 sebuah kegunaan. Buku: Subbab 8.4.3.

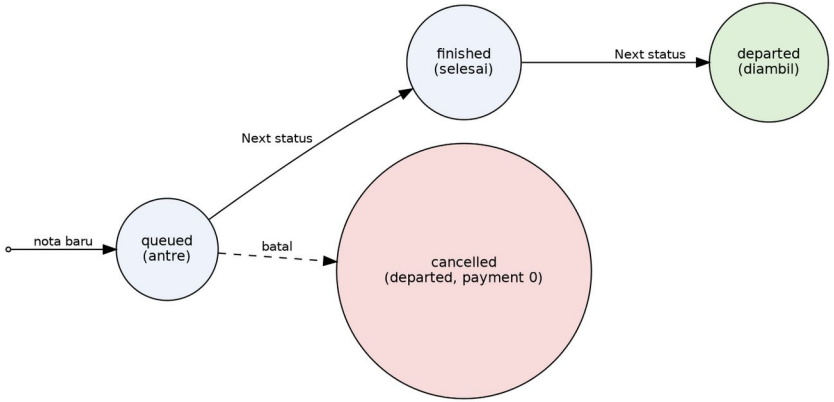
Satu Permintaan Melewati Laundry Desk



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.

Gambar 8.3: sesi, rute, pemeriksaan izin, validasi, basis data; halaman untuk `GET`, redirect sesudah `POST` yang mengubah sesuatu. Buku: Subbab 8.4.3.

Riwayat Sebuah Tagihan



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.

Gambar 8.4: `queued`, `finished`, `departed`; status tidak pernah mundur, dan setiap perpindahan menambah baris `StatusLog` yang menyebut karyawannya. Buku: Subbab 8.4.3.

F6: Tagihan Baru, Harganya dari Peladen

Kode 8.6 Cabang penyimpanan `/bills/new` dalam `public/index.php`: satu transaksi, id berikutnya, harga disalin ke tagihan, dan baris log pertama.

```
        } else {
            $pdo = db();
            $pdo->beginTransaction();                // nota beserta baris log pertamanya, atau tidak
keduanya
            $id = next_id('Bill', 'B');
            $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");
```

Sebelum Kode 8.6, handler memeriksa pelanggan, layanan, `unit` 1 sampai 100, dan raknya. Harga tidak pernah menjadi kolom formulir: tagihan yang diambil dari peramban adalah tagihan yang dapat dipilih pelanggan. Buku: Subbab 8.4.3.

F4: Basis Data Berkata Tidak

Kode 8.7 Fitur F4 dalam `public/index.php`: coba `DELETE`; *foreign key* `Bill_Customer` menolaknya selama tagihan masih ada.

```
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');
}
```

Kode 8.7 membiarkan basis data menegakkan aturannya dan melaporkannya dengan kata-kata. F8 menolak pembayaran yang melebihi sisa tagihan, dihitung sebagai tagihan dikurangi `COALESCE(SUM(p.amount), 0)`*. Buku: Subbab 8.4.3.*

Di Mana MVC-nya, dan di Mana Laravel Menaruhnya

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)

Model adalah basis data beserta constraint-nya ditambah SQL-nya; view mengubah data menjadi HTML; controller merutekan dan memutuskan (Tabel 8.3). Perintah Laravel ada di buku tanpa keluaran: perintah itu memerlukan unduhan yang tidak dapat dilakukan mesin penyusun. Buku: Subbab 8.4.3.

Soal 8.3 — Memercayai Peramban

Penyelesaian yang menggoda: `<input name="unit" type="number" min="1" max="100" required>` menolak 0, jadi peladen tidak perlu memeriksa.

Mengapa keliru: siapa pun dapat mengirim permintaan tanpa formulir, dengan `curl` atau Python; pemeriksaan penerimaan F6 mengirim 0 dan ketiga peladen menjawab "The quantity must be a whole number from 1 to 100."

> **UJI DIRI.** Fitur kesepuluh, "keluhan": tabel mana, status apa, pemeriksaan apa? **Jawaban:** baris `StatusLog` dengan `status = 'complaint'`; peladen memeriksa bahwa penggunanya staf, tagihannya ada, dan statusnya `departed`.

Pemeriksaan peramban adalah kesopanan bagi pengguna yang jujur, bukan perlindungan. Buku: Subbab 8.4.3.

Soal 8.4 — Bukti yang Dapat Diulang

> **SOAL.** Tunjukkan bahwa setiap fitur bekerja, tulis laporannya, terapkan situs di hosting gratis, dan terbitkan kode sumbernya di repositori publik.

Uji penerimaan memakai situs persis seperti peramban: mengirim formulir, menyimpan *cookie*, tidak pernah mengikuti *redirect* sendiri, dan hanya membaca tanda yang tetap (`<h1>`, `<p class="msg">`, `<p class="err">`, `<tr><td>`).

Karena tidak tahu apa pun selain itu, `acceptance.py` yang sama memeriksa peladen PHP, Java, dan Python; `run_acceptance.sh` memberinya basis data baru setiap kali.

Tangkapan layar membuktikan fitur pernah bekerja sekali; pengujian membuktikannya lagi dalam dua detik sesudah setiap perubahan. Buku: Subbab 8.4.4.

Contoh Terkerjakan: Uji Penerimaan, Autentikasi

```
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
```

A1 sampai A8 membuktikan Soal 8.2 beserta kasus tepinya: surel yang sama dua kali, kata sandi yang salah, pelanggan di halaman staf, halaman sesudah logout. Buku: Subbab 8.4.4.

Contoh Terkerjakan: Uji Penerimaan, Fitur

```
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.
```

Penolakan juga diperiksa: pelanggan yang punya tagihan, 0 kg, pembayaran melebihi sisa, tagihan yang tak dapat maju. PHP, Java, dan Python mencetak baris yang sama; PHP juga lulus di MariaDB. `LAUNDRY_NOW` *membekukan jam, sehingga tagihan barunya selalu* `B2954`*. Buku: Subbab 8.4.4.*

Laporan

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

Tulislah Tabel 8.4 sambil membangun. Pernyataan kontribusi hanya dapat dipercaya bila dapat dicocokkan dengan repositori: “Putri 55%: basis data, login, F6 sampai F8, dan uji penerimaan”, bukan “Putri mengerjakan backend”. Buku: Subbab 8.4.4.

Repositori Tanpa Rahasia

```
--- 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
```

> **Rahasia yang sekali di-*push* tetap tinggal di riwayat.** `git log -p` memperlihatkannya kepada siapa saja. Satu-satunya perbaikan adalah *mengganti* rahasia itu.

Tanpa `.gitignore`, `git add .` akan menerbitkan `.env` dan data pribadi 256 pelanggan. Rahasia datang dari lingkungan; `env.example` di-commit dengan nama dan tanpa nilai. Terapkan lewat pengaturan lingkungan hosting, lalu jalankan uji penerimaan terhadap alamat publiknya. Buku: Subbab 8.4.4.

Soal 8.5 — Penerimaan per Gerai dan Metode

> **SOAL.** Untuk September 2026, setiap gerai dan metode pembayaran: jumlah pembayaran dan uang yang diterima, hanya kombinasi yang sedikitnya Rp 400.000, urut per gerai dan terbesar lebih dulu.

Rancangan, menurut urutan kerja basis data: `FROM Payment JOIN Bill`; `WHERE` menyaring *baris*; `GROUP BY` gerai dan metode; `HAVING SUM(p.amount) >= 400000` menyaring *kelompok*; `SELECT`, `ORDER BY`.

Rentang setengah terbuka: setiap saat mulai `2026-09-01` sampai, tetapi tidak termasuk, `2026-10-01`.

`WHERE` tidak boleh memakai `SUM`, karena ketika ia berjalan belum ada kelompok. Buku: Subbab 8.5.1.

Soal 8.5 — Kueri dan Pengikatannya

Kode 8.8 Kueri laporan dalam `p85_report.sql`, dengan rentang tanggal setengah terbuka.

```
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;
```

Kode 8.9 Dari `p85_report.php`: setiap *placeholder* diikat dengan tipe yang semestinya.

```
$st = db()->prepare($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();
```

Pada Kode 8.9, `PDO::PARAM_INT` bukan hiasan: `execute([...])` mengirim teks, dan di SQLite setiap bilangan bulat "lebih kecil" daripada teks apa pun, sehingga Kode 8.8 kembali kosong tanpa galat. MySQL mengonversi dan menyembunyikan masalahnya. Buku: Subbab 8.5.1.

Soal 8.5 — Laporannya

```
+--------+----------+----------+------------+
| 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 `credit` dan transfer di O02 (Rp 302.000) dibuang oleh `HAVING`. PHP, Java, dan Python mencetak laporan yang sama. Buku: Subbab 8.5.1.

Soal 8.5 — `BETWEEN` Kehilangan Hari Terakhir

```
| 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 |
```

`BETWEEN '2026-09-01' AND '2026-09-30'` berhenti pada *saat pertama* 30 September: tunai O01 kehilangan 3 pembayaran dan Rp 99.000, QRIS O03 satu pembayaran Rp 60.000; seluruhnya sembilan pembayaran senilai Rp 346.000.

Laporan yang salah satu hari dari tiga puluh tampak sepenuhnya masuk akal, dan itulah yang membuat kesalahan ini berbahaya. Pakailah `>= awal AND < hari berikutnya`, selalu. Buku: Subbab 8.5.1.

Soal 8.6 — Cek Status yang Tidak Dapat Dikelabui

Kode 8.10 Dari `p86_status.php`: SQL tetap dengan dua *placeholder*, dan versi menggoda yang menempelkan masukan ke dalam teksnya.

```
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;
}
```

Pelanggan mengetik nomor tagihan dan empat angka terakhir teleponnya; Kode 8.10 menjawab dengan status serta inisial dan nama belakang saja. `SUBSTR(c.phone, -4)` *bekerja di SQLite dan MySQL. Buku: Subbab 8.5.2.*

Soal 8.6 — Tiga Masukan, Dua Versi

```
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)
```

Serangan itu membuat teks versi tak aman menjadi `… WHERE b.id = 'x' OR 1=1 -- ' AND SUBSTR(c.phone, -4) = '' …`: petik menutup teks, `OR 1=1` benar untuk setiap baris, `--` menjadikan pemeriksaan telepon komentar.

Halaman tak aman membocorkan `B0001`, tagihan orang lain; di versi aman, karakter yang sama hanyalah nomor tagihan aneh yang tidak cocok dengan apa pun. Escaping manual harus diingat pada setiap pemakaian; placeholder tidak perlu diingat. Buku: Subbab 8.5.2.

Soal 8.7 — *Cookie* Bukanlah Sesi

```
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
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
```

Cookie adalah klaim pengunjung, bukan fakta: `Cookie: role=staff` *membuka halaman staf versi yang salah. Sesi menyimpan faktanya di peladen. Sesi bertanda tangan milik Flask dapat dibaca (muatannya Base64* `{"role":"customer"}`*) tetapi tidak dapat diubah: muatan palsu mendapat 403. Buku: Subbab 8.5.3.*

Soal 8.8 — Tagihan Sehari sebagai XML

Kode 8.11 Perulangan pembangun pohon dalam `p88_export.php`: DOM menulis tag dan meng-*escape* setiap karakter.

```
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');
}
```

Pada Kode 8.11 pengenal menjadi atribut dan hal yang dibaca orang menjadi isi; jumlah uang adalah bilangan bulat polos dengan `currency="IDR"`. Tidak ada tempat program menulis `&`: `saveXML()` yang melakukannya. Buku: Subbab 8.6.1.

Soal 8.8 — Hasil Ekspor, dan Versi Rangkaian String

```
<?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>
Glued string well-formed? no: xmlParseEntityRef: no name (column 76)
```

Wash, dry & fold menjadi `&` dan Ma'ruf tidak memerlukan apa pun. String rangkaian bertemu `& fold` pada kolom 76 dan parser menolak seluruh dokumen: kantor pusat tidak menerima apa-apa. Versi itu lolos setiap uji pada hari tanpa `&` dalam deskripsi. Buku: Subbab 8.6.1.

Soal 8.9 — Valid, Bukan Sekadar *Well-Formed*

Kode 8.12 Model isi sebuah `bill` dalam `bills.xsd`: tipe bernama, bilangan bulat positif, dan mata uang yang ditetapkan `IDR`.

```
        <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>
```

Bagian lain Kode 8.12: `Status` adalah enumerasi tiga kata, id tagihan cocok dengan `B[0-9]{4}`, dan `date` bertipe `xs:date`. Buku: Subbab 8.6.2.

Soal 8.9 — Tiga Kesalahan Ditemukan

Kode 8.13 `bills-broken.xml`: *well-formed*, dan keliru di tiga tempat.

```xml
<?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>
```

```
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'.
```

Kode 8.13 dapat diurai: well-formed. Skema menemukan `lost`, `five`, *dan* `USD`. *Berhenti sesudah parser berarti membukukan empat puluh ribu dolar AS sebagai rupiah. lxml Python mencetak hal yang sama;* `ErrorHandler` *Java mengumpulkan setiap galat. Buku: Subbab 8.6.2.*

Soal 8.10 — Harga yang Benar-Benar Bilangan

Kode 8.14 Dari `p810_prices.php`: harga di-*cast* ke bilangan bulat, dan `json_encode` menulis lariknya.

```
$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 8.15 `p810_client.js`: klien yang berhitung dengan harga-harga yang diterimanya.

```
// 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);
```

Kode 8.14 dengan cast; Kode 8.15 menjumlahkan harga cuci dan harga setrika. `int(p)` di Python paling penting dengan PyMySQL, yang `Decimal`-nya ditolak `json.dumps`. Buku: Subbab 8.6.3.

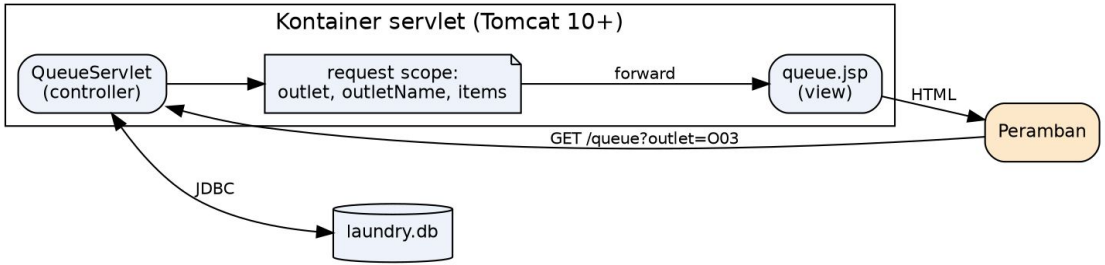
Soal 8.10 — Tanpa *Cast*, di MySQL

```
{
    "currency": "IDR",
    "services": [
        {
            "id": "S0001",
            "description": "Wash, dry & fold (per kg)",
            "price": "8000.00"
typeof price: string
wash + iron, per kg: 8000.006000.00
```

PDO mengembalikan `DECIMAL` dari MySQL sebagai teks, `json_encode` menulis string, dan `+` JavaScript merangkai string: `8000.006000.00`. Dengan *cast*: `typeof price: number` dan `14000`.

Perkalian menyembunyikan masalahnya (`"8000.00" * 2` bernilai 16000), sebab itu bug semacam ini bertahan sampai penjumlahan pertama. Buku: Subbab 8.6.3.

Soal 8.11 — Halaman Antrean sebagai MVC



Gambar 8.5 Halaman antrean dalam susunan MVC: *servlet* membaca dan memutuskan, model berpindah melalui *request scope*, halaman JSP hanya menampilkan.

Gambar 8.5: `QueueServlet` *memeriksa gerai, membaca tagihan dengan JDBC, menaruh* `outlet`, `outletName`, *dan* `items` *di scope request, lalu forward ke* `/WEB-INF/queue.jsp`*, yang tidak dapat dibuka peramban secara langsung. Buku: Subbab 8.7.1.*

Soal 8.11 — *Controller*

Kode 8.16 Akhir `QueueServlet.doGet`: baris menjadi *bean* `QueueItem`, model masuk ke *scope* request, dan permintaan di-*forward*.

```
            try (PreparedStatement st = c.prepareStatement(SQL)) {
                st.setString(1, outlet);
                ResultSet rs = st.executeQuery();
                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);
```

Pada Kode 8.16 kueri diurutkan dengan `b.status <> 'finished'`, sehingga tagihan yang sudah selesai tampil lebih dulu; gerai yang tidak dikenal mendapat `sendError(404, …)`. Servlet tidak menulis HTML. Buku: Subbab 8.7.1.

Soal 8.11 — *View*

Kode 8.17 Perulangan dalam `queue.jsp`: Expression Language untuk membaca, JSTL untuk mengulang, `c:out` untuk meng-*escape*.

```
<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>
```

`<td>${b.customer}</td>` tidak meng-*escape*. Hari ketika seorang pelanggan mendaftar dengan nama `<script>alert('hi')</script>`, setiap layar antrean menjalankannya. `|safe` di Jinja adalah lubang yang sama.

`${b.customer}` pada Kode 8.17 memanggil `getCustomer()`, sebab itu bean memerlukan getter. `${b.unit}` dibiarkan telanjang karena bertipe `int`. Buku: Subbab 8.7.1.

Soal 8.11 — Dijalankan di *Container* Pengganti

```
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
```

Mesin penyusun tidak dapat memasang container, jadi servlet dikompilasi terhadap jar API resmi dan dijalankan di `StubContainer.java`, yang mencetak model yang akan diterima halaman JSP. Versi Flask dan JDK menghasilkan halaman yang identik sampai ke bitanya. Buku: Subbab 8.7.1.

Soal 8.12 — Empat Tempat Menyimpan Sebuah Bilangan

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

Tabel 8.5: container membuat satu objek servlet dan mengirim setiap permintaan dari setiap pengunjung kepadanya, sering di beberapa thread sekaligus. Buku: Subbab 8.7.2.

Soal 8.12 — *Servlet*-nya

Kode 8.18 Dari `VisitServlet.doGet`: variabel lokal, *field*, atribut sesi, dan atribut aplikasi.

```
        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);
        }
```

Pada Kode 8.18 kedua penghitung bersama diperbarui di dalam blok `synchronized`, kalau tidak dua thread dapat membaca nilai lama yang sama dan kehilangan satu kunjungan. `init()` memasang atribut aplikasi bernilai 0. Buku: Subbab 8.7.2.

Contoh Terkerjakan: Rina, Rina, Bayu, Rina

```
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
```

local selalu 1; **field** dan **application** menghitung semua orang: 1, 2, 3, 4; hanya **session** yang menjawab "kunjungan Anda": Rina 1, 2, 3 dan Bayu 1.

Penyelesaian keliru yang menggoda menyimpan "kunjungan Anda" di field, dan kunjungan pertama Bayu menunjukkan 3. Di halaman JSP kesalahan yang sama ditulis `<%! int count = 0; %>`*. Buku: Subbab 8.7.2.*

Perbandingan Soal 8.1 sampai 8.6

Tabel 8.6 Soal 8.1 sampai 8.6: apa yang dilatih masing-masing dan kasus yang salah ditangani penyelesaian menggodanya.

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

Bacalah kolom terakhir Tabel 8.6: setiap penyelesaian yang menggoda bekerja pada contoh biasa dan gagal pada satu masukan tertentu. Buku: Subbab 8.8.

Perbandingan Soal 8.7 sampai 8.12

Tabel 8.7 Soal 8.7 sampai 8.12: apa yang dilatih masing-masing dan kasus yang salah ditangani penyelesaian menggodanya.

Soal	Pokok	Gagasan utama	Yang keliru pada penyelesaian yang menggoda
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

Tabel 8.7 adalah alasan terkuat untuk Langkah 6: tuliskan kasus tepi sebelum menulis kode. Buku: Subbab 8.8.

Profitina dalam Praktik

Akun dan data pribadi: *hash* Argon2id, *login* dengan pembatas laju, ekspor dan penghapusan menurut UU PDP 27/2022; setiap fitur adalah sekelompok rute dengan pemeriksaan izin di depannya.

Satu basis data, dua mesin: PostgreSQL di produksi, SQLite di sebagian jalur pengujian, dengan penjaga; Soal 8.5 menunjukkan alasannya.

Data terstruktur untuk mesin: schema.org, `robots.txt`, `/llms.txt`: *well-formed*, sesuai skema, dan bertipe benar.

Pengujian sebagai gerbang rilis: jauh lebih dari seribu pengujian; uji penerimaan Soal 8.4 dalam ukuran satu bab.

Produk memberi saran dan penggunanya yang memutuskan; versi yang dapat dijalankan dari setiap gagasan adalah kode Profitina Laundry. Buku: Subbab 8.9.

Kesalahan yang Sering Terjadi

Rancangan dan data: halaman sebelum tabel; *foreign key* yang tidak ditegakkan; uang sebagai bilangan titik mengambang atau string.

Akun: kata sandi yang dapat dibalik atau di-*hash* cepat tanpa garam; *session id* yang sama melewati *login*; memercayai *cookie*.

Permintaan: memeriksa hanya di peramban; mengubah data dengan `GET`; dua operasi tulis yang terkait tanpa transaksi; rahasia di repositori.

SQL, XML, JSON, JSP: `BETWEEN` pada stempel waktu; bilangan diikat sebagai teks; SQL dari masukan; string yang dirangkai menjadi XML atau JSON; berhenti pada *well-formed*; mencetak tanpa *escape*; data per pengunjung di *field servlet*.

Setiap kesalahan tercantum di buku beserta perbaikannya dan soal yang memperlihatkannya. Buku: Subbab 8.10.

Ringkasan: Lima Hal untuk Diingat

1. Selesaikan soal web dalam delapan langkah: data lebih dulu, lalu URL, izin, dan pengujian, baru kemudian kode.

2. Rancang pada dua tingkat, konseptual dan fisik; *foreign key* hanya sebaik penegakannya.

3. Simpan *hash* lambat yang bergaram, simpan identitas di peladen, ganti *session id* saat *login*, dan *redirect* sesudah setiap `POST` yang berhasil.

4. Saring baris dengan `WHERE` dan kelompok dengan `HAVING`, pakai rentang setengah terbuka, ikat setiap nilai dengan tipenya; tulis XML dan JSON dengan pustaka dan validasi apa yang tiba.

5. *Servlet* mengisi *scope* request lalu *forward* ke *view* yang hanya menampilkan dan selalu meng-*escape*; data per pengunjung tempatnya di sesi.

Tiga aplikasi dalam tiga bahasa di dua sistem basis data sepakat sampai ke rupiahnya. Buku: Subbab 8.11.

Pertemuan Berikutnya

Bab 9 — ASP.NET dan Rich Internet Application. Ajax, platform .NET, Razor Pages di .NET 10, dan Web Forms yang diperlihatkan sebagai konsep.

Status tagihan langsung Profitina Laundry sebagai fitur Ajax, dengan kontrak yang sama dilayani ASP.NET Core, PHP, Java, dan Python.

Sebelum itu: kerjakan Soal Latihan 8.1, 8.2, 8.5, 8.7, dan 8.10, dan pasang .NET 10 SDK.

Penyelesaian setiap soal latihan, beserta keluaran nyata setiap kueri, ada di buku. Buku: Subbab 8.12 dan 8.13.

Kuliah 9 — ASP.NET dan Rich Internet Application

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 9

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini sebuah halaman web belajar mengubah satu bagian dirinya tanpa dimuat ulang, dan kita berkenalan dengan teknologi server terakhir yang kita pelajari: ASP.NET Core di atas .NET 10. Buku: Bab 9, Subbab 9.1–9.17.

Cakupan Pertemuan Ini

1. **Rich internet application** — apa arti “kaya”, masa *plugin*, dan mengapa HTML5 menang.
2. **Ajax** — meminta *data*, bukan halaman: `XMLHttpRequest`, `fetch()`, *promise*, dan JSON.
3. **.NET dan ASP.NET** — dari Web Forms pada 2002 hingga ASP.NET Core di atas .NET 10 LTS; memasang SDK.
4. **Razor Pages** — `Program.cs`, halaman dan *page model*, sintaks Razor, *handler*, *binding*, validasi, *tag helper*.
5. **Web Forms sebagai konsep** — kontrol server, *postback*, *view state*, UpdatePanel, dan bentuk modernnya.
6. **Profitina Laundry** — status nota langsung dengan ETag dan *polling*, satu *front end* JavaScript, empat server.

Kedua bagian bertemu pada fitur Laundry: halaman Ajax yang dilayani Razor Pages, lalu oleh PHP, Java, dan Python. Buku: Subbab 9.1.

Posisi Kita: Klik, Tunggu, Halaman Baru

Bab 1: **peramban meminta, server menjawab**. Bab 2 sampai 7 mengisi kedua sisinya: HTML, CSS, PHP, MySQL, XML, *servlet*, dan JSP.

Di setiap bab sejauh ini iramanya sama: pengunjung mengeklik, server membangun **halaman baru yang utuh**, peramban membuang halaman lama.

Program desktop tidak berperilaku begitu: tekan sebuah tombol, **satu angka berubah**, dan segala hal lain tetap di tempatnya.

Dua pertanyaan hari ini: **bagaimana halaman web dapat terasa seperti program desktop?** dan **apa itu ASP.NET**, *framework* server di balik banyak bank, kantor, dan pabrik?

Subbab 9.2 dan 9.3 menjawab yang pertama, Subbab 9.4 sampai 9.9 yang kedua, dan Subbab 9.10 menyatukannya. Buku: Subbab 9.1.

Apa Arti “Kaya”

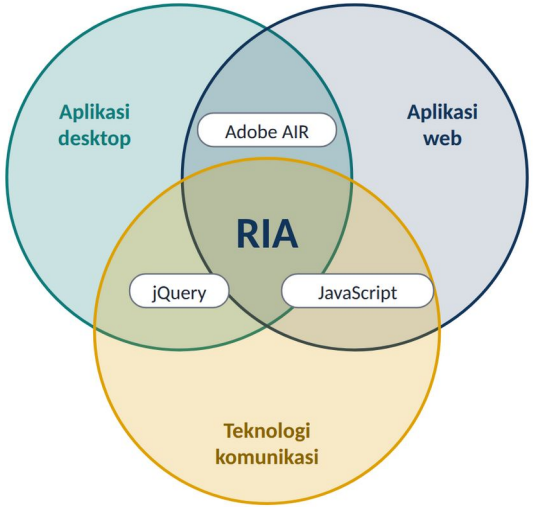
> **DEFINISI. Rich internet application** (RIA), disebut juga *rich web application*, adalah aplikasi web yang memiliki banyak ciri perangkat lunak desktop: tanggapan seketika, bagian layar yang memperbarui dirinya sendiri, seret-dan-lepas, penyuntingan di tempat. Ia dikirim melalui peramban dan tidak perlu dipasang.

Aplikasi desktop: seketika, tetapi harus dipasang dan diperbarui di setiap mesin. **Aplikasi web:** tanpa pemasangan, diperbaiki untuk semua orang sekaligus — tetapi secara klasik satu perjalanan pulang-pergi dan satu halaman utuh per tindakan.

Uji kekayaan: ketika satu hal berubah, apa yang terjadi pada sisa halaman? Pada aplikasi yang kaya, sisanya tetap persis di tempatnya, termasuk apa yang sudah Anda ketik dan posisi gulir Anda.

RIA berusaha memiliki kedua kelebihan itu. Aplikasi satu halaman (SPA) adalah kerabatnya, yang dipisahkan di Subbab 9.3.7. Buku: Subbab 9.2.1.

Tiga Lingkaran Beririsan



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.

Lingkaran ketiga pada Gambar 9.1, teknologi komunikasi, sering dilupakan: RIA menjadi kaya karena ia berbicara dengan server di latar belakang, sedikit demi sedikit, sementara pengguna terus bekerja. Buku: Subbab 9.2.1.

Sebelum HTML5: Zaman *Plugin*

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

Tabel 9.1: plugin adalah program terpisah yang dipasang ke dalam peramban dan mengambil alih sepetak halaman. Flash paling sukses; applet lebih tua; Silverlight adalah jawaban Microsoft atas Flash. Buku: Subbab 9.2.2.

Ke Mana Perginya *Plugin*

Perangkat bergerak. iPhone (2007) tidak pernah menjalankan Flash; ketika banyak pengunjung sama sekali tidak dapat melihat halaman Flash, membangunnya tidak lagi masuk akal.

Keamanan. *Plugin* menjalankan apa pun yang diberikan halaman; Flash, Java, dan Silverlight termasuk program desktop yang paling sering diserang, hingga peramban memblokir lalu membuangnya.

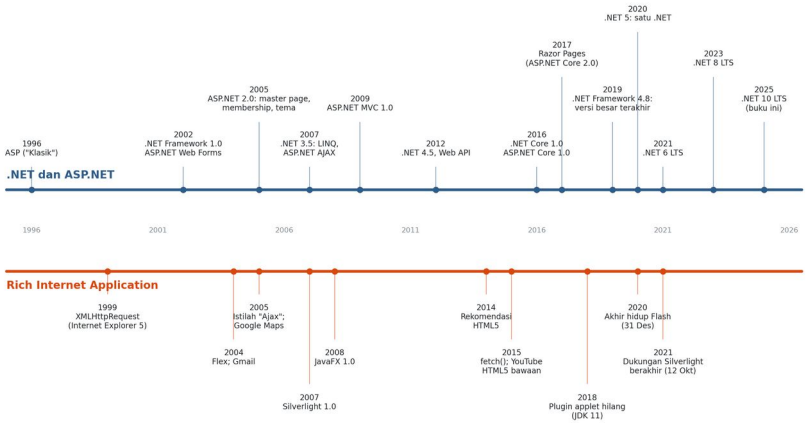
HTML5 membuatnya tidak diperlukan: *canvas* dan WebGL, audio dan video, penyimpanan, *worker*, WebSocket, `XMLHttpRequest` — tertanam di setiap peramban, tidak dimiliki perusahaan mana pun.

YouTube: aplikasi Flash pada 2005, HTML5 sebagai bawaan pada Januari 2015, *streaming* dengan MPEG-DASH lewat HTTP biasa.

> **WAWASAN KUNCI.** Setiap teknologi RIA yang memerlukan *plugin* di dalam peramban telah mati, betapa pun baik mutunya dan siapa pun pendukungnya. Yang bertahan adalah apa yang dipahami peramban itu sendiri: HTML, CSS, JavaScript, dan HTTP.

Sejak 2019 hanya ada satu HTML Living Standard, dipelihara WHATWG dan diperbarui terus-menerus. Buku: Subbab 9.2.3 dan 9.2.4.

Dua Sejarah Berdampingan



Gambar 9.2 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).

Gambar 9.2 adalah peta seluruh pertemuan ini: garis .NET di atas, dari ASP klasik hingga .NET 10 LTS, dan garis RIA di bawah, dari `XMLHttpRequest` pada 1999 hingga berakhirnya Flash dan Silverlight. Buku: Subbab 9.2.3.

Ajax: RIA Tanpa *Plugin*

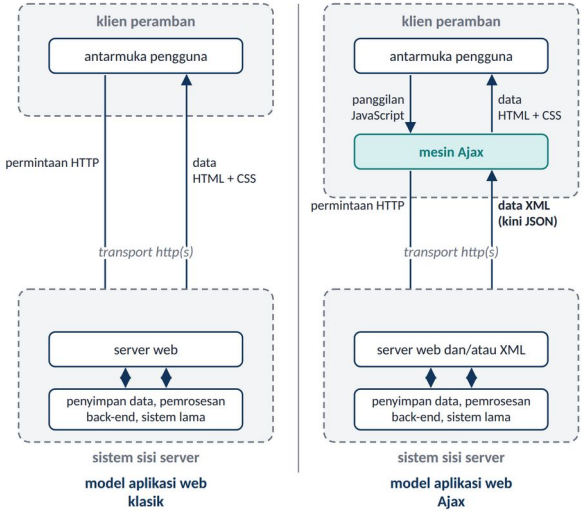
> **DEFINISI. Ajax** adalah teknik mengirim permintaan HTTP dari JavaScript selagi sebuah halaman ditampilkan, tanpa meninggalkan halaman itu, lalu memakai jawabannya untuk mengubah sebagian halaman melalui DOM. *Asinkron* berarti skrip tidak berhenti untuk menunggu: halaman tetap dapat dipakai, dan sebuah fungsi dijalankan kemudian saat jawaban tiba.

Dinamai pada Februari 2005 oleh Jesse James Garrett, setelah Gmail (2004) dan Google Maps (2005): *Asynchronous JavaScript and XML*.

Pada model klasik aktivitas pengguna **berhenti total** di setiap permintaan; pada model Ajax aktivitas itu tidak pernah berhenti — celah itulah seluruh perbedaannya.

Sebagian masukan dijawab mesin Ajax seketika, di peramban; sebagian lagi dikirim ke server, yang bekerja selagi pengguna melanjutkan. Buku: Subbab 9.3.1 dan 9.3.2.

Model Klasik dan Model Ajax



Gambar 9.3 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 J. J. Garrett (2005).

Di kiri Gambar 9.3, semua bab sejauh ini: permintaan HTTP, halaman utuh kembali. Di kanan, satu kotak ditambahkan di dalam peramban, mesin Ajax; yang berpindah adalah data — XML pada 2005, JSON kini. Buku: Subbab 9.3.1.

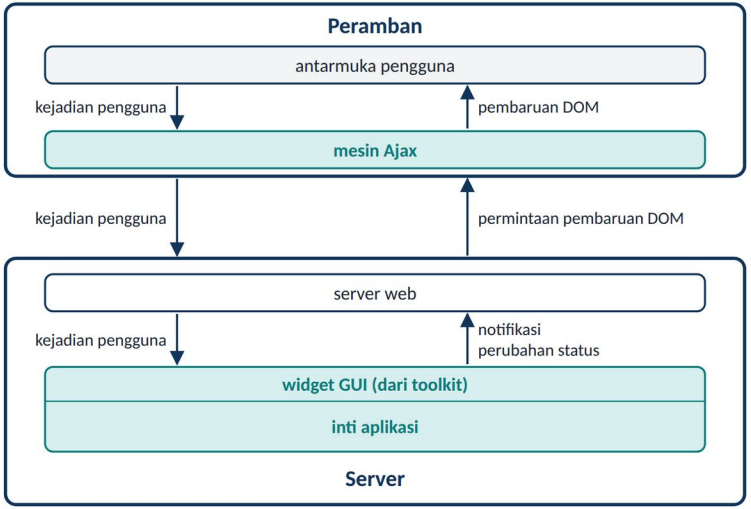
Bertanya Gaya 2005

Kode 9.1 `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
}
```

Kode 9.1: `open` *menyiapkan permintaan (*`true` *= asinkron),* `send()` *langsung kembali, dan fungsi* `onreadystatechange` *berjalan ketika jawaban lengkap. Di Chromium terhadap jalur Flask hasilnya* `200 B2952 In process 5 kg` *untuk telepon yang benar dan* `404 null` *untuk telepon yang salah. Buku: Subbab 9.3.2.*

Klien Tipis dan Klien Tebal



Gambar 9.4 *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 Gambar 9.4 server menyimpan keadaan layar setiap pengunjung. Pada klien tebal, widget berjalan di peramban dan memanggil layanan web hanya untuk data. Tipis dan tebal adalah dua ujung sebuah penggeser; halaman Laundry dekat ujung tipis. Buku: Subbab 9.3.3.

fetch() dan *Promise*

Kode 9.2 `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);
```

`fetch()` mengembalikan promise; `await` hanya menjeda fungsi itu, tidak pernah halamannya. Kode 9.2 bertanya dua kali: jawaban pertama `200 "B2952-1-1"` beserta datanya, yang kedua `304` dengan badan kosong, karena permintaannya membawa versi yang sudah dimiliki. Buku: Subbab 9.3.5.

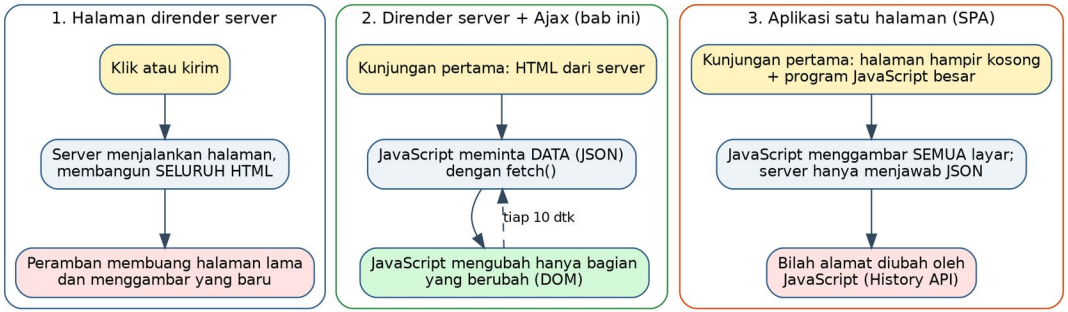
JSON: Kontraknya

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

Tabel 9.2 adalah kontrak antara empat server pertemuan ini dan satu front end. Tanpa nama, alamat, atau surel: UU PDP No. 27/2022 menuntut minimisasi data untuk halaman yang dapat dibuka siapa saja yang memegang nomor nota dan nomor telepon. Buku: Subbab 9.3.6.

Tiga Cara Membangun Aplikasi Web



Gambar 9.5 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.

Gambar 9.5: (1) halaman dirender server, semua yang sejauh ini; (2) halaman dirender server dengan Ajax, pertemuan ini; (3) aplikasi satu halaman, tempat JavaScript menggambar setiap layar. Buku: Subbab 9.3.7.

Memilih di Antara 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

Tabel 9.3: kolom tengah menjaga halaman tetap berfungsi tanpa JavaScript — progressive enhancement. SPA memindahkan pekerjaan, keadaan, dan bug ke peramban; pilihlah untuk layar yang dipakai berjam-jam. Buku: Subbab 9.3.7.

Apa Itu .NET

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, mengumpulkan sampah memori, dan menegakkan keamanan tipe. **Pustaka kelas** adalah kumpulan besar kelas siap pakai bagi setiap program .NET. Sebuah *bahasa* .NET — C#, F#, VB.NET — adalah bahasa apa saja yang kompilatornya menghasilkan IL.

Perbandingan dengan Bab 7 tepat sekali: CLR untuk JVM, IL untuk *bytecode*, pustaka kelas untuk pustaka Java, **C# untuk Java**.

Sebelumnya: CGI (satu program per permintaan) dan ASP klasik (VBScript di dalam halaman, diinterpretasi, hanya Windows). ASP.NET (2002) **mengompilasi** halaman menjadi kelas dalam bahasa bertipe kuat.

.NET Framework yang asli hanya berjalan di Windows, dan fakta itu membentuk dasawarsa berikutnya. Buku: Subbab 9.4.1 dan 9.4.2.

Dari ASP.NET ke ASP.NET Core dan Satu .NET

2002–2010: Web Forms di atas .NET Framework; versi 2.0 menambahkan *master page* dan *membership*, 3.5 ASP.NET AJAX dan MVC, 4.0 *routing* dan sintaks penyandian `<%: %>`.

2016: ASP.NET Core 1.0 dan .NET Core — ramping, modular (NuGet), sumber terbuka, lintas platform, dengan server web sendiri, **Kestrel**. **2017:** Razor Pages.

2019: .NET Framework dibekukan pada 4.8. **2020:** .NET 5 membuang kata “Core” dan menjadi *satu-satunya* .NET; versi baru 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). **.NET 10**, dirilis November 2025, adalah LTS saat ini, didukung hingga November 2028.

Razor Pages (2017) berdiri di samping Web Forms, MVC, minimal API, dan Blazor; ia paling dekat dengan yang sudah kita kenal: satu berkas menjawab alamatnya sendiri, seperti PHP, dengan `OnGet` dan `OnPost` seperti `doGet` dan `doPost` pada servlet. Buku: Subbab 9.4.3–9.4.5.

Memasang dan Memeriksa SDK

Pasang **SDK**, bukan hanya *runtime*: Windows `winget install Microsoft.DotNet.SDK.10`; macOS `.pkg` atau Homebrew; Ubuntu `sudo apt install dotnet-sdk-10.0`. Lalu, di VS Code, **C# Dev Kit**.

```
dotnet --version
dotnet --list-sdks
dotnet --list-runtimes
```

Kode 9.3 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]
```

Kode 9.3: dua digit pertama harus `10.0`. `Microsoft.AspNetCore.App` adalah shared framework — Kestrel, routing, Razor Pages, JSON, antiforgery — sehingga proyek web tidak perlu mengunduh apa pun. Buku: Subbab 9.5.1–9.5.3.

Satu Perintah, Satu Aplikasi Web

```
dotnet new webapp -o FirstApp
cd FirstApp
dotnet run --urls http://127.0.0.1:8901
```

Kode 9.4 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.
```

`dotnet new webapp -o FirstApp` membuat proyek; `dotnet run` memulihkan, mengompilasi, dan menjalankannya. Kode 9.4: aplikasi itu adalah server webnya (Kestrel), menyala dalam sekitar satu detik, dan lingkungannya Development. Buku: Subbab 9.6.1.

Anatomi Proyek Razor Pages

```
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
```

Folder `Pages` adalah peta situs: jalur sebuah berkas di bawah `Pages` adalah alamatnya. `wwwroot` berperan sebagai document root Apache; `Program.cs` sebagai `web.xml` dan kontainer pada Bab 7. Buku: Subbab 9.6.2.

Program.cs: Layanan

Kode 9.5 Baris-baris pertama `Program.cs` dari templat: membuat *builder*, mendaftarkan layanan, membangun aplikasi.

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorPages();

var app = builder.Build();
```

Tanpa kelas dan tanpa `Main`: C# mengizinkan *top-level statement* dalam satu berkas.

`CreateBuilder(args)` membaca konfigurasi (berkas `appsettings`, variabel lingkungan, baris perintah), menyiapkan *logging*, dan menyiapkan **kontainer layanan**.

Layanan adalah objek apa pun yang mungkin diminta aplikasi kelak; kontainer membuat dan menyerahkannya — **injeksi dependensi**.

Kode 9.5: `AddRazorPages()` mendaftarkan apa yang dibutuhkan Razor Pages. Setiap program ASP.NET Core punya tiga bagian yang sama: layanan, pipeline, jalankan. Buku: Subbab 9.6.3.

Program.cs: *Pipeline*

Kode 9.6 Sisa `Program.cs` dari templat: setiap baris `Use…` atau `Map…` menambahkan *middleware*, dalam urutan *pipeline*; `app.Run()` menyalakan Kestrel.

```
// 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();
```

Kode 9.6 adalah rantai filter Bab 7 dalam bentuk lain. Halaman galat Development jangan pernah tampil di server publik. Buku: Subbab 9.6.3.

Halaman dan *Page Model*-nya

Kode 9.7 `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>
```

`@page` memberi berkas alamatnya: `Pages/Index.cshtml` menjawab `/`, `Pages/Privacy.cshtml` menjawab `/Privacy`.

Untuk setiap permintaan ASP.NET Core membuat *page model* **baru**, memanggil *handler* yang cocok (`OnGet`), merender *markup*, lalu membuang objek itu.

Kode 9.7: tidak ada yang bertahan antarpermintaan kecuali disimpan di tempat yang bertahan — aturan PHP, kebalikan dari field servlet bersama yang diperingatkan Bab 7. Buku: Subbab 9.6.4.

Razor: Satu Karakter, @

Kode 9.8 Paruh pertama `Pages/Razor.cshtml`: blok C# berisi variabel, lalu ekspresi eksplisit dan implisit, pemanggilan metode, penyandian, `@Html.Raw`, alamat surel, dan `@@`.

```
@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>
```

Kode 9.8: `@(2 + 3)` eksplisit; `@kilos` implisit dan berakhir di spasi, sehingga perkalian perlu tanda kurung. `@note` mencetak `<b>`: Razor menyandikan HTML untuk semua yang dicetaknya. Buku: Subbab 9.7.1.

Razor: Keputusan, Perulangan, Komentar

Kode 9.9 Paruh kedua `Pages/Razor.cshtml` (tanpa `@switch`-nya): `@if` dan `@foreach` memilih dan mengulang HTML; `<text>` menulis teks dari blok kode; komentar Razor dan komentar HTML.

```
    @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>
    }
    <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. -->
<p>Rendered at @DateTime.Now.ToString("HH:mm:ss") by the server.</p>
```

Pada Kode 9.9 kurung kurawal `@if` dan `@foreach` kembali berisi HTML; `@switch` bekerja dengan cara yang sama. `@* … *@` dibuang di server; `<!-- … -->` terkirim ke setiap peramban. Buku: Subbab 9.7.1.

Razor di Samping PHP dan JSP

Tabel 9.4 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 /* … */ ?>`	`<%-- … --%>`

Tabel 9.4: di Razor bentuk pendek adalah bentuk yang aman. `@Html.Raw` hanya untuk markup buatan program Anda sendiri — jangan untuk ketikan pengunjung. Setiap halaman berbagi satu bingkai, `_Layout.cshtml`, yang `@RenderBody()`-nya menandai lubang yang diisi halaman. Buku: Subbab 9.7.1 dan 9.7.2.

Page Model yang Meminta Kebutuhannya

Kode 9.10 Kelas dalam `Pages/Index.cshtml.cs` milik LiveStatus: konstruktor primer meminta `ILaundryStore`; `OnGet` mengisi dua *property* untuk daftar harga.

```
// 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();
    }
}
```

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.

Kode 9.10 tidak pernah menulis `new CsvLaundryStore(...)`*: ia tidak tahu implementasi mana yang diterimanya, sehingga penyimpanan CSV dan SQLite dapat saling menggantikan tanpa halaman berubah. Buku: Subbab 9.8.1.*

Handler: Metode Mana yang Menjawab

Tabel 9.5 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` (merender *markup*) atau `IActionResult`: `Page()`, `RedirectToPage(…)`, `new JsonResult(data)`, `StatusCode(304)`.

Tabel 9.5: kebebasan itulah kunci fitur Laundry — satu halaman dapat menjawab peramban dengan HTML dan skrip dengan JSON. Buku: Subbab 9.8.2.

Model Binding: Kelas Masukan

Kode 9.11 Kelas masukan dalam `Pages/Quote.cshtml.cs`: atribut validasi menggambarkan formulir; `[Display]` menyediakan label yang dibaca orang.

```
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);
```

Kode 9.11: binding menyalin field `Input.ServiceId` dan `Input.Amount` ke property ini, mengubah `"2.1"` menjadi `decimal` 2,1; validasi memeriksa `[Required]` dan `[Range]` dan mencatat kegagalan di `ModelState`. Buku: Subbab 9.8.3.

Validasi dan Harga di OnPost

Kode 9.12 Paruh pertama `OnPost` dalam `Pages/Quote.cshtml.cs`: dua aturan yang tidak dapat dinyatakan atribut, dan formulir yang sama lagi bila `ModelState` tidak sah.

```
    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 9.12: bila `ModelState.IsValid` bernilai false, `Page()` mengembalikan formulir yang sama dengan nilai pengunjung dan pesannya. Penawaran harga tidak mengubah apa pun, jadi `Page()` tepat; handler yang mengubah data sebaiknya diakhiri `RedirectToPage` (Post/Redirect/Get). Bila sah, `OnPost` membulatkan ke atas dengan `Math.Ceiling` dan mengisi `Result`. Buku: Subbab 9.8.3 dan 9.8.5.

Aturan Harga, Dikerjakan

Dengan a jumlah yang dibawa dan p_s harga layanan s per satuan, Persamaan (9.1) membulatkan kilogram **ke atas** dan Persamaan (9.2) mengalikannya:

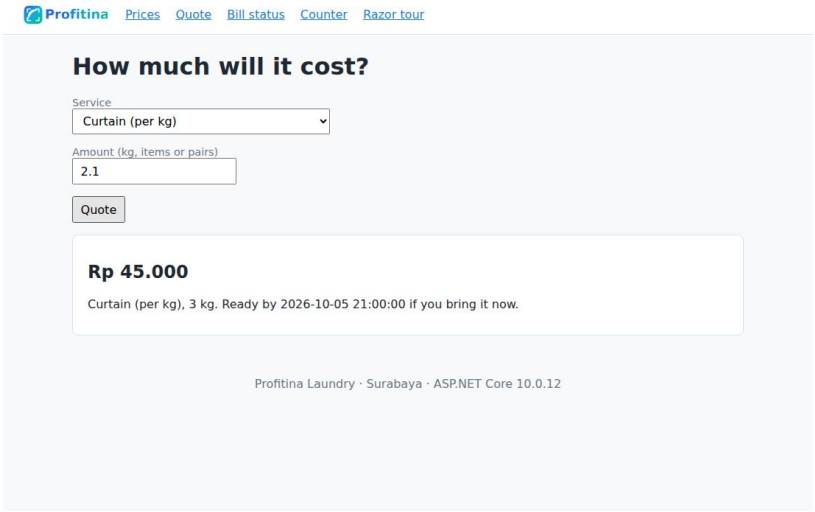
$$u = \lceil a \rceil \tag{9.1}$$

$$P = u \times p_s \tag{9.2}$$

CONTOH. 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. 2,5 *bed cover* (S0008, per potong) memenuhi `[Range]`, tetapi `OnPost` menambahkan “A whole number of items, for example 2.”

Untuk potong dan pasang, a sudah bulat; fungsi ceiling adalah `Math.Ceiling` dalam C#. Buku: Subbab 9.8.3.

Halaman Penawaran Harga



Gambar 9.6 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.

Gambar 9.6 persis contoh tadi: Chromium headless di mesin build mengirim 2,1 kg gorden dan menggambar Rp 45.000 untuk 3 kg, siap lima hari kemudian. Buku: Subbab 9.8.3.

Tag Helper dan Token *Antiforgery*

Kode 9.13 Formulir dalam `Pages/Quote.cshtml`: `asp-for`, `asp-items`, dan `asp-validation-for` adalah *tag helper* yang menulis nama, id, opsi, dan pesan.

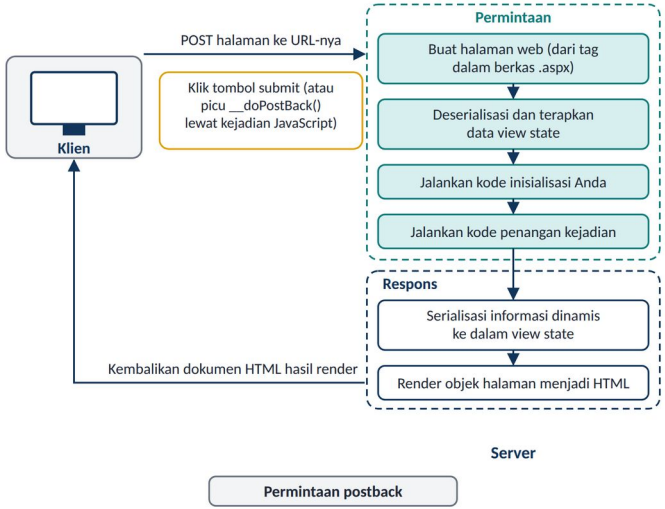
```
<form method="post">
    <p>
        <label asp-for="Input.ServiceId"></label>
        <select asp-for="Input.ServiceId" asp-items="Model.ServiceItems"></select>
        <span asp-validation-for="Input.ServiceId" class="error"></span>
    </p>
    <p>
        <label asp-for="Input.Amount"></label>
        <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>
```

Tag helper adalah kelas C# yang berjalan di server dan menulis ulang elemen yang cocok; formulir juga menerima `__RequestVerificationToken` tersembunyi.

CSRF: halaman di situs *lain* membuat peramban pengunjung mengirim POST ke situs *Anda*, lengkap dengan *cookie*. Razor Pages memeriksa token untuk setiap *handler* `POST` secara otomatis.

Kode 9.13: helper juga menambahkan field tersembunyi `__Invariant`, sehingga `2.1` tidak pernah terbaca 21 di komputer berpengaturan Indonesia. Bab 7 harus menulis filter untuk CSRF; di sini POST tanpa token menerima 400. Buku: Subbab 9.8.4.

Postback dan *View State*



Gambar 9.7 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.

Gambar 9.7: klik tombol mengirim seluruh formulir kembali ke halaman yang sama; server membuat ulang halaman, memulihkan setiap kontrol dari `__VIEWSTATE` *tersembunyi, menjalankan* `Page_Load` *dengan* `IsPostBack == true`*, memanggil handler, lalu merender lagi. Buku: Subbab 9.9.3.*

Harga Menyembunyikan Web

Sintaks halaman: direktif `<%@ … %>`, blok `<script runat="server">`, blok *render* `<%= %>`, kontrol server seperti `<asp:Label runat="server">`.

Produktif untuk formulir internal: `btnQuote_Click` dipanggilkan untuk Anda, kontrol menyimpan nilainya, validator seperti `RequiredFieldValidator` duduk di *markup*.

Mahal ketika web tumbuh: *view state* ratusan kilobita per klik, satu formulir per halaman, HTML ditulis kontrol, setiap interaksi sebuah `POST`.

UpdatePanel memberi *perilaku* Ajax sambil mempertahankan *modelnya*: `Timer` mengirim seluruh formulir setiap 10 detik, dan server menjalankan seluruh siklus hidup halaman demi satu panel.

> **WAWASAN KUNCI.** Web Forms berusaha menyembunyikan web di balik model formulir desktop. Semua penggantinya 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`.

Versi modern status nota mengirim `GET` dengan satu header dan, bila tidak ada yang berubah, menerima `304 Not Modified` tanpa badan. Buku: Subbab 9.9.2–9.9.6.

Setiap Gagasan Web Forms, Versi Modern

Tabel 9.6 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

Tabel 9.6 juga peta Bab 11 sampai 15: GridView di Bab 13, validasi dan state di Bab 14, ADO.NET di Bab 15. Buku: Subbab 9.9.7.

Status Nota Langsung: Ceritanya

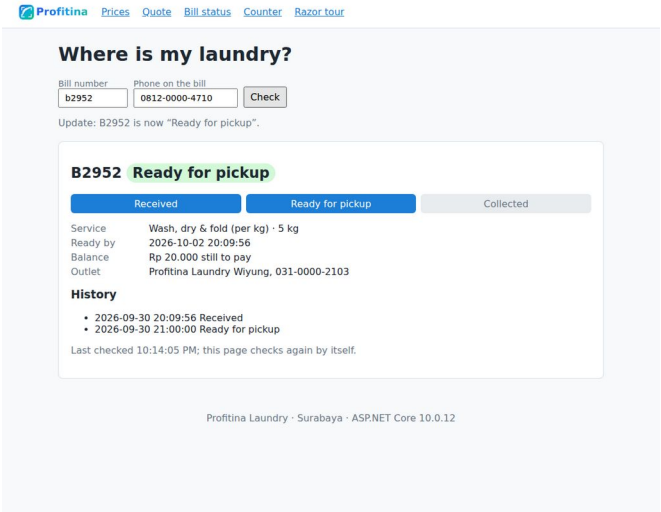
Seorang pelanggan membawa 5 kg ke gerai Wiyung pukul 20.09 pada 30 September 2026, membayar Rp 20.000 dari Rp 40.000, dan pulang membawa nota **B2952**.

Di rumah: *Where is my laundry?*, nomor nota dan telepon — *In process*, siap pukul 20.09 pada 2 Oktober, masih kurang Rp 20.000. Halamannya dibiarkan terbuka.

Di gerai, petugas cuci menekan *Mark finished*. Dalam sepuluh detik, **tanpa ada yang menyentuh halaman pelanggan**, lencananya berbunyi *Ready for pickup*.

Bilah kemajuan, riwayat, dan satu kalimat untuk pembaca layar ikut berubah. Halaman berikut menunjukkan hasilnya, ditangkap peramban headless mesin build. Buku: Subbab 9.10.1.

Yang Dilihat Pelanggan



Gambar 9.8 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.)

Gambar 9.8 tidak pernah dimuat ulang: lencana, bilah kemajuan, riwayat, dan pesan diubah oleh `live-status.js`. Di konter, Hand over kemudian ditolak karena masih ada utang Rp 20.000. Buku: Subbab 9.10.1.

Aturan Sebelum Kode

Waktu janji, dengan $h(s)$ lama pengerjaan layanan s (24 jam ekspres, 48 cuci-lipat, 120 gorden, ...), pada Persamaan (9.3); sisa tagihan setelah pembayaran $m_1, \ldots, m_k$ pada Persamaan (9.4):

$$t_{\text{promised}} = t_{\text{arrived}} + h(s) \tag{9.3}$$

$$\text{balance} = \text{payment} - \sum_{i=1}^{k} m_i \tag{9.4}$$

> **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: balance $= 40000 - 20000 = 20000$.

Jumlahnya 0 bila $k = 0$. Keempat jalur harus menerapkan persis aturan ini. Buku: Subbab 9.10.2.

Label Versi: ETag

Keadaan nota yang terlihat hanya berubah bila ditambahkan peristiwa status atau pembayaran, sehingga versinya adalah nomornya, banyak peristiwa statusnya e_b, dan banyak pembayarannya k_b:

$$v(b) = (b, e_b, k_b) \tag{9.5}$$

DEFINISI. ETag (*entity tag*) adalah teks pendek yang dikirim server dalam *header* `ETag` untuk melabeli versi sebuah jawaban. Klien yang memiliki versi itu mengirimkannya kembali dalam **`If-None-Match`**; bila masih cocok, server menjawab **`304 Not Modified`** tanpa badan.

Persamaan (9.5) ditulis `"b-e-k"`: `"B2952-1-1"` *di awal,* `"B2952-2-1"` *setelah Mark finished. Aturan konter:* `queued` → `finished` *kapan saja;* `finished` → `departed` *hanya bila sisa tagihan nol; selain itu* `409 Conflict`*. Buku: Subbab 9.10.2.*

Aturan Konter dalam Satu *Lock*

Kode 9.14 Inti `Advance` dalam `Data/CsvLaundryStore.cs`: di dalam satu `lock (gate)`, hitung sisa tagihan dan terapkan dua aturan status.

```
            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.");
            }
```

Kode 9.14: `case "finished" when balance > 0` *menyatakan "selesai, tetapi masih ada utang".* `lock` *berperan sebagai transaksi: dua kasir yang menekan bersamaan tidak mungkin sama-sama melihat* `queued`*. Halaman hanya melihat antarmuka* `ILaundryStore`*. Buku: Subbab 9.10.3.*

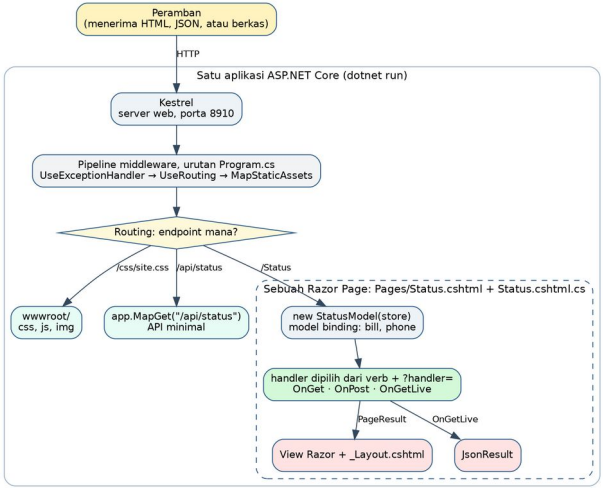
Satu *Minimal API* di Program.cs

Kode 9.15 Akhir `Program.cs` LiveStatus: `/api/status` dipetakan ke sebuah *lambda*.

```
// 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)
    {
        return Results.Json(p.Body, LiveJson.Options, statusCode: p.Status);
    }
});
```

Kode 9.15: minimal API, parameter diisi menurut nama dan tipe. Buku: Subbab 9.10.4.

Satu Permintaan Melintasi Aplikasi



Gambar 9.9 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.

Gambar 9.9: Kestrel menerima; middleware berjalan menurut urutan `Program.cs`; routing memilih berkas statis, minimal API, atau Razor Page; page model dibuat, diisi, lalu diminta menjalankan handler yang dinamai metode dan `?handler=`. Buku: Subbab 9.10.5.

HTML untuk Manusia, JSON untuk Skrip

Kode 9.16 `OnGetLive` dalam `Pages/Status.cshtml.cs`: *endpoint* Ajax di `/Status?handler=Live`, dengan ETag, 304, dan galat JSON.

```
    // 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)
        {
            return new JsonResult(p.Body, LiveJson.Options) { StatusCode = p.Status };
        }
    }
```

Kode 9.16: jawaban 304 tidak berbadan dan tanpa serialisasi sama sekali. Di sampingnya, `OnPost` menggambar jawaban yang sama di server untuk peramban tanpa JavaScript: halaman itu diperkaya secara progresif. Buku: Subbab 9.10.5.

Mesin Ajax: check()

Kode 9.17 Paruh pertama `check` dalam `live-status.js`: bertanya dengan `If-None-Match`, mundur, abaikan 304.

```
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();

  if (response.status === 304) {                       // same as before: nothing to draw
    timer = setTimeout(() => check(bill, phone), delay);
    return;
  }
```

Kode 9.17: `cache: "no-store"` menyingkirkan cache peramban; kegagalan jaringan menggandakan jeda (backoff). Buku: Subbab 9.10.6.

Mesin Ajax: Jawabannya

Kode 9.18 Paruh kedua `check`: jawaban galat menyembunyikan kartu dan berhenti; jawaban yang baik digambar, dan pemeriksaan berikut dijadwalkan kecuali nota sudah diambil.

```
    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);
  }
```

Kode 9.18: nota atau telepon yang salah tidak ditanyakan lagi. `draw` memasukkan setiap nilai dengan `textContent`, sehingga teks tetap teks. Buku: Subbab 9.10.6.

Mengirim Tanpa Memuat Ulang

Kode 9.19 *Listener* `submit` dalam `live-status.js`: `event.preventDefault()` mengubah formulir biasa menjadi formulir Ajax.

```
  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);
  });
```

Bilah alamat menerima `?bill=…` tetapi **bukan teleponnya**: alamat berakhir di riwayat, *bookmark*, dan tangkapan layar.

Setiap nilai digambar dengan `textContent`, tidak pernah `innerHTML`; baris pesan adalah wilayah `aria-live`; tab tersembunyi berhenti bertanya.

Tanpa baris pertama Kode 9.19, peramban akan mengirim formulir dan memuat halaman baru. Skrip yang sama, byte demi byte, melayani keempat jalur. Buku: Subbab 9.10.6.

Percakapan Selengkapnya



Gambar 9.10 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.

Gambar 9.10: pemeriksaan pertama menggambar nota; sepuluh detik kemudian 304 kosong; konter mengirim Mark finished; pemeriksaan berikutnya menerima 200 dengan ETag baru, dan halaman berubah. Buku: Subbab 9.10.6.

Menyaksikannya Bekerja

Kode 9.20 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]
```

Kode 9.20: jawaban 200, 304, 200; halaman pelanggan berubah 9,0 detik setelah klik. Skrip konter mengirim token antiforgery di header `RequestVerificationToken` *dan menonaktifkan tombolnya selama menunggu. Buku: Subbab 9.10.7 dan 9.10.8.*

Polling: Berapa Biayanya

Dengan N halaman terbuka yang masing-masing bertanya setiap T detik, server menerima Persamaan (9.6) permintaan per jam:

$$R = \frac{3600\,N}{T} \tag{9.6}$$

CONTOH. 300 pelanggan dengan $T = 10$ dtk: $R = 3600 \times 300/10 = 108000$ permintaan per jam, $108000/3600 = 30$ per detik, hampir semuanya 304 tanpa badan. 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. Buku: Subbab 9.10.9.

Dari Bertanya ke Diberi Tahu

Tabel 9.7 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

Tabel 9.7: bagi Laundry, polling setiap sepuluh detik sudah tepat — perubahan terjadi sekali atau dua kali sepanjang hidup nota, dan cara ini berjalan melewati setiap proxy dan di setiap ponsel. Buku: Subbab 9.10.9.

Alamat yang Sama dalam PHP

Kode 9.21 Cabang status dalam `php/router.php`: cari, pasang `ETag` dan `Cache-Control`, bandingkan dengan `If-None-Match`, jawab 304 atau JSON.

```
    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);
```

Kode 9.21 adalah `OnGetLive` lagi. Server bawaan PHP menjalankan `router.php` untuk setiap permintaan; ia menjawab `/api/…` dan mengembalikan `false` untuk yang lain, sehingga berkas statis dikirim apa adanya. Buku: Subbab 9.11.

Alamat yang Sama dalam Python

Kode 9.22 *View* status dalam `python/app.py`: lima langkah yang sama dalam Flask.

```
@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
```

Jalur PHP, Java, dan Flask menuntut `application/json` untuk permintaan *advance*, atau menjawab **415**: formulir biasa tidak dapat mengirim JSON lintas situs — itulah pertahanan CSRF mereka.

Kode 9.22: route, cari, versi, bandingkan, jawab. Jalur Java melakukan hal yang sama secara manual dengan `sendResponseHeaders(304, -1)`, di mana `-1` berarti tanpa badan. Buku: Subbab 9.12 dan 9.13.

Satu Kontrak, Empat Server

Tabel 9.8 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

Tabel 9.8: `tests/check_tracks.py` mengajukan pertanyaan yang sama ke setiap server dan membandingkan JSON hasil parsing — identik. Begitu peramban meminta data, front end dan back end hanya diikat oleh kontrak: alamat, kode status, dan bentuk JSON. Buku: Subbab 9.14.

Profitina dalam Praktik

Data, bukan halaman: *dashboard* `app.profitina.com` digambar dari satu panggilan JSON, `GET /api/me/dashboard` — kolom tengah Tabel 9.3.

Mendorong bila *polling* keliru: umpan komunitas memakai Server-Sent Events.

Pembaruan optimistis, penyandian tanpa kecuali, minimisasi data menurut UU PDP No. 27/2022.

Profitina adalah Python dengan FastAPI dan PostgreSQL, bukan .NET; gagasannya tetap berlaku karena menyangkut peramban dan HTTP. Studi kasus yang dapat dijalankan selalu Profitina Laundry. Buku: Subbab 9.15.

Kesalahan yang Sering Terjadi

Persiapan: memasang *runtime*, bukan SDK; mencampuradukkan .NET Framework dan .NET; menjalankan server publik dalam Development.

Razor Pages: lupa `@page`; salah menamai *handler*; mengirim POST dari skrip tanpa token *antiforgery*; mengurai angka dengan budaya mesin.

Keamanan: `@Html.Raw` atau `innerHTML` dengan data yang tidak Anda tulis; jawaban berbeda untuk "nota tidak ada" dan "telepon salah"; data pribadi di bilah alamat.

Ajax: lupa `event.preventDefault()`; *polling* terlalu sering atau tanpa henti; ETag yang tidak berubah saat datanya berubah; membiarkan *cache* peramban menyembunyikan 304.

Setiap kesalahan tercantum di buku beserta obatnya. Buku: Subbab 9.16.

Ringkasan: Lima Hal untuk Diingat

1. *Plugin* telah lenyap; HTML, CSS, JavaScript, dan HTTP bertahan. Ajax meminta **data** dan mengubah sebagian halaman.

2. .NET 10 adalah LTS hingga November 2028; ASP.NET Core berjalan di mana saja di atas Kestrel; Web Forms adalah sejarah dengan pewaris modern.

3. Razor Page adalah *markup* `@page` ditambah *page model* dengan *handler*; `Program.cs` mendaftarkan layanan, lalu membangun *pipeline*.

4. Razor menyandikan apa yang dicetaknya; *binding*, validasi, dan *tag helper* membentuk formulir; token *antiforgery* otomatis.

5. ETag membuat “tidak ada yang baru” murah; biaya *polling* adalah Persamaan (9.6); satu kontrak JSON membuat empat server berbagi satu *front end*.

Buku: Subbab 9.17.

Pertemuan Berikutnya

Bab 10 — Tiga Sistem Didemonstrasikan: PHP, JSP, ASP.NET. Satu aplikasi kecil dibangun tiga kali, dengan *login*, sesi, dan pola Post/Redirect/Get.

Sebelum itu: kerjakan Soal Latihan 9.1, 9.2, 9.4, dan 9.5 dengan tangan, lalu periksa setiap jawaban pada jalur yang sedang berjalan.

Penyelesaian kedua belas soal latihan ada di buku. Buku: Subbab 9.18 dan 9.19.

Kuliah 10 — Tiga Sistem Didemonstrasikan: PHP, JSP, dan ASP.NET

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 10

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini satu bagian kecil Profitina Laundry dibangun lima kali, dan kita belajar membaca salah satu dari kelimanya lalu menunjuk kembarannya di empat yang lain. Buku: Bab 10, Subbab 10.1–10.16.

Cakupan Pertemuan Ini

1. **Satu kontrak, lima bangunan** — kumpulan fitur, aturan toko, dan empat lapisan yang dimiliki setiap bangunan.
2. **Server-servernya** — XAMPP, Tomcat, IIS, dan .NET SDK di Windows, beserta padanan lintas platformnya.
3. **Bagian bersama** — aturan dalam satu berkas per bahasa; satu format *hash* kata sandi yang diverifikasi dalam empat bahasa.
4. **Lima bangunan** — PHP 8 dengan PDO, Jakarta Servlet dan JSP, HttpServer milik JDK, ASP.NET Core Razor Pages, Flask.
5. **Satu pengujian penerimaan** — delapan belas pemeriksaan dari luar; di mana bangunan-bangunan berbeda di jalur kabel.
6. **Berdampingan, *deployment*, dan mana yang dipilih.**

Bila kelimanya melakukan hal yang sama, hal itu adalah web; bila berbeda, perbedaannya adalah keputusan rancangan. Buku: Subbab 10.1.

Posisi Kita: Web atau Sistemnya?

Bab 3 dan 5 mengajarkan **PHP** dengan MySQL lewat PDO; Bab 7 *servlet* dan JSP **Java** di dalam *container*; Bab 9 **ASP.NET Core Razor Pages** di atas .NET 10.

Satu sistem pada satu waktu adalah cara yang tepat untuk mempelajari sistem — dan cara yang keliru untuk memahami web.

Apakah web yang mengharuskan teks di-*escape*, atau PHP? Apakah web membutuhkan *cookie* sesi, atau Tomcat? Apakah "*route*" itu gagasan, atau dekorator Flask?

Hari ini menjawabnya lewat **percobaan**: satu kumpulan fitur, lima bangunan, satu pengujian.

Empat dari lima bangunan dijalankan dan diuji lewat HTTP sungguhan; bangunan servlet dikompilasi dan dikemas tetapi tidak di-deploy, karena Tomcat tidak dapat diunduh di lingkungan build. Buku: Subbab 10.1.

Kontraknya

Tabel 10.1 Kumpulan fitur, bagian pertama: *login*, daftar pelanggan, dan formulir nota baru.

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

Tabel 10.1 tidak menyebut PHP, Java, C#, atau Python; ia menyatakan apa yang dikirim peramban dan apa yang diterimanya kembali. Hanya itu yang dapat diketahui peramban tentang server, dan karena itu satu pengujian dapat memeriksa lima program. Buku: Subbab 10.2.1.

Kontraknya, Lanjutan

Tabel 10.2 Kumpulan fitur, bagian kedua: halaman nota dan dua aturan yang berlaku untuk setiap alamat.

Metode dan alamat	Siapa yang boleh	Apa yang terjadi	Jawaban
`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`

Tabel 10.2: permintaan yang ditolak tidak pernah mencapai kode halaman. Status redirect mana yang dikirim setiap sistem — 302 atau 303 — merupakan salah satu temuannya. Buku: Subbab 10.2.1.

Tiga Aturan Toko

Peran: tanpa penyelia → **pemilik**; disebut sebagai penyelia orang lain → **penyelia**; selain itu **kasir** (bagian `transaction`) atau **petugas cuci** (bagian `wash`). Petugas cuci tidak menerima nota baru.

Nota yang sah: pelanggan seperti `C0001` yang ada; salah satu dari dua belas layanan; satuan bilangan bulat 1 sampai 100; rak `A01` sampai `C20`.

Tagihan, disalin ke dalam nota agar perubahan harga kelak tidak menulis ulang sejarah:

$$\text{ditagihkan} = \text{harga per satuan} \times \text{jumlah satuan} \tag{10.1}$$

> **CONTOH.** Tiga kilogram cuci-lipat ekspres (`S0002`, Rp 12.000 per kilogram): $12000 \times 3 = 36000$, ditulis Rp 36.000. Itulah nota yang dibuat setiap bangunan dalam pengujiannya.

Persamaan (10.1): masukan dipangkas dan diubah ke huruf kapital bila masuk akal, sehingga `c0001` dan `a05` diterima sebagai `C0001` dan `A05`. Buku: Subbab 10.2.1.

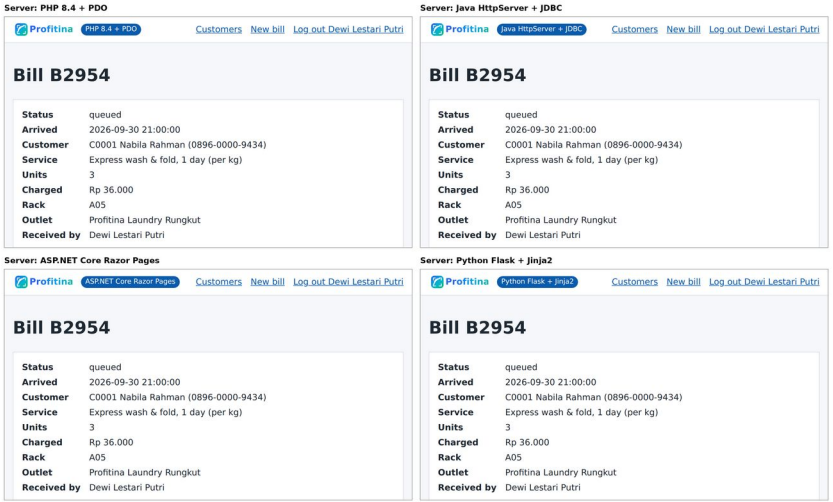
Lima Bangunan

Tabel 10.3 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

Tabel 10.3: semuanya membaca `Chapter10/data/laundry.db` yang sama, kecuali bangunan ASP.NET (Subbab 10.8). Kedua bangunan Java berbagi satu paket kelas model, basis data, dan aturan. Buku: Subbab 10.2.2.

Satu Nota, Empat Sistem



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.

Gambar 10.1: login yang sama, formulir yang sama, basis data yang sama — dan halaman yang sama. Hanya lencana di kepala halaman yang membedakan sistemnya. Buku: Subbab 10.2.2.

Empat Lapisan di Setiap Bangunan

Aturan — keputusan toko, tanpa web dan tanpa SQL: `rules.php`, `Rules.java`, `Rules.cs`, `rules.py`.

Store — setiap pernyataan SQL, masing-masing dengan parameter terikat, dalam satu kelas atau modul.

Halaman — kode yang berjalan untuk satu alamat: fungsi PHP, *servlet*, kelas `PageModel`, *view* Flask.

Tampilan — HTML dengan lubang: templat PHP, JSP, *text block* Java, `.cshtml`, Jinja2.

> **WAWASAN KUNCI.** Letakkan aturan di tempat yang tidak terjangkau *framework* mana pun. Empat ejaan dari satu kumpulan keputusan, tak satu pun mengimpor apa pun dari *framework* web: itulah yang memungkinkan Anda berganti *framework* tanpa mengubah aturan toko.

Inilah susunan Model–View–Controller dari Bab 7. Di bawahnya, constraint `CHECK` dan foreign key pada skema adalah pertahanan berlapis. Buku: Subbab 10.2.3 dan 10.4.1.

Tiga Server di Satu Mesin Windows

XAMPP: Apache, MySQL (MariaDB), dan PHP dalam satu pemasang, dengan panel kendali dan phpMyAdmin; `htdocs` adalah *document root*.

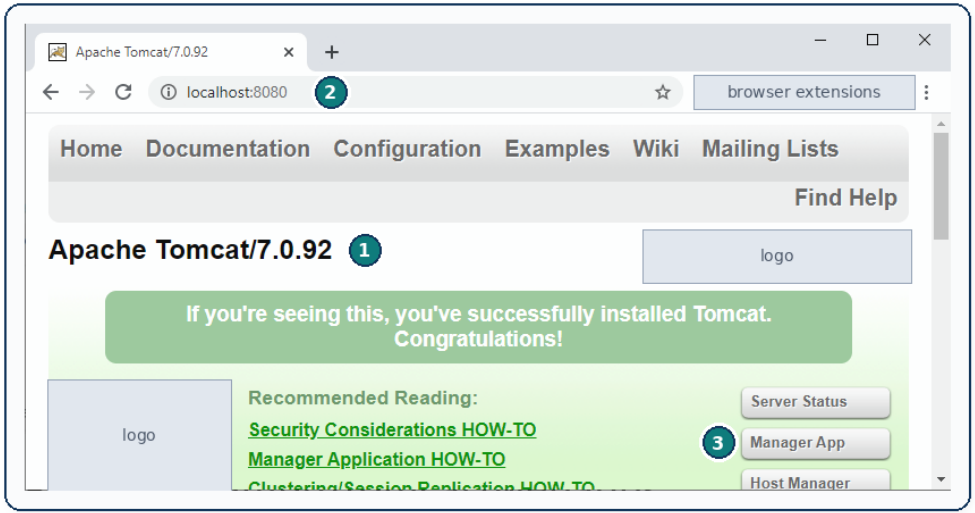
Tomcat di dalam XAMPP berasal dari era `javax` — 7.0 pada layar-layar ini, 8.5.96 pada XAMPP 8.2.12 sekarang: WAR `jakarta` menjawab 404 di sana. Buku ini memakai Tomcat 11.

IIS adalah bagian Windows, dinyalakan lewat *Turn Windows features on or off*; **.NET SDK** memberi `dotnet new` dan `dotnet run`; Visual Studio hanya untuk Windows.

Tanpa Windows: `php -S`, Tomcat 11 dalam bentuk zip, Kestrel di mana saja, Flask di mana pun Python berjalan.

Port adalah nomor yang dipakai sistem operasi untuk mengantarkan lalu lintas ke program yang tepat; dua program tidak dapat mendengarkan di port yang sama. Buku: Subbab 10.3.1–10.3.7.

Tomcat yang Dibawa XAMPP



Gambar 10.2 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 10.2: versi 7.0.92. Nomor versi menentukan paket Java mana yang harus diimpor servlet Anda — `javax` di sini, `jakarta` sejak Tomcat 10. Buku: Subbab 10.3.2.

Aturan dalam Python: Pola dan Peran

Kode 10.1 Bagian pertama `python/rules.py`: tiga pola dan peran seorang karyawan, ditentukan dari baris `Employee`.

```python
"""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"
```

Kode 10.1 tidak mengimpor apa pun dari Flask dan tidak menjalankan SQL. `php/src/rules.php`, `Rules.java`, dan `Rules.cs` adalah terjemahan baris demi baris. Buku: Subbab 10.4.2.

Aturan dalam Python: Nota yang Sah

Kode 10.2 `validate_bill` dalam `python/rules.py`: bersihkan setiap *field*, lalu periksa; kembalikan nilai yang sudah bersih beserta galatnya.

```python
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
```

Kode 10.2: nilai bersih ikut dikembalikan untuk formulir dan store. Buku: Subbab 10.4.2.

Empat Rincian, Empat *Bug* Terhindar

`fullmatch`, bukan `match`: `C0001; DROP TABLE` diawali id yang sah. PHP memerlukan `^…$` dan pengubah `D`, atau `A05\n` lolos.

`[0-9]`, bukan `\d`: di Python 3 `\d` juga cocok dengan angka Arab-Hindi dan Dewanagari, dan `int()` mengonversinya.

Uji teksnya, baru konversi: `int("0.5")` memunculkan galat, `int(" 3 ")` tidak, `float("1e2")` bernilai seratus.

Kembalikan nilai yang sudah bersih, dan simpan hanya itu — jangan formulir mentahnya.

Masing-masing adalah bug dalam program sungguhan milik seseorang. Buku: Subbab 10.4.2.

Kata Sandi: Mengapa Iterasi Penting

Disimpan sebagai `pbkdf2_sha256$600000$<salt>$<hash>`: algoritme, jumlah iterasi, enam belas byte acak *salt*, 32 byte hasil, keduanya dalam Base64.

Mesin yang menghitung R *hash* tunggal per detik terhadap n iterasi menguji sekitar Persamaan (10.2) tebakan per detik:

$$G = \frac{R}{n} \tag{10.2}$$

CONTOH. Dengan $R = 10^{10}$ dan $n = 600000$: $G \approx 16667$ tebakan per detik, sehingga 10^9 kata sandi yang mungkin memerlukan sekitar 16,7 jam per karyawan, dan — karena *salt*-nya berbeda-beda — sekitar 233 jam untuk keempat belas karyawan. Satu SHA-256 tanpa *salt* hanya memerlukan sepersepuluh detik untuk semuanya.

Satu login memerlukan 0,16 sampai 0,24 detik pada empat bangunan yang berjalan, sebagian besar untuk perhitungan ini — harga yang tidak dirasakan siapa pun di konter. Buku: Subbab 10.4.3.

Satu Format *Hash* dalam PHP

Kode 10.3 `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 10.3: `true` meminta `hash_pbkdf2` mengembalikan byte mentah; `hash_equals` membandingkan dalam waktu konstan. Versi Java memakai `PBKDF2WithHmacSHA256` dan `MessageDigest.isEqual`, versi C# `Rfc2898DeriveBytes.Pbkdf2` dan `FixedTimeEquals`. Buku: Subbab 10.4.3.

Format yang Sama dalam Python

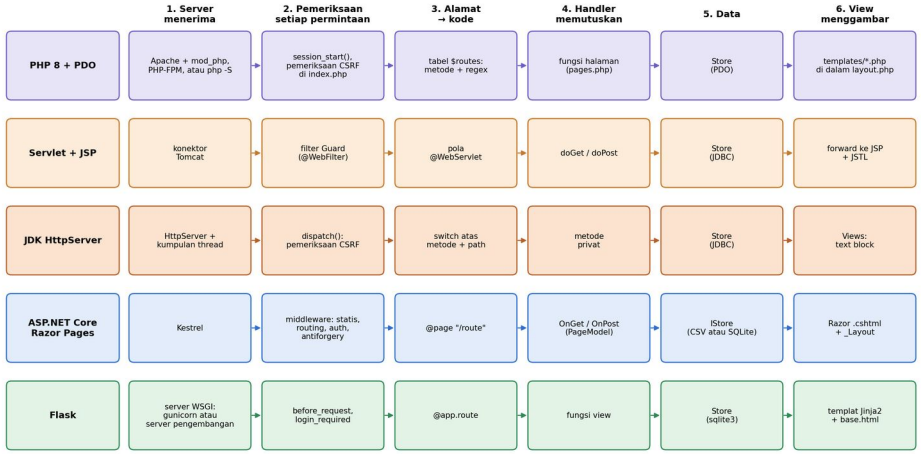
Kode 10.4 `verify_password` dalam Python: `hashlib.pbkdf2_hmac` dari pustaka standar.

```python
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
```

PERINGATAN. Tidak satu bangunan pun menyimpan atau membandingkan kata sandi polos, dan tidak satu pun menyebut *field* mana yang salah: “Wrong e-mail or password.” untuk keduanya, atau orang asing akan tahu alamat mana milik staf.

Kode 10.4: `==` biasa berhenti pada byte pertama yang berbeda dan membocorkan waktu; fungsi waktu-konstan memeriksa setiap byte. Hash yang rusak menjawab `false`, tidak pernah memunculkan pengecualian. Buku: Subbab 10.4.3.

Enam Tahap, Lima Bangunan



Gambar 10.3 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.

Gambar 10.3 adalah peta untuk sisa pertemuan: server menerima, pemeriksaan lintas-bagian berjalan, alamat menjadi kode, kode memutuskan, data dibaca atau ditulis, tampilan menggambar jawaban. Hanya nama di dalam kotaknya yang berubah. Buku: Subbab 10.5.

PHP: Tidak Ada yang Tersisa

***Shared-nothing*:** setiap permintaan menjalankan `public/index.php` dari baris pertamanya dengan variabel kosong; di akhir, semuanya dibuang.

Pemaaf — variabel yang rusak tidak memengaruhi pengunjung berikutnya — tetapi apa pun yang harus bertahan disimpan di luar program: berkas sesi, basis data.

Satu pintu: *front controller*; hanya `public/` yang terlihat oleh server web, sehingga kode sumber, templat, dan konfigurasi tidak dapat diminta.

PHP satu-berkas-per-alamat punya tiga kelemahan: setiap berkas harus ingat melakukan pemeriksaan, `.php` bocor ke URL, dan setiap berkas dapat diminta langsung. Buku: Subbab 10.5.1 dan 10.5.2.

Front Controller: Sesi dan Token

Kode 10.5 Paruh pertama `php/public/index.php`: berkas sungguhan dikembalikan ke server bawaan, kode sumber dimuat, sesi dimulai, dan setiap POST diperiksa tokennya.

```
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.');
}
```

Kode 10.5: cookie sesi bersifat `HttpOnly` (tak terlihat oleh skrip halaman) dan `SameSite=Lax`. Karena pemeriksaan CSRF ada di sini, sebelum kode halaman mana pun, tidak ada halaman yang dapat melupakannya. Buku: Subbab 10.5.2.

Front Controller: Tabel *Route*

Kode 10.6 Paruh kedua `php/public/index.php`: metode, pola, dan fungsi per baris; kecocokan pertama menang, dan bagian yang ditangkap menjadi argumen.

```
// 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.');
```

Pada Kode 10.6 urutan adalah makna: route spesifik seperti `/bills/new` didahulukan sebelum pola. Di Apache, `.htaccess` mengirim setiap alamat yang bukan berkas sungguhan ke `index.php`. Buku: Subbab 10.5.2.

Halaman PHP: *Login*

Kode 10.7 `login` dalam `php/src/pages.php`: 401 yang netral beserta formulirnya, atau id sesi baru dan sebuah *redirect*.

```
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');
}
```

Kode 10.7: surel dipangkas dan diubah ke huruf kecil; `session_regenerate_id(true)` *menggagalkan session fixation. Setiap fungsi halaman dideklarasikan* `: never` *— selalu mengakhiri permintaan. Buku: Subbab 10.5.3.*

Halaman PHP: Post/Redirect/Get

Kode 10.8 `create_bill` dalam `php/src/pages.php`: 422 beserta formulir dan pesannya, atau simpan nota lalu alihkan ke halamannya.

```
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");
}
```

Kode 10.8: setelah POST yang sah, permintaan berikutnya dari peramban adalah `GET` yang tidak berbahaya, sehingga Reload tidak pernah membuat nota kedua. `redirect` mengirim 303 See Other. Buku: Subbab 10.5.3.

Escaping dengan Tangan

Kode 10.9 `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>
```

Kode 10.9: escaping bersifat opt-in. `<?= $c['name'] ?>` *akan mencetak nama tanpa escaping dan tidak ada yang memperingatkan Anda.* `e()` *adalah* `htmlspecialchars` *dengan* `ENT_QUOTES` *dan* `ENT_SUBSTITUTE`*. Buku: Subbab 10.5.4.*

PDO: *Placeholder* dan LIKE

Kode 10.10 `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()];
    }
```

Kode 10.10: placeholder menjaga `%' OR 1=1 --` tetap sebagai teks; escape `!` membuat `%` dan `_` harfiah, atau pencarian `_` akan cocok dengan semua orang. Satu-satunya nilai yang disambungkan adalah konstanta `PAGE`. Buku: Subbab 10.5.6.

Servlet: Satu Objek, Banyak *Thread*

Tomcat membuat **satu** objek per *servlet* dan *filter* dan memanggilnya dari banyak *thread*; sebuah *listener* membangun satu `Store` bersama saat aplikasi mulai.

Baiknya: hal yang mahal dibangun sekali. Bahayanya: sebuah *field* dipakai bersama oleh setiap pengunjung — **jangan pernah menaruh data permintaan di *field***.

Pemetaan: **tepat** (`/bills/new`), **awalan jalur** (`/bills/*`, sisanya di `getPathInfo()`), **string kosong** (akar). Yang tepat mengalahkan awalan, apa pun urutan pemuatannya.

Di PHP urutan baris yang menentukan; di sini jenis polanya yang menentukan. Buku: Subbab 10.6.1 dan 10.6.2.

Filter: Satu Penjaga untuk Setiap Permintaan

Kode 10.11 Akhir `doFilter` dalam `Guard`: token diperiksa pada setiap POST, dan siapa pun yang belum *login* dikirim ke `/login`.

```
        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) {
            resp.sendRedirect(req.getContextPath() + "/login");
            return;
        }
        chain.doFilter(req, resp);
```

Kode 10.11: baris-baris sebelumnya memasang UTF-8 dan membuat token per sesi; `setCharacterEncoding` harus mendahului `getParameter` pertama; `chain.doFilter` meneruskan permintaan — filter yang kembali tanpanya telah menjawab sendiri. Buku: Subbab 10.6.3.

LoginServlet.doPost

Kode 10.12 Dari `doPost` milik `LoginServlet`: 401 netral yang sama, lalu `changeSessionId` sebelum pengguna disimpan.

```
        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");
```

Kode 10.12: `changeSessionId` adalah `session_regenerate_id` versi API servlet. `Pages.view` meneruskan ke JSP di bawah `WEB-INF`, yang tidak dapat dicapai URL mana pun secara langsung. Buku: Subbab 10.6.4.

Tanpa *Container*: Bangunan HttpServer

Kode 10.13 Tabel *route* dalam `dispatch` di `Main.java`: `switch` atas metode dan jalur, dan ekspresi reguler untuk `/bills/<id>` di cabang `default`.

```
            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.");
                    }
                }
            }
```

Kode 10.13: `dispatch` adalah container dan filter dalam satu metode — berkas statis, pemeriksaan CSRF, lalu `switch` (kecocokan tepat) dan regex (awalan). Sesi ditulis dengan tangan: id yang tak tertebak, `ConcurrentHashMap`, cookie `HttpOnly`. Buku: Subbab 10.7.1 dan 10.7.2.

Program.cs: Layanan

Kode 10.14 Dari paruh layanan `aspnet/Program.cs`: autentikasi *cookie*, dan *login* diwajibkan di setiap halaman kecuali satu.

```
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
});
```

Kode 10.14: dua konvensi itu menyatakan sekali apa yang diulang PHP di setiap halaman. Tepat di atasnya, `AddSingleton<IStore>` membangun satu store — `CsvStore`, karena NuGet (dan dengan itu driver SQLite) tidak terjangkau; `SqliteStore` terkompilasi dengan `-p:UseSqlite=true`. Buku: Subbab 10.8.1 dan 10.8.2.

Program.cs: *Pipeline*

Kode 10.15 Paruh *pipeline* `aspnet/Program.cs`: berkas statis, *routing*, autentikasi sebelum otorisasi, lalu halaman-halamannya.

```
// ---- 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();
```

Tidak ada kode CSRF: Razor Pages menolak setiap `POST` tanpa token *antiforgery* yang sah dengan **400**, dan *tag helper* formulir menambahkan tokennya.

Kode 10.15 adalah rantai filter yang merupakan seluruh aplikasi. Konfigurasi berlapis: berkas JSON, lingkungan (`Laundry__Store=sqlite`), baris perintah. Buku: Subbab 10.8.2.

Razor Page: *Route*, *Model*, Tampilan

Kode 10.16 `Customers.cshtml` (tanpa blok judulnya): *markup* yang sama dengan tampilan PHP dan JSP, dengan `@` beralih ke C# untuk satu ekspresi setiap kalinya.

```
@page "/customers"
@model CustomersModel
<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>
```

Kode 10.16: `@page "/customers"` *adalah route-nya; yang tidak ada adalah* `e()` *— setiap ekspresi* `@` *disandikan HTML. Pada* `/bills/new` *melawan* `/bills/{id}`*, segmen harfiah mengalahkan parameter. Buku: Subbab 10.8.3.*

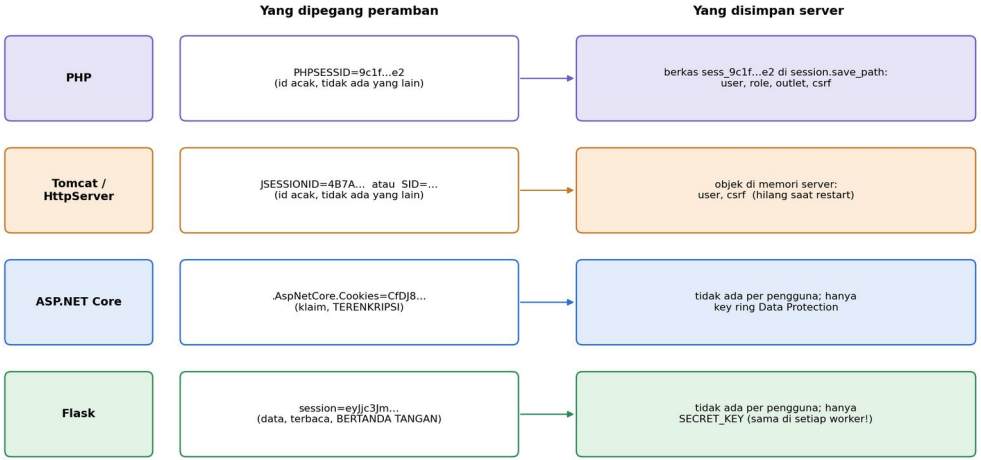
Login sebagai *Claim* dalam *Cookie*

Kode 10.17 Akhir `OnPostAsync` dalam `Login.cshtml.cs`: setelah pemeriksaan kata sandi yang sama, empat *claim* masuk ke *cookie* terenkripsi.

```
        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");
```

Kode 10.17: cookie dienkripsi dan ditandatangani Data Protection; server tidak mengingat apa pun per pengguna. `SignInAsync` menggantikan apa pun yang dipegang peramban, sehingga session fixation tidak punya celah. Buku: Subbab 10.8.4.

Di Mana *Login* Disimpan



Gambar 10.4 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.

Gambar 10.4: restart tidak melupakan login berbasis cookie dan banyak salinan menerima satu cookie bila berbagi kunci; tetapi cookie-nya lebih besar (560 karakter lawan 32) dan tidak dapat dicabut dengan menghapus sesi. Buku: Subbab 10.8.4.

Sebuah Temuan: Token yang Hilang

Kode 10.18 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  <--
```

Formulir *login* pertama ditulis `<form method="post" action="/login">`; *tag helper* menganggap `action` harfiah sebagai formulir ke situs lain dan tidak menyertakan token.

Perbaikannya satu atribut: `<form method="post" asp-page="/Login">` — setelah itu kedelapan belas pemeriksaan lolos.

Kode 10.18: bawaan yang aman hilang diam-diam, dan tertangkap oleh pengujian yang berperilaku seperti peramban. Buku: Subbab 10.8.5.

Flask: CSRF dan Dekorator *Login*

Kode 10.19 Dari `create_app` dalam `python/app.py`: pemeriksaan token 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
```

Kode 10.19: sebuah `context_processor` memberi setiap sesi tokennya; `before_request` adalah filter servlet; `teardown_appcontext` menutup koneksi seperti try-with-resources. `@login_required` bersifat opt-in per view, seperti `require_login()` di PHP. Flask adalah aplikasi WSGI (PEP 3333). Buku: Subbab 10.9.1 dan 10.9.2.

Flask: *View* Nota Baru

Kode 10.20 `new_bill` dalam `python/app.py`: 403 untuk petugas cuci, formulirnya, 422 beserta pesannya, atau notanya dan sebuah 303.

```
    @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"]),
                                      values["rack"], user["id"], user["outlet"], now())
        return redirect(url_for("show_bill", bill_id=bill_id), code=303)
```

Kode 10.20: route adalah dekorator; `/bills/new` dicocokkan sebagai harfiah sebelum `/bills/<bill_id>`. `url_for` membangun alamat dari nama view, seperti `RedirectToPage`. Buku: Subbab 10.9.2.

Jinja2: *Escaping* secara Bawaan

Kode 10.21 `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 %}
```

Kode 10.21: halaman memperluas layout dan mengisi blok-bloknya; setiap `{{ … }}` di-escape, dan HTML mentah memerlukan filter `safe` — frasa yang dicari dalam tinjauan kode. Buku: Subbab 10.9.3.

Sesi Flask Ditandatangani, Bukan Dienkripsi

```
{"user":{"id":"E0004","name":"Dewi Lestari Putri","outlet":"O01","role":"cashier"}}
```

Itulah bagian pertama *cookie* sesi kasir, setelah didekode dari Base64: **siapa pun yang memegang *cookie* dapat membacanya**.

Tak seorang pun dapat *mengubahnya* tanpa `SECRET_KEY`: sebuah cap waktu dan tanda tangan HMAC mengikuti datanya.

Aturannya: taruh di sesi Flask hanya apa yang boleh dilihat pengguna — jangan pernah rahasia.

Kunci itulah keamanan setiap login; `create_app` mengarang kunci acak bila `LAUNDRY_SECRET_KEY` tidak dipasang, dan itu jebakan di produksi. Buku: Subbab 10.9.4.

Satu Pengujian dari Luar

`acceptance/acceptance.py`, Python pustaka standar, berperilaku seperti **peramban yang cermat**: menyimpan *cookie*, mengirim *field* tersembunyi formulir, dan tidak mengikuti *redirect*.

Ia tidak pernah tahu nama *field* anti-pemalsuan — `csrf` di empat bangunan, `__RequestVerificationToken` di ASP.NET Core.

Pemeriksaannya adalah sifat keamanan: 04 dan 12 CSRF, 03 galat *login* yang netral, 09 *escaping*, 10 injeksi SQL, 17 *logout* yang sungguhan, 18 otorisasi.

Tak satu pun terlihat bila aplikasi hanya didemonstrasikan dengan mengeklik ke sana kemari. Buku: Subbab 10.10.1.

Hasilnya

Tabel 10.4 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	Subbab 10.5.8
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	Subbab 10.12.4
Servlet + JSP di Tomcat 11	8080	tidak dijalankan	Tomcat dan JSTL tidak tersedia (Subbab 10.6.7)

Tabel 10.4: setiap bangunan yang berjalan lolos kedelapan belas pemeriksaan — bangunan ASP.NET setelah perbaikan Subbab 10.8.5; bangunan servlet tidak dapat dijalankan di sini. Buku: Subbab 10.10.2.

Di Mana Bangunan Berbeda di Jalur Kabel

Tabel 10.5 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	—

Tabel 10.5: setelah POST, 303 See Other menyatakan maksudnya; 302 bekerja di setiap peramban. Token yang hilang mendapat 403 dari pemeriksaan buatan sendiri dan 400 dari ASP.NET Core. `ReturnUrl` harus diperiksa, atau menjadi open redirect. Buku: Subbab 10.10.3.

Dari Alamat ke Kode

Tabel 10.6 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

Tabel 10.6: semua route di satu tempat (PHP, HttpServer) atau setiap route di samping kodenya (servlet, Razor, Flask). Setiap mekanisme membuat `/bills/new` yang tetap menang atas pola; PHP bergantung pada urutan Anda. Buku: Subbab 10.11.1.

Templat dan *Escaping*

Tabel 10.7 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 10.7 memuat perbedaan keamanan terpenting bab ini: di tiga bangunan, escaping yang terlupa adalah lubang XSS yang tampak seperti kode yang benar; di dua bangunan, lupa itu aman. Pilih mesin templat yang melakukan escaping secara bawaan. Buku: Subbab 10.11.2.

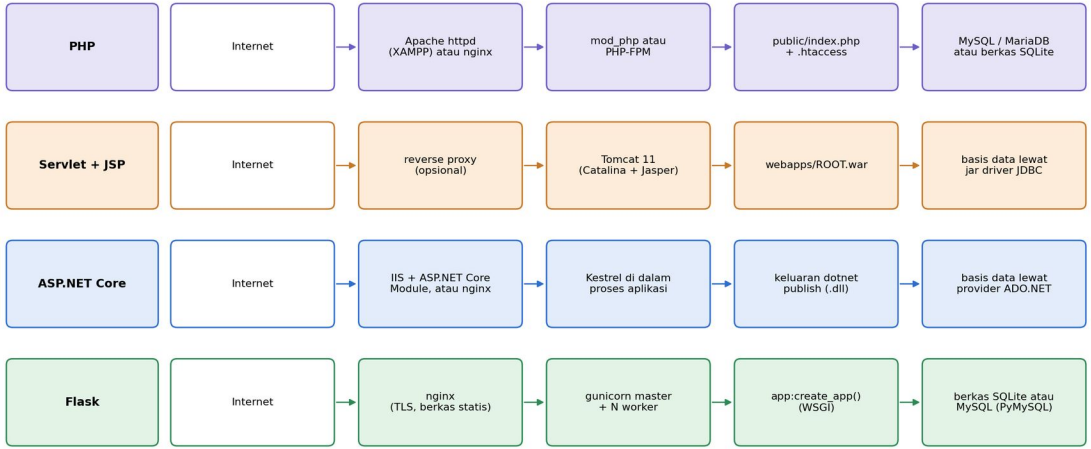
Keempat Sistem Dibandingkan

Tabel 10.8 Keempat sistem bab ini dibandingkan: bahasa, model proses, titik masuk, pemeriksaan, keadaan *login*, dan perlindungan CSRF.

	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)

Tabel 10.8: model proses menjelaskan sebagian besar baris lainnya. API basis data, konfigurasi, dan deployment dibandingkan dalam tabel lengkap di buku. Buku: Subbab 10.11.3.

Server Depan di Depan Setiap Aplikasi



Gambar 10.5 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.

Gambar 10.5: Apache, nginx, atau IIS memegang sertifikat TLS, melayani berkas statis, dan menghadapi klien yang lambat; server aplikasi menjalankan kode Anda. PHP di-deploy dengan menyalin berkas, Java sebagai WAR, ASP.NET Core sebagai folder `dotnet publish`*. Buku: Subbab 10.12–10.12.3.*

gunicorn, dan *Bug Deployment* Sungguhan

Titik awal gunicorn untuk jumlah *worker*:

$$\text{jumlah worker} = 2 \times \text{jumlah inti CPU} + 1 \tag{10.3}$$

Tanpa `LAUNDRY_SECRET_KEY`, setiap *worker* menjalankan `create_app` dan **mengarang kuncinya sendiri**: *cookie* yang ditandatangani satu *worker* adalah pemalsuan bagi *worker* lain. Pengujian gagal di enam pemeriksaan — yang berbeda pada setiap jalan.

> **PELAJARANNYA.** Apa pun yang dikarang sebuah proses saat mulai berbeda di setiap proses. Pasang satu kunci rahasia untuk setiap *worker*; instans ASP.NET Core harus berbagi *key ring* Data Protection karena alasan yang sama.

Menurut Persamaan (10.3) mesin build dua-inti akan memakai lima worker; dua sudah cukup. Di server pengembangan yang satu-proses bug ini tidak mungkin muncul, dan justru karena itulah ia sampai ke produksi. Buku: Subbab 10.12.4.

Mana yang Dipilih

Tabel 10.9 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

Tabel 10.9: untuk kantor belakang sebuah toko, keempatnya memadai. Yang menentukan adalah apa yang dikuasai orangnya, apa yang harus dihubungkan aplikasi, di mana ia berjalan, seberapa banyak framework bekerja untuk Anda, dan versi yang masih didukung. Buku: Subbab 10.13.

Profitina dalam Praktik

Pilihan mengikuti koneksinya: pemindaian, penilaian, dan model Qwen2.5 lokal hidup di Python, sehingga aplikasi webnya FastAPI di atas Uvicorn (ASGI), dengan PostgreSQL.

Server depan dan server aplikasi terpisah: Cloudflare Tunnel menghadap Internet; aplikasi berjalan sebagai layanan Windows di bawah NSSM.

Gerbang penerimaan: sebuah rilis dianggap berhasil hanya bila `/api/health` menjawab dan lebih dari seribu tes hijau; kata sandi disimpan sebagai *hash* Argon2id.

Studi kasus yang dapat dijalankan tetap Profitina Laundry; yang terbawa adalah satu kontrak, beberapa implementasi, dan satu pengujian yang mengikat setiap implementasi. Buku: Subbab 10.14.

Kesalahan yang Sering Terjadi

***Escaping* dan formulir:** lupa `e()` di satu tampilan PHP, JSP, atau *text block*; `action` harfiah pada formulir Razor Pages.

***Deployment*:** kunci rahasia acak per proses; WAR `jakarta` di Tomcat `javax`; memercayai server pengembangan.

Kode: data permintaan di *field servlet*; urutan *route* tanpa disadari; `match` alih-alih `fullmatch`; SQL yang dibangun dari teks pencarian.

Perilaku: tanpa *redirect* setelah POST; `ReturnUrl` yang tidak diperiksa; menyebut bagian *login* mana yang salah; demonstrasi alih-alih pengujian.

Setiap kesalahan tercantum di buku beserta obatnya. Buku: Subbab 10.15.

Ringkasan: Lima Hal untuk Diingat

1. Satu kontrak, lima bangunan, enam tahap yang sama: terima, periksa, *route*, putuskan, data, tampilan.

2. Simpan aturan dan pemeriksaan kata sandi di luar *framework*; iterasi membuat penebakan mahal.

3. PHP mulai dari nol; *servlet* hidup lama; ASP.NET Core dan Flask memberi *escaping* secara bawaan.

4. Sesi ada dua keluarga — id dengan data di server, atau data di *cookie* terlindungi yang memerlukan satu kunci bersama.

5. Pengujian dari luar membuktikan penolakannya juga; memilih adalah soal orang, koneksi, dan *hosting*.

Buku: Subbab 10.16.

Pertemuan Berikutnya

Bab 11 — Halaman Dinamis dan Web Form. Apa yang membuat halaman dinamis, halaman satu-berkas dan *code-behind*, dan kontrol web dalam bentuk modernnya.

Sebelum itu: kerjakan Soal Latihan 10.2, 10.3, 10.5, dan 10.6.

Penyelesaian kedua belas soal latihan ada di buku. Buku: Subbab 10.17 dan 10.18.

Kuliah 11 — Halaman Dinamis dan Web Form

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 11

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini kita memeriksa dengan saksama sesuatu yang dipakai setiap bab sebelumnya tanpa diperiksa: formulir — apa yang dikirimnya, byte demi byte, bagaimana Web Forms menyembunyikannya, dan bagaimana formulir sungguhan dibangun dengan aman. Buku: Bab 11, Subbab 11.1–11.16.

Cakupan Pertemuan Ini

1. **Halaman statis dan dinamis** — kapan isi ditentukan; penghitung pengunjung dan pembaruannya yang hilang.
2. **Tempat kode berada** — *inline*, satu berkas, dan *code-behind*; halaman adalah kelas.
3. **Formulir HTML secara mendalam** — atribut, kontrol yang *successful*, GET dan POST, jenis masukan, tiga pengodean, unggahan.
4. **Model Web Forms** — kontrol server, *postback*, ViewState, siklus hidup, *event*.
5. **Formulir Razor Pages** — *model binding*, *tag helper*, *handler*, validasi, Post/Redirect/Get, *antiforgery*.
6. **Profitina Laundry** — formulir nota baru dan unggahan foto nota, dalam empat teknologi, serta keamanan formulir.

Web Forms hanya berjalan di Windows dengan .NET Framework 4.8; listing-nya adalah gagasan yang ditandai "tidak dijalankan", masing-masing diikuti formulir Razor Pages yang dijalankan. Buku: Subbab 11.1.

Posisi Kita: Formulir

Bab 9 memperkenalkan ASP.NET dan Razor Pages; Bab 10 menjalankan PHP, JSP, dan ASP.NET berdampingan; Bab 7 membangun formulir Laundry dengan *servlet* dan JSP.

Login, mencari, menerima cucian, membayar, mengunggah foto: **semuanya formulir**.

Pemrogram yang tidak tahu persis apa yang dikirim formulir menulis aplikasi yang keliru dengan cara yang tidak disadari siapa pun — sampai seorang pelanggan, atau penyerang, menyadarinya.

Empat pertanyaan: apa yang membuat halaman dinamis; apa yang dikirim formulir; bagaimana Web Forms menyembunyikan HTTP dan apa penggantinya; bagaimana formulir sungguhan dibangun dengan aman dan diuji. Buku: Subbab 11.1.

Halaman Statis dan Halaman Dinamis

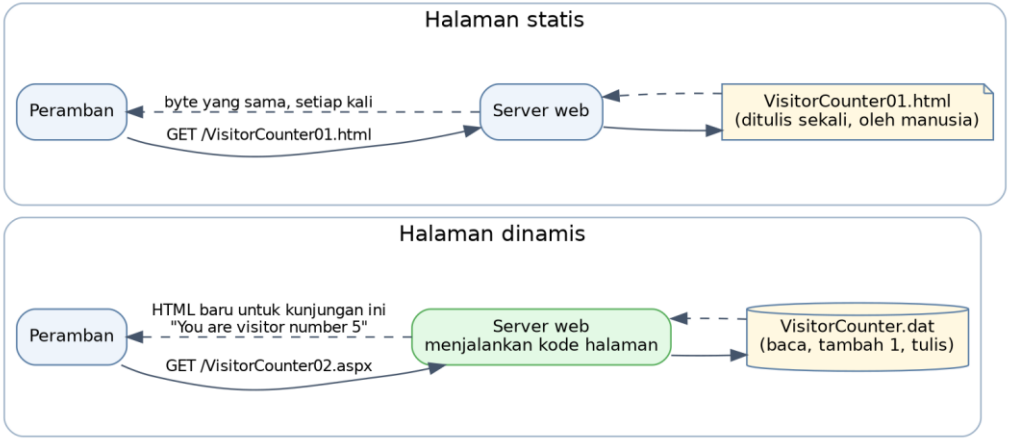
> **DEFINISI. Halaman statis** adalah berkas yang byte-nya sudah ditentukan sebelum ada yang memintanya; server mengirimkannya tanpa perubahan kepada setiap pengunjung. **Halaman dinamis** dihasilkan sebuah program pada saat diminta, sehingga dua permintaan ke alamat yang sama dapat menerima byte yang berbeda.

`VisitorCounter01.html` yang statis berkata *pengunjung nomor 1* kepada setiap pengunjung, selamanya: menghitung memerlukan ingatan yang berubah pada setiap permintaan.

Tak satu pun lebih baik: daftar harga boleh statis; apa pun yang bergantung pada siapa yang bertanya, apa yang diketik, atau isi basis data bersifat dinamis.

Setiap berkas HTML biasa bersifat statis; halaman `.aspx`, `.php`, `.jsp`, atau `.cshtml` bersifat dinamis. Buku: Subbab 11.2.1 dan 11.2.2.

Dua Jenis Halaman



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.

*Pada **Gambar 11.1** tugas server untuk halaman statis hanyalah menemukan berkas dan menyalinnya keluar; untuk halaman dinamis ia menjalankan kode yang membaca dan memperbarui berkas penghitung dan menulis HTML yang sesaat sebelumnya belum ada. Buku: Subbab 11.2.1.*

Penghitung yang Aman di Razor Pages

Kode 11.1 *Page model* dalam `Counter.cshtml.cs`: `lock` menjadikan baca, tambah, dan tulis satu langkah.

```
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());
        }
    }
}
```

> **Baca, ubah, tulis tidak pernah aman dengan sendirinya di server web**: dua permintaan dapat saling menyela — *lost update*.

***Kode 11.1**: versi Web Forms tanpa perlindungan membiarkan dua pengunjung sama-sama membaca 41 dan menulis 42. Buku: Subbab 11.2.3 dan 11.2.4.*

Lima Puluh Pengunjung Sekaligus

Kode 11.2 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
```

Setelah 4 kunjungan, 50 permintaan dikirim sepuluh sekaligus dengan `xargs -P 10 curl`, lalu satu lagi: $4 + 50 + 1 = 55$. Tidak ada kunjungan yang hilang.

`lock` melindungi satu proses server; di balik *load balancer* penghitung harus tinggal di basis data dan dinaikkan di sana, dalam satu pernyataan.

***Kode 11.2**: persis begitulah Laundry menomori notanya — di basis data, bukan di memori. Buku: Subbab 11.2.4.*

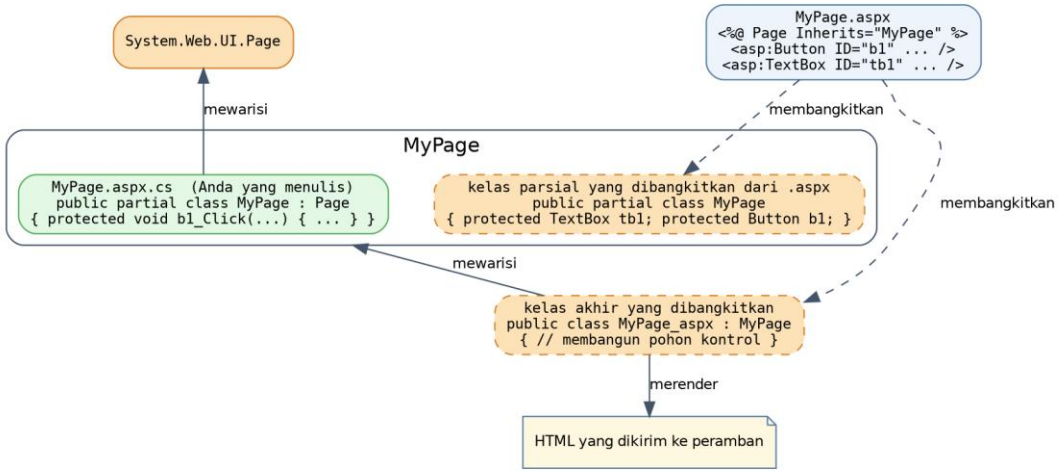
Code-Behind: Dua Berkas, Satu Kelas

Kode 11.3 Metode penghitung dalam `MyVisitor2.aspx.cs`, *code-behind* dari kelas turunan `System.Web.UI.Page`. Web Forms; tidak dijalankan di sini.

```
    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;
    }
```

***Kode 11.3**: halaman `MyVisitor2.aspx` menyatakan `Inherits="MyVisitor2"` dan memanggil `<%=MyCounter()%>`; kode C# tinggal di berkasnya sendiri, tempat kompilator, debugger, dan pengujian dapat menjangkaunya. Buku: Subbab 11.3.2.*

Halaman adalah Kelas yang Menyamar



Gambar 11.2 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.

***Gambar 11.2**: `MyPage` adalah satu kelas `partial` dalam dua bagian — milik Anda, dan satu yang dibangkitkan dengan satu field per kontrol; kelas akhir yang dibangkitkan merender halaman. Setiap permintaan dilayani objek baru yang tidak tahu apa pun tentang permintaan sebelumnya. Razor Pages mempertahankan pemisahan itu: `.cshtml` ditambah `PageModel`. Buku: Subbab 11.3.3 dan 11.3.4.*

Formulir Terkecil yang Berguna

Kode 11.4 `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>
```

***Kode 11.4**: elemen `<form>` membawa petunjuk pengiriman, kontrol membawa datanya. Diisi 10 dan 25, `/echo` menerima `total=10`, `amount=25`, dan `OK=Pay`: tombol mengirim namanya sendiri karena tombol itulah yang ditekan; Balance: dan Euro bukan data. Buku: Subbab 11.4.1.*

Atribut Elemen form

Tabel 11.1 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

***Tabel 11.1**: `formaction` milik sebuah tombol mengalahkan `action` formulir — Razor Pages memakai persis ini untuk handler bernama. Buku: Subbab 11.4.1.*

Apa yang Dikirim Formulir: Kontrol *Successful*

Tabel 11.2 Kontrol mana yang dikirim peramban, bagian pertama. Baris yang paling sering menimbulkan *bug* adalah *checkbox*: bila tidak dicentang, ia tidak mengirim *apa pun*.

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

***Tabel 11.2**: checkbox yang tidak dicentang tidak mengirim apa pun, sehingga nama yang tidak ada harus berarti mati; nama yang berulang tiba beberapa kali. Buku: Subbab 11.4.2.*

Kontrol *Successful*, Lanjutan

Tabel 11.3 Kontrol mana yang dikirim peramban, bagian kedua. Hanya tombol *submit* yang ditekan yang dikirim.

Kontrol	Dikirim bila	Dikirim sebagai
`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

***Tabel 11.3**: hanya tombol yang ditekan yang dikirim, dan dengan cara itulah satu formulir dengan dua tombol memberi tahu server tombol mana yang dimaksud; berkas ikut terkirim bersama byte-nya hanya dalam `multipart/form-data`. Buku: Subbab 11.4.2.*

GET dan POST di Jaringan

Kode 11.5 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 11.6 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
```

Kode 11.5 *dan* ***Kode 11.6*** *adalah byte sungguhan dari* `curl --trace-ascii`*. Pasangannya identik,* `customer=C0047&note=Kebaya+%26+batik`*; hanya alamatnya yang berubah — baris permintaan atau badan. Buku: Subbab 11.4.3.*

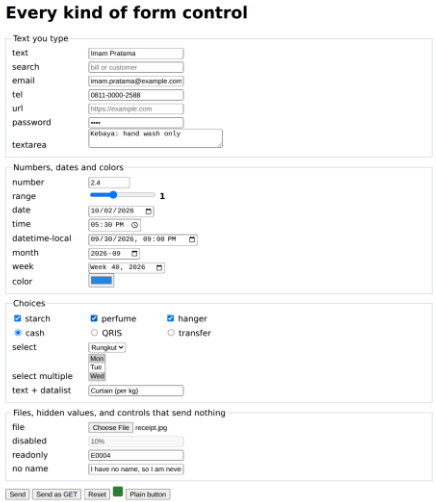
GET dan POST Dibandingkan

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

***Tabel 11.4**: jangan pernah mengirim kata sandi, atau apa pun yang mengubah data, dengan GET. URL berakhir di riwayat, bookmark, header `Referer`, dan setiap log; dikirim dengan GET, formulir uji menaruh `pin=4321` di bilah alamat. Buku: Subbab 11.4.3.*

Setiap Jenis Masukan di Satu Halaman



Gambar 11.3 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*.

***Gambar 11.3**: peramban menggambar kalender, jam, pemilih warna, dan penggeser sendiri — tetapi **yang tiba selalu teks** dalam format tetap: `2026-10-02` untuk tanggal, `2.4` dengan titik untuk angka, apa pun tampilannya. Buku: Subbab 11.4.4.*

Menyandikan Badan: urlencoded

Kode 11.7 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]
```

Setiap byte yang bukan huruf, angka, atau `-._~` menjadi `%XX`; spasi menjadi `+`. Dengan n_s byte tetap dan n_e byte di-*escape*:

$$L = n_s + 3\,n_e \tag{11.1}$$

CONTOH. `Kebaya & batik, 2½ kg` terdiri atas 22 byte (½ dihitung dua). Tanda &, koma, dan dua byte ½ di-*escape*: $n_e = 4$, $n_s = 18$, $L = 18 + 3 \times 4 = 30$.

***Kode 11.7**; menurut Persamaan (11.1) sebuah foto, yang hampir semua byte-nya "tidak aman", akan membengkak hampir tiga kali — karena itu tidak ada yang mengirim berkas dengan cara ini. Buku: Subbab 11.4.5.*

Menyandikan Badan: multipart

Kode 11.8 Badan *multipart* seperti dikirim Chromium: satu bagian per *field*, dipisahkan *boundary*; 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
```

***Kode 11.8**: alih-alih meng-escape, multipart memilih boundary yang tidak muncul di nilai mana pun dan mengumumkannya di `Content-Type`; foto 2 MB berbiaya 2 MB ditambah beberapa ratus byte. RFC 7578 mendefinisikannya. Buku: Subbab 11.4.5.*

Tiga Pengodean

Tabel 11.5 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

***Tabel 11.5**: `text/plain` menulis baris `name=value` tanpa escaping — untuk dibaca manusia di surel, bukan oleh program. Unggahan memerlukan `method="post"` **dan** `enctype="multipart/form-data"`. Buku: Subbab 11.4.5 dan 11.4.6.*

Unggahan, dan Mengapa Pemeriksaan Peramban Tidak Cukup

Bagian unggahan membawa **nama berkas**, **tipe konten**, dan **byte**. Hanya byte yang merupakan fakta; nama dan tipe dipilih pengirim.

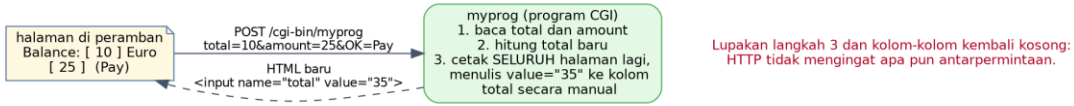
Setiap server membatasi ukuran permintaan: `upload_max_filesize` PHP, `MAX_CONTENT_LENGTH` Flask, `[RequestSizeLimit]` ASP.NET Core, `MAX_BODY` jalur Java.

`required`, `pattern`, `min`, `max` membantu pengguna yang jujur — dan dapat dibuang dalam hitungan detik dengan *developer tools*, `novalidate`, atau `curl`.

> **WAWASAN KUNCI.** Validasi peramban adalah kemudahan bagi pengguna yang jujur; validasi server adalah satu-satunya validasi. Server juga harus memeriksa apa yang tidak dapat diketahui peramban — bahwa pelanggannya ada, bahwa raknya kosong, berapa harganya.

`capture="environment"` *meminta ponsel membuka kamera belakang — yang diinginkan kasir saat memotret nota bertanda tangan. Buku: Subbab 11.4.6 dan 11.4.7.*

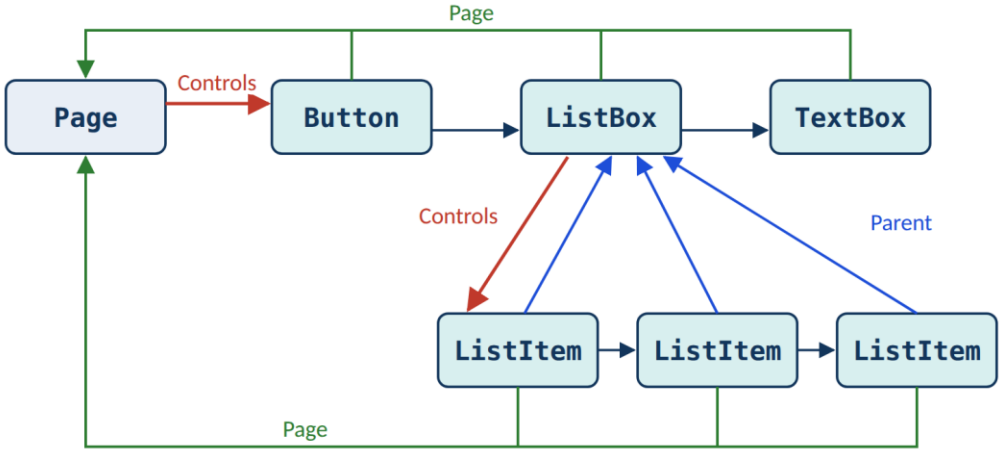
Masalah yang Hendak Dipecahkan Web Forms



Gambar 11.4 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.

***Gambar 11.4**: HTTP tidak mengingat apa pun, sehingga program CGI harus menulis ulang seluruh halaman, menyalin kembali setiap nilai yang diketik dengan escaping yang benar. Pemrogram desktop punya objek dengan event `Click`; Web Forms ingin memberikan hal yang sama kepada pemrogram web. Buku: Subbab 11.5.*

Kontrol Server dalam Sebuah Pohon



merah: Controls (turun ke anak) · biru: Parent (naik satu tingkat) · hijau: Page (ke akar) · biru tua: urutan dalam koleksi

Gambar 11.5 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`.

*`<asp:Label ID="total" Text="Hello" runat="server" />` menjadi objek kelas `Label` untuk setiap permintaan; kode menulis `total.Text = "35";` dan kontrol merender dirinya sebagai `<span>`. **Gambar 11.5**: `Controls` ke bawah pohon, `Parent` ke atas, `Page` ke akar. Buku: Subbab 11.6.1 dan 11.6.2.*

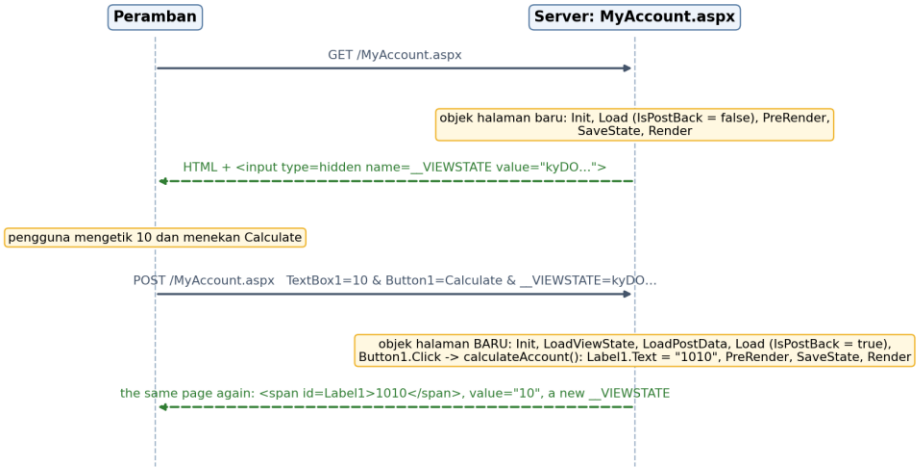
Event Handler yang Tak Pernah Menyentuh HTTP

Kode 11.9 Dari `MyAccount.aspx.cs`: dua *handler* membaca dan menulis kontrol sebagai objek. Web Forms; tidak dijalankan di sini.

```
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");
    }
```

***Kode 11.9**: ketik 10, tekan Calculate, dan label menampilkan 1010; + 10% mengubah 100 menjadi 110.00. Tidak ada yang mengurai permintaan dan tidak ada yang menyalin nilai ketikan kembali — janji yang dibuat Web Forms. Buku: Subbab 11.6.3.*

Postback



pengguna mengetik 10 dan menekan Calculate

Gambar 11.6 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 dan `true` pada setiap *postback*.

***Gambar 11.6**: kolom tersembunyi `__VIEWSTATE` dan `__EVENTTARGET` ikut dikirim. Buku: Subbab 11.6.4.*

ViewState dan Biayanya

Sebelum merender, keadaan setiap kontrol yang berubah diserialisasi, dilindungi MAC, disandikan Base64 ke `__VIEWSTATE`, dan dipulihkan pada *postback*. Untuk n byte keadaan:

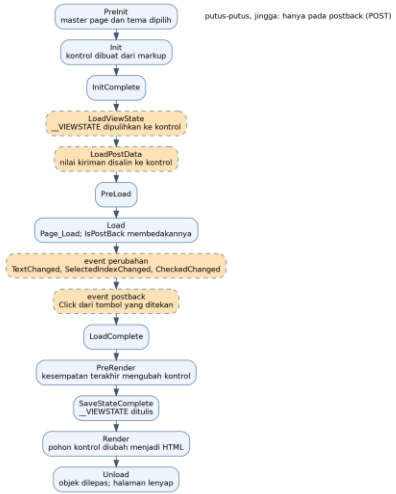
$$L_{64} = 4\lceil \frac{n}{3} \rceil \tag{11.2}$$

CONTOH. `GridView` berisi 300 nota masing-masing 100 byte: $n = 30000$, sehingga Persamaan (11.2) memberi $L_{64} = 40000$ karakter — di halaman, dan lagi di setiap klik, bahkan klik yang hanya mengurutkan daftar.

ViewState dapat dibaca. Base64 adalah pengodean, bukan enkripsi; jangan pernah mematikan MAC-nya, dan jauhkan rahasia darinya.

Label "ingat" 1010 karena ingatannya ikut pulang-pergi di dalam halaman. Buku: Subbab 11.6.5.

Siklus Hidup Halaman Web Forms



Gambar 11.7 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*.

***Gambar 11.7**: nilai kiriman dimuat sebelum `Load`; `Click` berjalan setelah `Load`. `Page_Load` yang mengisi ulang daftar di setiap permintaan, tanpa `if (!IsPostBack)`, menghapus pilihan pengguna sebelum `Click` sempat membacanya. Event perubahan diantrekan hingga postback berikutnya, kecuali `AutoPostBack="true"`. Buku: Subbab 11.6.6 dan 11.6.7.*

Gagasan Web Forms, Formulir Razor Pages

Tabel 11.6 Gagasan Web Forms dan padanannya di Razor Pages, bagian pertama: berkas, kontrol, *property*, formulir, tombol, dan kunjungan pertama.

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

***Tabel 11.6**: Web Forms tidak membuat web menjadi stateful; ia membawa keadaan di kolom tersembunyi dan membangun ulang halaman. Razor Pages mempertahankan gagasan yang berhasil dan membuang yang melawan web. Buku: Subbab 11.7.1.*

Gagasan Web Forms, Lanjutan

Tabel 11.7 Gagasan Web Forms dan padanannya di Razor Pages, bagian kedua. Perubahan terbesar ada di baris pertama: tidak ada yang disimpan di kolom tersembunyi kecuali Anda menaruhnya di sana.

Web Forms	Razor Pages	Apa yang berubah
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

***Tabel 11.7**: keadaan menjadi eksplisit dan kecil; token antiforgery melindungi permintaan, bukan keadaan. Buku: Subbab 11.7.1.*

MyAccount Dibangun Ulang: *Handler*

Kode 11.10 Tiga *handler* dalam `Account.cshtml.cs`: setiap *event* Web Forms menjadi metode bernama `OnPost` ditambah nama *handler*-nya.

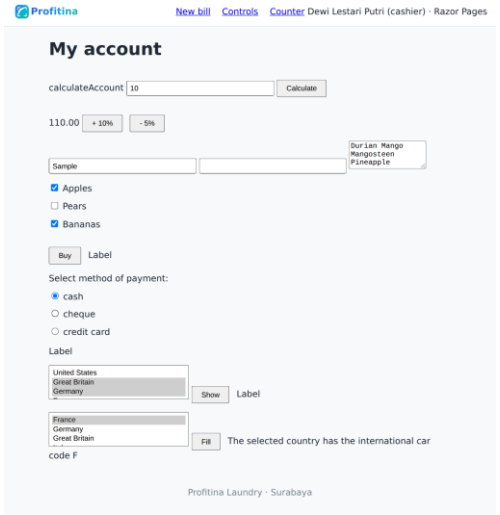
```
    public void OnPostCalculate()                                    // OnClick="calculateAccount"
    {
        int total = 1000;
        Label1 = int.TryParse(TextBox1, out int value) ? (total + value).ToString() : "Not a whole number";
    }

    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 " :
"");
    }
```

***Kode 11.10**: nilai terikat yang diubah memerlukan `ModelState.Remove`, atau nilai kiriman yang akan tampil lagi. Buku: Subbab 11.7.2.*

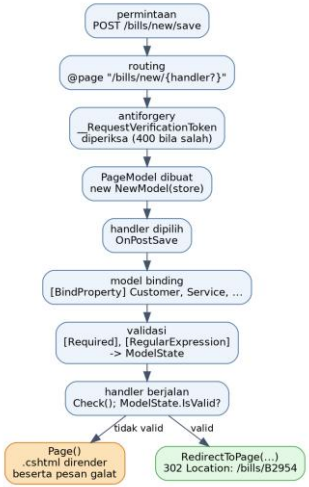
Tak Menyimpan Apa Pun Kecuali Diminta



Gambar 11.8 `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.

***Gambar 11.8**: semua yang dikirim peramban bertahan; 110.00 bertahan di kolom tersembunyi karena disengaja; label dari kiriman sebelumnya kembali menjadi `Label`. Web Forms menyimpan segalanya kecuali diminta tidak; Razor Pages tidak menyimpan apa pun kecuali diminta. Buku: Subbab 11.7.2.*

Kiriman Formulir Melintasi Razor Pages



Gambar 11.9 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.

***Gambar 11.9**: routing (`@page "/bills/new/{handler?}"`), antiforgery (400 tanpa token), page model baru dengan layanannya, pemilihan handler (`save` → `OnPostSave`), model binding dan validasi, lalu handler-nya. Buku: Subbab 11.8.1.*

Model Binding: *Property* Terikat

Kode 11.11 *Property* terikat pertama pada `NewModel`: masing-masing diisi dari *field* kiriman bernama sama; atribut setelah `BindProperty` adalah aturan validasi.

```
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; }
```

***Kode 11.11**: binding mencari di formulir, nilai route, query string, dan berkas, menurut nama tanpa memedulikan huruf besar-kecil, lalu mengonversi teks ke tipe property. Buku: Subbab 11.8.2.*

Empat Rincian *Binding*

Semuanya teks lebih dulu. `Quantity` sengaja bertipe `string`: sebagai `decimal`, `2,4` akan menjadi 2,4 di server Indonesia dan 24 di server Amerika.

Kegagalan konversi bukan *exception*. Galatnya masuk ke `ModelState`; *handler* harus memeriksa `ModelState.IsValid`.

Ikat hanya yang boleh diubah formulir. `[BindProperty] public Bill Bill` membuat siapa pun dapat menambahkan `Bill.Payment=1`: *overposting*.

GET tidak diikat secara bawaan. Halaman pencarian menyatakan `[BindProperty(SupportsGet = true)]`.

Rangkaian uji mengirim `payment=1` dan `price=1` bersama nota sungguhan; keduanya diabaikan, dan nota berharga sesuai daftar harga. Tag helper checkbox menambahkan `Express=false` tersembunyi. Buku: Subbab 11.8.2 dan 11.8.3.

Dua Tombol, Dua *Handler*

Kode 11.12 `OnPostSave`: periksa dulu, lalu gambar ulang dengan galat (400) atau simpan, taruh pesan kilat di `TempData`, dan alihkan.

```
    /// <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 });
    }
```

***Kode 11.12**: `<button asp-page-handler="Save">` menjadi `formaction="/bills/new/Save"`. `OnPostPreview` melakukan pemeriksaan yang sama dan hanya memberi penawaran harga; ia diletakkan lebih dulu di formulir, sehingga Enter tidak pernah menyimpan. Buku: Subbab 11.8.4 dan 11.9.2.*

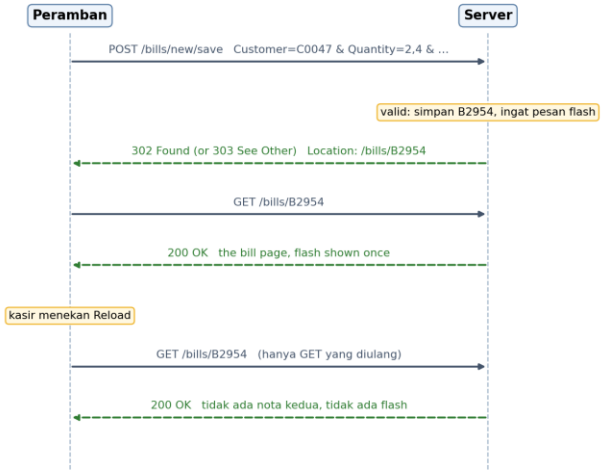
Validasi yang Memerlukan Data Toko

Kode 11.13 Kuantitas dalam `Check`: persepuluhan, rentangnya, lalu kilogram dibulatkan ke atas atau jumlah potong yang bulat.

```
        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;
            }
        }
```

***Kode 11.13**: atribut memeriksa bentuk; `ModelState.AddModelError` menambahkan yang memerlukan data — pelanggan yang tidak dikenal, ekspres pada layanan yang salah, rak yang terisi. Buku: Subbab 11.8.5.*

Post/Redirect/Get



Gambar 11.10 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.

***Gambar 11.10**: POST yang menyimpan nota dijawab dengan redirect (302 dari `RedirectToPage`, 303 di jalur lain); Reload hanya mengulang GET yang tidak berbahaya. Pesan sekali-tampil dibawa oleh `TempData`. Buku: Subbab 11.8.6.*

Token *Antiforgery*

Kode 11.14 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
```

Setiap `<form method="post">` menerima `__RequestVerificationToken` tersembunyi, berpasangan dengan sebuah *cookie*; dengan `app.UseAntiforgery()`, POST tanpa pasangan yang cocok ditolak.

Tidak ada yang perlu ditulis untuk mendapatkannya — itulah nilai bawaan sebuah *framework*.

***Kode 11.14**: 400, bahkan sebelum page model dibuat. Jalur PHP, Java, dan Python memperlihatkan apa yang dikerjakan bawaan itu untuk Anda. Buku: Subbab 11.8.7.*

Aturan Harga Laundry

Satuan, dibulatkan ke atas (Persamaan (11.3)), dan harga (Persamaan (11.4)); ekspres adalah layanan tersendiri, S0002, sehingga harganya terurai sebagai Persamaan (11.5):

$$u = \lceil w \rceil \tag{11.3}$$

$$P = p_s \cdot u \tag{11.4}$$

$$P_{\text{express}} = p_{\text{S0002}}\, u = p_{\text{S0001}}\, u + (p_{\text{S0002}} - p_{\text{S0001}})\, u \tag{11.5}$$

*Berat dibulatkan **ke atas** ke kilogram utuh; untuk potong dan pasang, u harus sudah bulat. Harga dihitung server dari daftar harganya sendiri, tidak pernah diambil dari formulir. Buku: Subbab 11.9.1.*

Biaya Tambahan, Dikerjakan

$$s = \frac{p_{\text{S0002}} - p_{\text{S0001}}}{p_{\text{S0001}}} \times 100\% = \frac{12000 - 8000}{8000} \times 100\% = 50\% \tag{11.6}$$

CONTOH. Imam Pratama (C0047) membawa 2,4 kg untuk ekspres: $u = \lceil 2.4 \rceil = 3$; biasa $8000 \times 3 = 24000$; ekspres $12000 \times 3 = 36000$; biaya tambahan 12000, 50 %. Disimpan sebagai S0002, 3 satuan, Rp 36.000. Gorden 2,1 kg (S0010): $15000 \times 3 = 45000$. 2,5 kemeja (per potong): ditolak.

Persamaan (11.6) dihitung dari daftar harga, tidak ditulis mati di program, sehingga penawaran mengikuti perubahan harga oleh pemilik. Buku: Subbab 11.9.1.

Pembulatan Tanpa *Floating Point*

Dalam *floating point* biner $0.1 + 0.2 = 0.30000000000000004$: berat yang sedikit di atas bilangan bulat akan dibulatkan satu kilogram terlalu jauh.

Maka teks diubah menjadi **persepuluhan** utuh t (`2,4` adalah $t = 24$), dan dengan pembagian bilangan bulat:

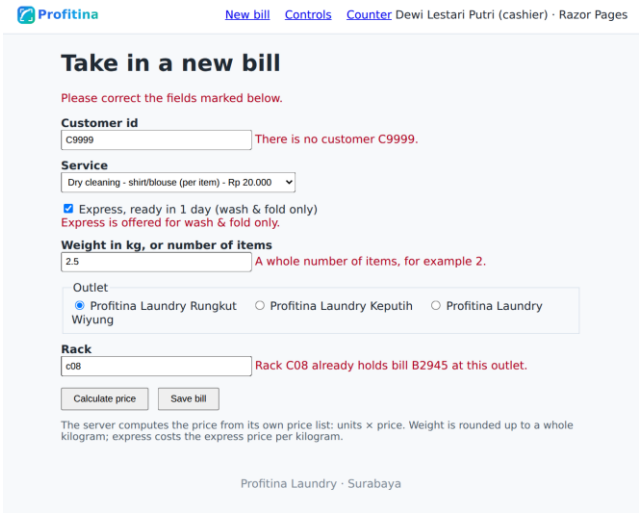
$$u = \left\lfloor \frac{t + 9}{10} \right\rfloor \tag{11.7}$$

Kode 11.15 `Tenths` dalam `BillRules.cs`: satu sampai tiga digit dan satu desimal opsional, dengan titik atau koma; selain itu `null`.

```
    /// <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);
    }
```

*Persamaan (11.7) adalah Persamaan (11.3) yang dihitung secara tepat: $t = 24$ memberi 3, $t = 30$ memberi 3, $t = 31$ memberi 4. **Kode 11.15** juga menghentikan `1e3`, `-2`, `2.45`, dan string kosong. Buku: Subbab 11.9.1.*

Jawaban Server untuk Nota yang Mustahil



Gambar 11.11 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.

***Gambar 11.11**: dengan pemeriksaan peramban dimatikan, C9999, kemeja, ekspres, 2,5, dan rak `c08` menghasilkan empat pesan, masing-masing di samping field-nya — tak satu pun dapat diperiksa peramban sendirian. Setiap nilai ketikan tetap di tempatnya. Buku: Subbab 11.9.3.*

Mengunggah Foto Nota

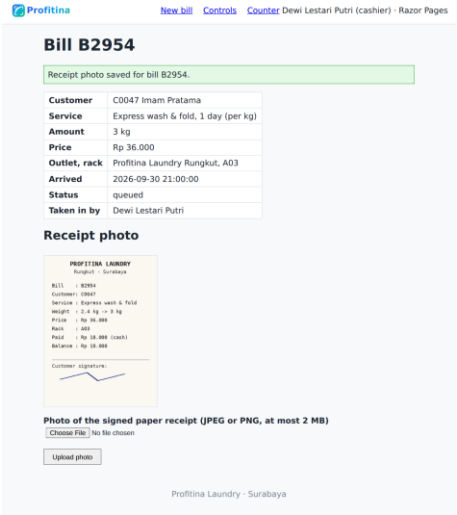
Kode 11.16 Dari `OnPostReceiptAsync` dalam `Details.cshtml.cs`: ada berkas, paling besar 2 MB, JPEG atau PNG menurut byte awalnya; lalu nama yang dibuat server.

```
        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)).ToLowerInvariant()}.{ext}";
        await using (var target = System.IO.File.Create(Path.Combine(Uploads, name)))   // our name, never theirs
            await photo!.CopyToAsync(target);
```

***Kode 11.16**: setiap JPEG diawali `FF D8 FF`, setiap PNG `89 50 4E 47 0D 0A 1A 0A`; nama dan tipe konten diabaikan. Foto disimpan di luar `wwwroot` dan dilayani satu endpoint dengan `nosniff`. Buku: Subbab 11.9.4.*

Nota Setelah Unggahan



Gambar 11.12 Nota B2954 setelah unggahan: pesan *flash* sekali-tampil, nota sebagaimana disimpan — S0002, 3 kg, Rp 36.000 — dan foto nota yang disajikan melalui `/receipts/`.

***Gambar 11.12**: pesan kilat sekali-tampil, nota seperti disimpan — S0002, 3 kg, Rp 36.000 — dan foto yang dilayani lewat `/receipts/`. Jalur Razor dijalankan dengan penyimpan CSV; penyimpan SQLite terkompilasi dengan `UseSqlite=true`. Buku: Subbab 11.9.4 dan 11.9.5.*

Formulir yang Sama di PHP: Token dengan Tangan

Kode 11.17 `csrf_token` dan `require_csrf` dalam `web.php`: 16 byte acak per sesi, dibandingkan dengan `hash_equals` sebelum *handler* POST mana pun.

```
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.');
    }
}
```

***Kode 11.17**: PHP tidak punya middleware antiforgery, jadi jalur ini menulisnya sendiri. `hash_equals` memakan waktu yang sama di mana pun string berbeda. Setiap tampilan mencetak lewat `h()`: satu `<?= $x ?>` tanpanya adalah lubang XSS. Buku: Subbab 11.10.*

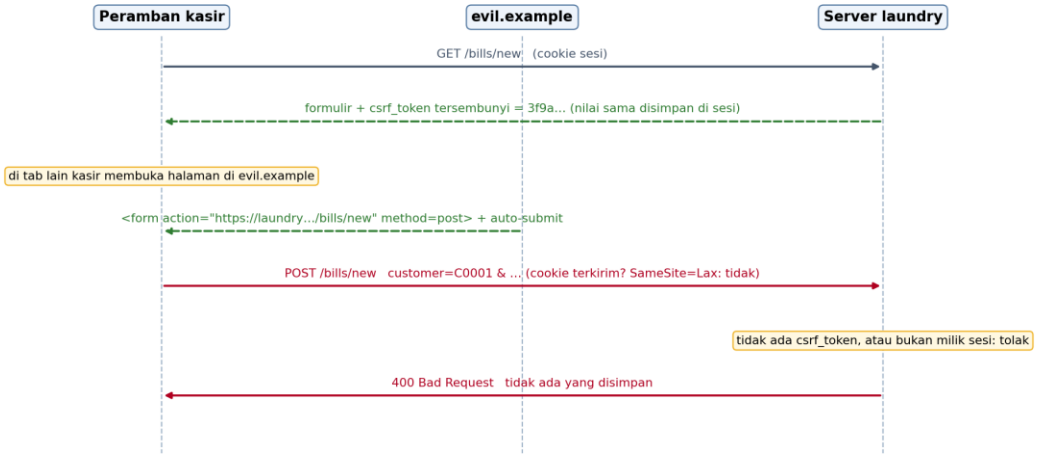
Formulir yang Sama di Flask

Kode 11.18 `post_bill` dalam `python/app.py`: periksa, lalu simpan dan alihkan dengan 303, atau beri penawaran harga, atau gambar ulang dengan 400.

```
@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)
```

***Kode 11.18**: `before_request` memeriksa token dengan `hmac.compare_digest`; `MAX_CONTENT_LENGTH` menolak lebih dari 3 MB dengan 413; Jinja2 meng-escape setiap nilai. Jalur Java mengurai badan urlencoded dan multipart dengan tangan. Buku: Subbab 11.11 dan 11.12.*

Cross-Site Request Forgery



Gambar 11.13 *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.

***Gambar 11.13**: halaman di `evil.example` membuat peramban kasir mengirim POST ke Laundry bersama cookie-nya. Token sinkronisasi menghentikannya — penyerang tidak dapat membaca maupun menebaknya — dan `SameSite=Lax` menahan cookie dari permintaan palsu itu. Buku: Subbab 11.13.3.*

Menebak Token

Token 16 byte acak berukuran 128 bit; satu tebakan berhasil dengan peluang Persamaan (11.8):

$$\Pr[\text{tebakan}] = 2^{-128} \approx 2.9 \times 10^{-39} \tag{11.8}$$

Validasi di server — bentuk, rentang, makna, konsistensi; jangan pernah menerima harga dari formulir.

***Escaping* pada keluaran** — `&`, `<`, `>`, `"`, dan `'`; `" autofocus onfocus="alert(1)` yang tidak di-*escape* membuat dapat menambah atribut.

Dengan semiliar tebakan per detik, penyerang memerlukan waktu jauh lebih lama daripada umur alam semesta. `SameSite` adalah tembok kedua, bukan pengganti. Buku: Subbab 11.13.1–11.13.3.

Unggahan dari Orang Asing

Tabel 11.8 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

***Tabel 11.8**: memeriksa byte awal bukan pemindaian malware. Foto nota bertanda tangan adalah data pribadi menurut UU No. 27 Tahun 2022: simpan dengan alasan, simpan dengan aman, jangan lebih lama dari perlu. Buku: Subbab 11.13.4.*

Dua Puluh Pemeriksaan, Empat Server

Tabel 11.9 Formulir yang sama dalam empat teknologi, bagian pertama: *routing*, penguraian, tombol yang ditekan, token, dan *escaping*.

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)

***Tabel 11.9**: setiap tugas dikerjakan di setiap jalur; yang berubah adalah seberapa banyak dikerjakan framework untuk Anda. `tests/form_suite.py` menjalankan dua puluh permintaan yang sama terhadap masing-masing: 20 lolos, 0 gagal, empat kali. Buku: Subbab 11.13.5.*

Empat Jalur, Lanjutan

Tabel 11.10 Formulir yang sama dalam empat teknologi, bagian kedua: pesan sekali-tampil, *redirect*, batas ukuran, data, dan *port*.

Tugas	Razor Pages	PHP	Java HttpServer	Flask
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

***Tabel 11.10**: Web Forms paling banyak bekerja dan paling banyak menyembunyikan; HttpServer JDK tidak mengerjakan dan tidak menyembunyikan apa pun. Pahami isi setiap baris, terutama ketika framework yang mengerjakannya. Buku: Subbab 11.13.5.*

Profitina dalam Praktik

Server yang memutuskan: biaya pemindaian dan tingkat yang mengizinkannya ditentukan dari akun, tidak pernah dari permintaan; FastAPI mengikat dan memvalidasi ke model bertipe.

Persetujuan yang bermakna: kotak yang tidak dicentang tidak mengirim apa pun, sehingga nilai yang tidak ada berarti *tidak*; penghapusan, ekspor, dan penggantian kata sandi adalah formulir yang dilindungi.

Unggahan dan teks dari orang asing: tipe menurut byte, nama dari server, keluaran di-*escape*; tautan yang diunggah disaring Link Safety.

Studi kasus yang dapat dijalankan selalu Profitina Laundry. Yang terbawa: masukan diperiksa di server, harga dihitung, keluaran di-escape, setiap perubahan keadaan membawa bukti asal. Buku: Subbab 11.14.

Kesalahan yang Sering Terjadi

Formulir: masukan berkas tanpa `multipart/form-data`; menganggap *checkbox* yang tidak ada sebagai galat; memercayai `required` dan `pattern`.

Kepercayaan: mengambil harga atau peran dari formulir; mengikat seluruh entitas; memakai nama atau tipe konten unggahan.

Alur: menjawab POST yang berhasil dengan halaman; POST tanpa token CSRF; mencetak nilai ketikan tanpa *escaping*; mengurai `2,4` dengan budaya server.

Web Forms dan Razor: mengisi ulang daftar tanpa `IsPostBack`; rahasia di ViewState; mengubah *property* terikat tanpa `ModelState.Remove`.

Setiap kesalahan tercantum di buku beserta obatnya. Buku: Subbab 11.15.

Ringkasan: Lima Hal untuk Diingat

1. Halaman statis adalah berkas, halaman dinamis adalah program; baca-ubah-tulis memerlukan perlindungan.

2. Formulir mengirim teks menurut aturan ketat: bernama, aktif, dicentang, ditekan; GET di URL, POST di badan; tiga pengodean.

3. Web Forms menyembunyikan HTTP dengan kontrol server, *postback*, dan ViewState; Razor Pages mempertahankan gagasannya tanpa menyembunyikan.

4. Server yang menghitung harga, memvalidasi, dan mengalihkan: Persamaan (11.7), Post/Redirect/Get, satu token *antiforgery* per formulir.

5. Empat kebiasaan: validasi di server, *escaping* pada keluaran, token pada setiap POST yang mengubah keadaan, unggahan diperlakukan sebagai ancaman.

Buku: Subbab 11.16.

Pertemuan Berikutnya

Bab 12 — Evaluasi dan Pembahasan III. Halaman server, formulir, dan *state* dari JSP hingga Razor Pages: soal-soal tentang Bab 9 sampai 11, dikerjakan tuntas.

Sebelum itu: kerjakan Soal Latihan 11.1 sampai 11.4 dan periksa masing-masing pada jalur yang sedang berjalan.

Penyelesaian soal latihan ada di buku. Buku: Subbab 11.17 dan 11.18.

Kuliah 12 — Evaluasi dan Pembahasan III: Halaman Server, Formulir, dan *State*, dari JSP hingga Razor Pages

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 12

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini tidak ada teknologi baru: sebelas soal, masing-masing diselesaikan dari halaman kosong dan dijalankan di mesin sungguhan, tentang gagasan Bab 9 sampai 11 dan JSP dari Bab 7. Buku: Bab 12, Subbab 12.1–12.12.

Cakupan Pertemuan Ini

1. **Sebuah metode** — enam pertanyaan untuk setiap soal tentang halaman, formulir, dan *state*, sebelum ada kode.
2. **Proyek (Soal 12.1–12.4)** — situs web JSP dengan delapan fitur CRUD: model data, fitur, rancangan yang terukur, bukti, dan penyerahan.
3. **Halaman dan *state* (12.5, 12.6)** — halaman Web Forms sebagai halaman Razor; penghitung pengunjung yang tidak kehilangan pengunjung.
4. **Formulir dan *handler* (12.7, 12.8)** — aturan yang bertahan apa pun yang dilakukan peramban; satu halaman, dua tombol.
5. **Klien kaya, lima sistem, formulir palsu (12.9–12.11)**, lalu soal-soal dibandingkan.

Setiap soal mengikuti satu urutan: soalnya, dengan kata-kata kita sendiri, rancangan, penyelesaian, Java dan Python, baris demi baris, penyelesaian keliru yang menggoda, uji diri. Buku: Subbab 12.1.

Titik Tolak Kita: Kotak Perkakas

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

***Tabel 12.1**: dua benang merah melintasi keempat bab — **halaman di server adalah program**, dan **web itu pelupa**: tidak ada yang bertahan antarpermintaan kecuali Anda sengaja menyimpannya. Buku: Subbab 12.1.*

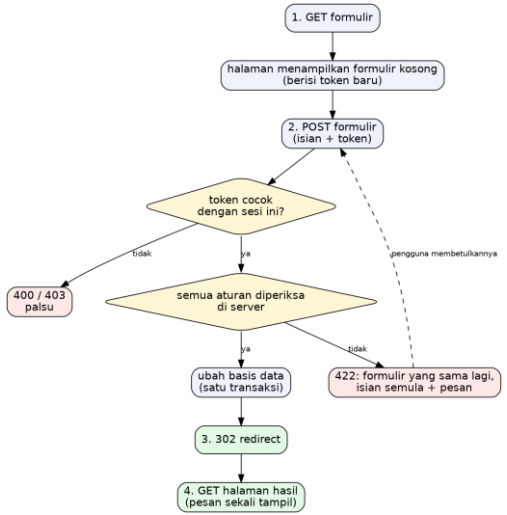
Enam Pertanyaan, Berurutan

1. **Apa saja permintaannya?** Metode, jalur, parameter; apa pun yang mengubah data adalah `POST`.
2. **Di mana setiap bagian *state* tinggal?** Basis data, sesi, aplikasi, permintaan, atau halaman itu sendiri.
3. **Siapa boleh melakukan apa?** Diperiksa di server untuk setiap permintaan; menyembunyikan tombol hanyalah sopan santun, bukan perlindungan.
4. **Apa yang harus diperiksa server?** Setiap nilai dari peramban adalah klaim, bukan fakta. **5. Bagaimana keluaran di-*escape*?** `<c:out>`, `@`, `{{ }}`, `h()`. **6. Bagaimana membuktikan bahwa ia bekerja?**

> **URUTAN ITU PENTING.** Permintaan, *state*, izin, pemeriksaan, pencetakan, bukti — baru kemudian halaman. Tim yang memulai dari halaman akan memindah-mindahkan *state* pada malam terakhir.

Pada soal kecil ini memakan lima menit; pada soal besar, inilah rancangannya. Buku: Subbab 12.2.

Riwayat Satu Formulir



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*.

***Gambar 12.1**: `GET` menampilkan formulir dengan token baru; token yang salah mengakhiri permintaan; field yang salah menampilkan formulir lagi beserta ketikannya; hanya formulir yang sah mengubah basis data, disusul redirect — Post/Redirect/Get. Buku: Subbab 12.2.*

Kasus dan Kodenya

Profitina Laundry: tiga gerai, empat belas karyawan, dua belas layanan, 256 pelanggan, 2.953 nota, satu tahun usaha yang berakhir pada 30 September 2026.

Tomcat tidak dapat diunduh, sehingga halaman JSP dijalankan di **MiniJSP**, mesin JSP kecil yang menerjemahkan setiap halaman menjadi kelas *servlet* dan melayaninya dengan server HTTP milik JDK.

Setiap server punya *port* sendiri antara 9200 dan 9299; Razor Pages membaca ekspor CSV; *listing* Web Forms hanya untuk dibaca.

`Code_new/Chapter12` *berisi* `jsp/`, `minijsp/`, `java/`, `python/`, `aspnet/LaundryPages/`, `webforms/`, `php/`, `design/`, `tests/`, *dan* `deploy/`; `VERIFIED.md` *mencatat persis apa yang dijalankan. Buku: Subbab 12.3.*

Proyeknya: Empat Soal

Tugasnya: situs web JSP dengan sedikitnya delapan fitur CRUD, dibuat seindah mungkin, dengan laporan (model data, fitur, tangkapan layar), *hosting* gratis, dan repositori publik.

12.1 model data · **12.2** delapan fitur · **12.3** “indah”, dibuat terukur · **12.4** bukti dan penyerahan.

Laundry Back Office, sisi pemilik: empat fitur untuk layanan (daftar, tambah, ubah, hapus) dan empat untuk gerai (ikhtisar, buka, ubah, tutup).

Setiap huruf CRUD muncul dua kali, dan kedua penghapusan itu persis yang kadang harus ditolak basis data. Login diperlukan tetapi tidak dihitung di antara kedelapannya. Buku: Subbab 12.4.

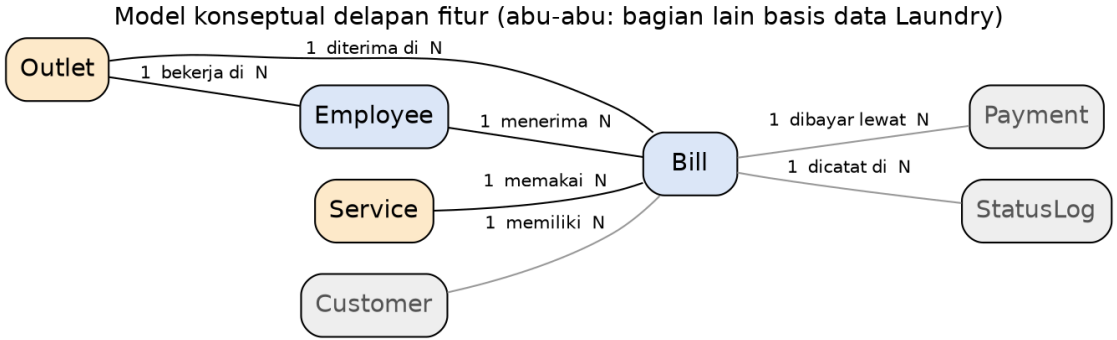
Soal 12.1: Konseptual dan Fisik

> **DEFINISI.** *Entitas* adalah sesuatu yang fakta-faktanya disimpan usaha itu; *relasi* menghubungkan dua entitas, dan *kardinalitas*-nya menyatakan berapa banyak yang satu berpasangan dengan satu yang lain (1 : N). Secara fisik entitas menjadi *tabel*, relasi 1 : N menjadi *foreign key* di sisi N.

Garis bawahi kata benda dalam fitur: kata benda dengan fakta sendiri adalah entitas (Service, Outlet, Bill, Employee); satu fakta adalah atribut; *antrean* hanyalah pertanyaan yang diajukan kepada Bill.

Model konseptual dikenali pemilik toko; model fisik adalah isi basis data yang sebenarnya — dan harus benar. Buku: Subbab 12.4.1.

Model Konseptual



Gambar 12.2 Model konseptual kedelapan fitur. Outlet dan Service adalah entitas yang diubah fitur-fitur itu; setiap garis bersifat satu-ke-banyak.

***Gambar 12.2**: Outlet dan Service berada di sisi "1" relasi dengan Bill, sehingga fitur F4 dan F8 harus mengubah penolakan basis data menjadi kalimat. Nota menyimpan apa yang ditagihkan, sehingga F3 boleh mengubah harga dengan bebas. Buku: Subbab 12.4.1.*

Model Fisik, dari Basis Data

Kode 12.1 `schema_report.py`: model fisik sebagai Markdown, dari `PRAGMA table_info` dan `PRAGMA foreign_key_list` milik SQLite sendiri.

```
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()
```

***Kode 12.1**: laporan tidak mungkin berbeda dari tabelnya. Keliru yang menggoda: tabel gambar tangan, atau foreign key yang tidak ditegakkan. Buku: Subbab 12.4.1.*

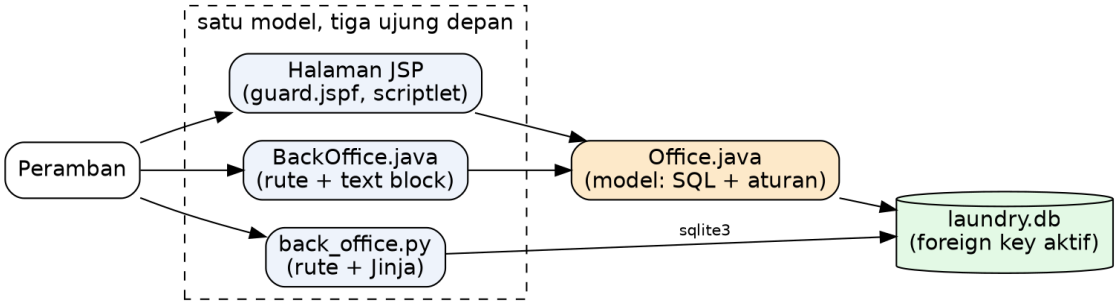
Soal 12.2: Delapan Fitur CRUD

Tabel 12.2 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

***Tabel 12.2**: layanan dan gerai tinggal di basis data; siapa yang login dan token formulir di sesi; baris yang dicetak halaman di permintaan; pesan sekali-tampil di sesi, melintasi redirect. Buku: Subbab 12.4.2.*

Satu Model, Tiga *Front End*



Gambar 12.3 Satu model, tiga ujung depan. Halaman JSP dan *server* JDK berbagi `Office.java`; versi Flask mencerminkannya dengan `sqlite3`.

***Gambar 12.3**: setiap pernyataan SQL dan setiap aturan tinggal di satu kelas Java biasa, `Office`, yang tidak tahu apa pun tentang HTML; halaman hanya memuat penjaga, satu panggilan, dan HTML yang mencetak hasilnya. Buku: Subbab 12.4.2.*

Penjaga, Ditempelkan di Setiap Halaman

Kode 12.2 `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 12.2**: disertakan dengan `<%@ include file="…" %>` dari `WEB-INF`, yang tidak dapat diminta peramban. `return;` langsung mengakhiri metode layanan halaman — sah karena halaman adalah metode. Token dibandingkan dalam waktu konstan. Buku: Subbab 12.4.2.*

F1: Daftar Harga

Kode 12.3 Baris-baris `services.jsp` (F1): setiap nilai lewat `<c:out>`, dan Delete sebagai formulir POST yang membawa token.

```
      <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>
```

***Kode 12.3**: scriptlet halaman hanya satu baris — minta model akan layanan yang cocok dengan `q` dan taruh di request scope; teks pencarian mencapai SQL hanya sebagai nilai untuk placeholder `?`. Buku: Subbab 12.4.2.*

F4: Hapus, Hanya dengan POST

Kode 12.4 `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");
%>
```

> **Penyelesaian keliru yang menggoda: tautan hapus.** `<a href="service-delete.jsp?id=S0005">` juga "diklik" oleh peramban yang melakukan *prefetch*, pratinjau aplikasi obrolan, atau `<img src=…>` di situs lain. `GET` harus *aman*.

Kode 12.4 *menolak* `GET` *dengan 405, membiarkan foreign key memutuskan, dan melaporkan putusannya setelah redirect. Buku: Subbab 12.4.2.*

Soal 12.3: Keindahan yang Dapat Diperiksa

1. Sedikit warna, masing-masing *custom property* bernama (`--brand: #0b5cad`). **2.** Setiap warna teks terbaca: kontras sedikitnya 4,5 : 1 (3 : 1 untuk teks besar).

3. Satu langkah jarak, `--step`. **4.** Satu skala huruf, angka tabular untuk rupiah.

5. Fokus yang terlihat dengan `:focus-visible`. **6.** Tidak ada yang rusak di ponsel: di bawah 40rem setiap baris tabel menjadi kartu.

Keindahan tidak dapat dibuktikan, tetapi sebagian besar hal yang membuat situs tampak terawat dapat — dan sisanya dapat dibuat konsisten. Satu stylesheet melayani semua versi, byte demi byte. Buku: Subbab 12.4.3.

Kontras, dari Definisinya

Setiap kanal $c_s = c/255$ dibuat linear dengan Persamaan (12.1); luminans relatif memberi bobot pada kanal linear dengan Persamaan (12.2); rasio yang lebih terang L_1 terhadap yang lebih gelap L_2 adalah Persamaan (12.3):

$$c_\ell = \begin{cases} c_s/12.92 & \text{jika } c_s \le 0.04045 \\ \left(\frac{c_s+0.055}{1.055}\right)^{2.4} & \text{selainnya.} \end{cases} \tag{12.1}$$

$$L = 0.2126\,R_\ell + 0.7152\,G_\ell + 0.0722\,B_\ell \tag{12.2}$$

$$\text{rasio} = \frac{L_1 + 0.05}{L_2 + 0.05}, \tag{12.3}$$

Mata paling peka terhadap hijau dan paling kurang terhadap biru; 0.05 *mewakili cahaya yang dipantulkan layar sungguhan bahkan saat menampilkan hitam. Rasionya dari 1 sampai 21. Buku: Subbab 12.4.3.*

Teks Sekunder, Dihitung dengan Tangan

CONTOH. `--muted` adalah `#52606d`: 82, 96, 109, diskalakan menjadi 0.3216, 0.3765, 0.4275. Linear: 0.0844, 0.1170, 0.1529. $L = 0.2126 \times 0.0844 + 0.7152 \times 0.1170 + 0.0722 \times 0.1529 = 0.1126$. Terhadap putih: $(1 + 0.05)/(0.1126 + 0.05) = 6.46$, di atas 4,5.

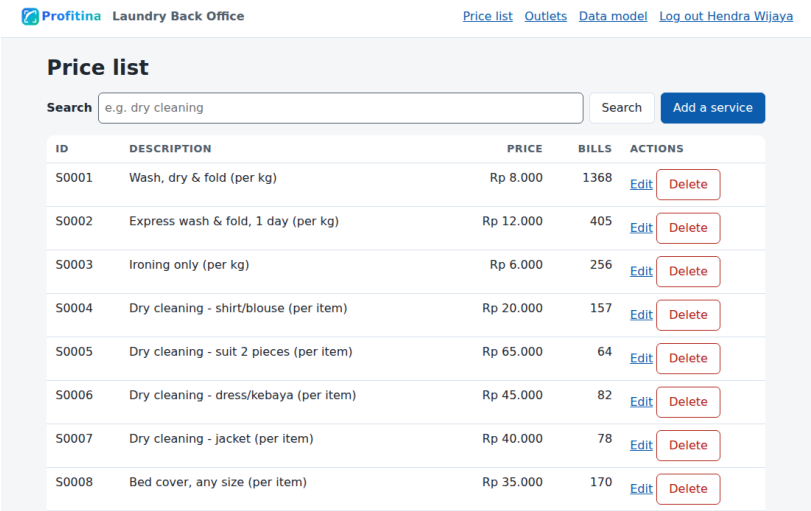
Kode 12.5 Dua fungsi `contrast.py`: luminans menurut Persamaan (12.1) dan Persamaan (12.2), dan rasionya menurut Persamaan (12.3).

```
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)
```

Kode 12.5 *mencetak setiap pasangan dalam stylesheet; abu-abu lembut yang menggoda,* `#9aa5b1` *di atas putih, hanya 2,50 : 1 — gagal bahkan untuk teks besar. Buku: Subbab 12.4.3.*

F1 Seperti Dilayani



Gambar 12.4 Fitur F1, daftar harga, disajikan halaman JSP di atas MiniJSP. Angka rata kanan; Delete berwarna merah dan meminta konfirmasi.

Gambar 12.4 *diambil oleh program, di Chromium headless, dari situs yang berjalan: angka rata kanan, Delete berwarna merah dan meminta konfirmasi. Buku: Subbab 12.4.3.*

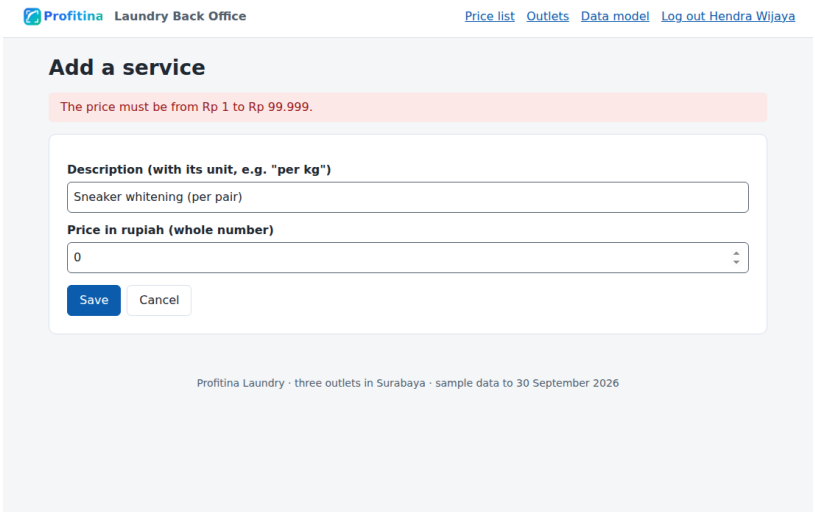
Soal 12.4: Uji Penerimaan

```
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
```

Uji ini berbicara HTTP, menyimpan *cookie*, tidak pernah mengikuti *redirect*, dan menemukan jalannya lewat beberapa tanda yang dicetak ketiga versi; putarannya berakhir *21 of 21 checks passed*.

Baris-baris pertama putaran terhadap halaman JSP, di atas salinan basis data yang baru. Tangkapan layar menunjukkan satu hari; ini membuktikannya lagi dalam satu detik setelah setiap perubahan. Buku: Subbab 12.4.4.

Penolakan, Dipotret oleh Program



Gambar 12.5 Fitur F2 menolak harga 0: status 422, pesan di atas, dan apa yang sudah diketik tetap berada di isian.

***Gambar 12.5**: F2 menolak harga 0 dengan status 422, pesan di atas, dan ketikan yang tetap di field-nya. `tests/screenshots.py` menggerakkan Chromium dengan Playwright. Buku: Subbab 12.4.4.*

Laporan, Disusun dari Keluaran Program

Tabel 12.3 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 12.2**
Bukti	satu tangkapan layar dan dua atau tiga kalimat per fitur; putaran uji penerimaan	`screenshots.py`; keluaran `acceptance.py`
Rancangan tampilan	keenam aturan; tabel kontras	Soal 12.3; keluaran `contrast.py`
Cara menjalankan dan memasang	perintah, pengaturan, alamat situs yang hidup	`README.md`; `build-war.sh`, `Dockerfile`
Tim	keluaran dan bagian setiap anggota	tabel seperti yang diuraikan di bawah

***Tabel 12.3**: tidak ada yang diketik dari ingatan bila program dapat mencetaknya. Untuk tim, sebutkan keluaran konkret dan bagian setiap anggota; jumlah commit bukan ukuran kerja. Buku: Subbab 12.4.4.*

Pengemasan, *Hosting*, Repositori

WAR untuk Tomcat 10.1 atau lebih baru: halaman di tingkat teratas, kelas di `WEB-INF/classes`, pustaka di `WEB-INF/lib`; Tomcat mengompilasi halaman JSP dan memerlukan dua *jar* JSTL.

***Hosting* gratis** yang menjalankan *container*: kebanyakan *hosting* gratis hanya menjalankan PHP. Disk *container* biasanya dihapus pada setiap *redeploy*, sehingga data sungguhan tempatnya di basis data ter-*hosting*.

Repositori: kode, bukan data atau rahasia — `.gitignore` menjauhkan `laundry.db`, `.env`, dan kunci; rahasia yang pernah ter-*push* harus diganti.

Pembangunan WAR dijalankan; Tomcat dan Docker tidak tersedia, sehingga langkah pemasangannya adalah templat. Meng-commit `laundry.db` akan menerbitkan nama dan telepon setiap pelanggan. Buku: Subbab 12.4.4.

Soal 12.5: Web Forms ke Razor Pages

Tabel 12.4 Setiap bagian halaman Web Forms dan padanannya di Razor Pages: halaman, kode, kunjungan pertama, tombol, kotak teks, dan daftar.

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="…">`

***Tabel 12.4**: Razor Pages mempertahankan pasangan berkas dan satu handler per permintaan, dan membuang ViewState. Buku: Subbab 12.5.1.*

Web Forms ke Razor Pages, Lanjutan

Tabel 12.5 Setiap bagian halaman Web Forms dan padanannya di Razor Pages: validator, label, ViewState, dan blok *render*.

Web Forms (`.aspx`)	Razor Pages (`.cshtml` + `.cshtml.cs`)
`<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)

***Tabel 12.5**: yang hilang adalah `__VIEWSTATE`, dan bersamanya ilusi bahwa halaman itu mengingat. Buku: Subbab 12.5.1.*

Penawaran Harga, Dihitung di Server

Kode 12.6 `OnPost` dalam `Quote.cshtml.cs`: cari layanannya, berhenti dengan 422 bila ada yang tidak sah, kalikan di server.

```
    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)}";
    }
```

***Kode 12.6**: `Units` adalah `[BindProperty]` dengan `[Range(1, 100)]`, sehingga `abc` gagal di model binding sebelum aturan mana pun berjalan; `Store` datang lewat injeksi dependensi. Buku: Subbab 12.5.1.*

Cara yang Menggoda: Total dari Peramban

```
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 dua potong seharga satu rupiah. *Field* tersembunyi adalah bagian formulir, dan formulir adalah apa pun yang dibuat pengirimnya.

Halaman `/wrong` di jalur Flask memercayai total yang dihitung JavaScript di peramban. Harga yang datang dari peramban adalah harga yang dipilih pelanggan. Buku: Subbab 12.5.1.

Soal 12.6: Penghitung dengan *Lock*

Kode 12.7 Dari `CounterModel`: `OnGet` menaikkan hitungan di dalam `lock (Gate)`; `OnGetNaive` tidak.

```
    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;
    }
```

***Kode 12.7**: `Gate` harus berupa `static readonly object` — ASP.NET Core membuat page model baru per permintaan, sehingga field biasa tidak mengunci apa pun. Buku: Subbab 12.5.2.*

Empat Ratus Pengunjung, Enam Belas Sekaligus

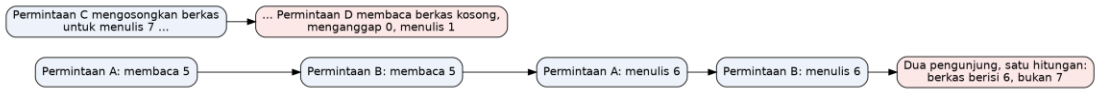
Pengukurannya menghitung apa yang hilang dari penghitung:

$$\text{hilang} = \text{pengunjung dikirim} - \text{nomor tertinggi yang tampil} \tag{12.4}$$

```
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
```

Menurut Persamaan (12.4), penghitung naif kehilangan 322 dari 400 di halaman Razor (287 di Java, 339 di Python); yang aman menghitung tepat 400. Angka naif berubah dari satu putaran ke putaran lain; angka yang aman tidak pernah. Buku: Subbab 12.5.2.

Mengapa Begitu Banyak yang Hilang



Gambar 12.6 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.

***Gambar 12.6**: lost update klasik hanya menghilangkan segelintir; ratusan datang dari membaca berkas ketika permintaan lain sedang mengosongkannya untuk ditulis ulang, lalu menganggap "kosong" sebagai nol. Di dua server hanya basis data yang dapat memutuskan:* `UPDATE Counter SET n = n + 1`*. Buku: Subbab 12.5.2.*

Soal 12.7: Aturan yang Bertahan Apa Pun yang Dilakukan Peramban

Kasir menerima nota: id pelanggan seperti `C0047` yang ada, sebuah layanan, kuantitas 1 sampai 100, rak seperti `A07`, sebuah gerai. Setiap kesalahan di samping *field*-nya; ketikan tetap ada.

Aturan **bentuk** dinyatakan sebagai atribut; aturan **keberadaan** memerlukan data dan ditulis di *handler*. Tagihan dihitung di server:

$$\text{tagihan} = \text{harga layanan} \times \text{jumlah}, \tag{12.5}$$

Persamaan (12.5) dihitung dari daftar harga saat nota diterima dan disimpan di nota, sehingga perubahan harga kelak tidak menulis ulang nota itu. Buku: Subbab 12.6.1.

Aturan Bentuk sebagai Atribut

Kode 12.8 Kelas masukan dalam `Bills/New.cshtml.cs`: aturan bentuk dinyatakan sebagai atribut, diperiksa selama *model binding*.

```
    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";
    }
```

***Kode 12.8**: field-nya bernama `Input.CustomerId` dan seterusnya; setiap kegagalan dicatat di `ModelState` di bawah namanya. Padanan Web Forms-nya adalah kontrol validator di samping setiap kotak teks. Buku: Subbab 12.6.1.*

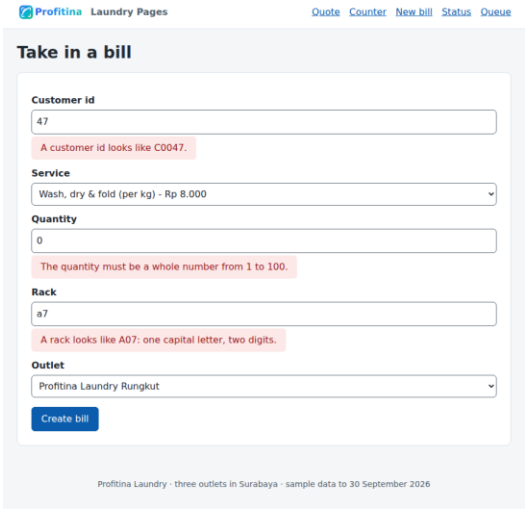
Aturan Keberadaan dalam Kode, Tagihan di Server

Kode 12.9 `OnPost` dalam `Bills/New.cshtml.cs`: aturan keberadaan ditambahkan ke `ModelState` yang sama, tagihan dihitung *store*, lalu *redirect*.

```
        // 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
```

***Kode 12.9**: pemeriksaan keberadaan hanya berjalan bila bentuknya sah; setelah berhasil, `TempData` membawa pesannya melintasi redirect ke halaman nota. Buku: Subbab 12.6.1.*

Tiga Kesalahan, Satu Jawaban



Gambar 12.7 Formulir Razor setelah `47`, `0`, dan `a7`: tiga pesan, status 422, tidak ada yang ditulis.

***Gambar 12.7**: `47`, `0`, dan `a7` menghasilkan tiga pesan dan status 422; tidak ada yang ditulis. Permintaan itu dikirim program yang tidak pernah menjalankan HTML halaman — `required` dan `pattern` peramban tidak pernah melihatnya. Buku: Subbab 12.6.1.*

Soal 12.8: Satu Halaman, Dua Tombol

Kode 12.10 `OnPostAdvance` dan `OnPostPay` dalam `Bills/Detail.cshtml.cs`: masing-masing meminta *store*, meninggalkan pesan di `TempData`, lalu mengalihkan.

```
    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
    }

    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 });
    }
```

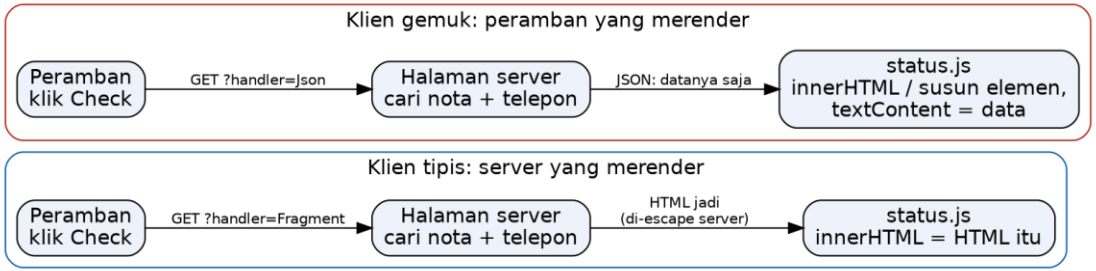
***Kode 12.10**: `asp-page-handler="Advance"` menulis `?handler=Advance`; aturannya (nota yang sudah diambil tidak dipindah, pembayaran tidak melebihi sisa) ada di store. Menjawab POST dengan halaman akan membuat Reload mencatat pembayaran dua kali. Buku: Subbab 12.6.2.*

Kasir, Diperankan oleh Skrip

```
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
```

Putaran `bills_check.py` terhadap halaman Razor; versi Java dan Python mencetak baris yang persis sama. Setiap POST pada baris 4 sampai 8 menjawab 302, dan permintaan palsu 400. Buku: Subbab 12.6.1 dan 12.6.2.

Soal 12.9: *Thin Client* dan *Fat Client*



Gambar 12.8 *Thin client* dan *fat client*. Pertanyaan dan pencariannya sama; yang berbeda adalah apakah *server* atau peramban yang mengubah data menjadi HTML.

***Gambar 12.8**: pertanyaan dan pencariannya sama; yang berbeda adalah apakah server atau peramban yang mengubah data menjadi HTML. Nota hanya ditampilkan kepada orang yang juga tahu nomor teleponnya. Buku: Subbab 12.7.1.*

Cara yang Menggoda: innerHTML

Kode 12.11 Mode yang menggoda dalam `status.js`: JSON ditempelkan ke dalam string HTML dan diberikan ke `innerHTML`.

```
// 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>`;
},
```

Mode *thin* menempelkan fragmen dari server, yang sudah di-*escape* Razor: `answer.innerHTML = await (await fetch(query("Fragment"))).text();`.

***Kode 12.11**: template string tampak seperti templat Razor tetapi tidak meng-escape apa pun; nama yang berisi markup diurai dan dijalankan. Buku: Subbab 12.7.1.*

Cara yang Benar: textContent

Kode 12.12 Mode *fat* yang aman dalam `status.js`: elemen dibangun dengan `createElement`, data dimasukkan dengan `textContent`.

```
// 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);
},
```

***Kode 12.12**: `textContent` tidak pernah diurai sebagai HTML, sehingga tidak ada nilai yang dapat menjadi markup. `status.js` yang sama, byte demi byte, melayani versi Razor, Java, dan Flask. Buku: Subbab 12.7.1.*

Apa yang Berpindah, dan Apa yang Berjalan

```
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
```

Putaran `status_demo.py`: 165 byte HTML lawan 132 byte JSON; untuk pelanggan yang "nama"-nya berupa `<img onerror=…>`, hanya mode `innerHTML` yang menjalankan markup-nya. Buku: Subbab 12.7.1.

Cross-Site Scripting, Tertangkap Basah



Gambar 12.9 *Fat client* yang tidak aman setelah satu klik: “nama” pelanggan kini menjadi elemen halaman, dan skripnya telah berjalan.

***Gambar 12.9**: nama itu menjadi elemen halaman dan skripnya berjalan — dalam serangan sungguhan, membaca halaman atau mengirim permintaan dengan sesi pelanggan. JSON dari server tidak bersalah; satu baris skriplah yang bersalah. Buku: Subbab 12.7.1.*

Soal 12.10: Halaman yang Sama di Lima Sistem

Kode 12.13 *Query* dalam `queue.php`: setiap gerai muncul, bahkan tanpa antrean — `LEFT JOIN`, syarat di `ON`, `COUNT(b.id)`.

```
$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]);
```

***Kode 12.13**: PHP, JSP, Razor Pages (`GroupJoin` LINQ), Java, dan Flask mencetak tabel yang sama: dua nota setrika menunggu di Rungkut, tidak ada di dua gerai lainnya. Buku: Subbab 12.7.2.*

Tiga *Query* yang Menggoda dan Satu yang Benar

```
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
```

A. `JOIN` biasa hanya menyimpan gerai yang punya pasangan: dua lenyap. **B.** Syarat di `WHERE` berjalan setelah *join* dan membatalkan `LEFT JOIN`.

C. `COUNT(*)` menghitung baris berisi *null*: Keputih dan Wiyung mendapat 1, bukan 0 — jawaban paling berbahaya, karena tampak masuk akal.

Mulailah dari tabel yang semua barisnya harus muncul, taruh syaratnya di `ON`, dan hitung satu kolom tabel yang di-join. Buku: Subbab 12.7.2.

Soal 12.11: Formulir yang Tidak Dapat Dipalsukan

Razor Pages menambahkan `__RequestVerificationToken` ke setiap formulir POST dan memeriksanya sebelum *handler* mana pun: selain itu 400.

Token sesi (Back Office, Flask): nilai acak di sesi, dicetak ke setiap formulir, dibandingkan pada setiap POST.

Double-submit cookie (halaman nota Java): nilai di *cookie*-nya sendiri dan di formulir; server tidak menyimpan apa pun dan membandingkan keduanya.

```
POST with no token                  -> 400
POST with the attacker's own token  -> 400
POST with the victim's own token    -> 302 /Bills/Detail/B2955
```

Dikirim dengan cookie korban: tanpa token dan dengan token milik penyerang ditolak; hanya formulir asli yang lolos. Hanya memeriksa `POST`, memeriksa `Referer`, satu token untuk semua, atau `[IgnoreAntiforgeryToken]` — semuanya menggoda dan keliru. Buku: Subbab 12.8.1.

Soal-Soal Dibandingkan

Tabel 12.6 Soal-soal bab ini dibandingkan, bagian pertama: proyek dan halaman Web Forms, beserta kasus yang digagalkan penyelesaian yang menggoda.

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

***Tabel 12.6**: bacalah kolom terakhir — setiap penyelesaian yang menggoda bekerja pada contoh biasa dan gagal dalam satu situasi tertentu. Buku: Subbab 12.9.*

Soal-Soal Dibandingkan, Lanjutan

Tabel 12.7 Soal-soal bab ini dibandingkan, bagian kedua: *state*, formulir, *handler*, klien, lima sistem, dan formulir palsu.

Soal	Pokok	Gagasan utama	Kekeliruan penyelesaian yang menggoda
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

***Tabel 12.7**: dua pengunjung sekaligus, markup di dalam nama, gerai tanpa antrean, permintaan yang tidak pernah dimulai pengguna — tak satu pun adalah contoh dalam soal. Tuliskan kasus-kasus yang janggal sebelum kodenya. Buku: Subbab 12.9.*

Profitina dalam Praktik

***Fat client* dengan kewajiban:** satu `GET /api/me/dashboard` mengembalikan JSON, dan setiap teks yang ditulis pengguna lain dimasukkan ke halaman sebagai teks, tidak pernah sebagai *markup*.

Formulir optimistis, tetap diperiksa di server; tautan melewati penyaringan keamanan di server.

Batas yang bertahan saat permintaan datang bersamaan — kanal per hari, percobaan *login* — diputuskan di basis data, dalam satu transaksi.

Tidak ada rilis sebelum lebih dari seribu tes hijau dan versi baru menjawab pemeriksaan kesehatannya. Versi yang dapat dijalankan dari setiap gagasan adalah kode Profitina Laundry. Buku: Subbab 12.10.

Kesalahan yang Sering Terjadi

Rancangan: menulis halaman lebih dulu; SQL dan aturan di *scriptlet* JSP; data dan rahasia di repositori.

Permintaan: mengubah data dengan `GET`; menjawab `POST` dengan halaman; melemahkan token anti-pemalsuan.

Pemeriksaan dan *state*: memercayai pemeriksaan peramban; lupa `ModelState.IsValid`; baca-tambah-tulis tanpa *lock*; *foreign key* yang tidak ditegakkan.

Keluaran dan SQL: data di `innerHTML`; `JOIN`, `WHERE`, atau `COUNT(*)` padahal yang dimaksud hitungan `LEFT JOIN`; teks berkontras rendah; bukti yang tidak dapat diulang.

Setiap kesalahan tercantum di buku bersama soal yang memperlihatkannya. Buku: Subbab 12.11.

Ringkasan: Lima Hal untuk Diingat

1. Enam pertanyaan sebelum halaman: permintaan, *state*, izin, pemeriksaan, *escaping*, bukti.

2. Satu kelas model di bawah halaman yang tipis; model fisik dicetak basis data; rancangan yang kontrasnya dihitung.

3. Uji penerimaan lewat HTTP membuktikan setiap hari apa yang dibuktikan tangkapan layar sekali saja.

4. *State* bersama memerlukan satu *lock* atau basis data; aturan dan harga di server; setiap POST diakhiri *redirect*.

5. `textContent`, bukan `innerHTML`; `LEFT JOIN` dengan `COUNT(b.id)`; satu token anti-pemalsuan per peramban pada setiap POST.

Buku: Subbab 12.12.

Pertemuan Berikutnya

Bab 13 — Data di Halaman: GridView dan Tabel Data. Tabel data, dari GridView Web Forms hingga bentuk modernnya.

Sebelum itu: kerjakan Soal Latihan 12.1, 12.3, 12.5, dan 12.9.

Penyelesaian kesepuluh soal latihan ada di buku. Buku: Subbab 12.13 dan 12.14.

Kuliah 13 — Data di Halaman: GridView dan Tabel Data

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 13

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini halaman web berhenti menampilkan satu rekaman dan mulai menampilkan semua rekaman: 2.953 nota yang dibagi per halaman, diurutkan, disaring, diubah, dihapus, dan diekspor. Buku: Bab 13, Subbab 13.1–13.25.

Cakupan Pertemuan Ini

1. **Masalahnya** — tujuh pekerjaan sebuah tabel data, dan data di baliknya: MyLaundry dan Profitina Laundry.
2. **GridView** — kolom, perintah, *postback*, templat, kontrol sumber data, dan *bug* yang layak dipelajari.
3. ***Grid* modern** — *state* di alamat, saringan yang diikat, daftar putih untuk pengurutan, `LIMIT ? OFFSET ?`.
4. **Kecepatan** — mengapa halaman 119 lebih lambat daripada halaman 1, *keyset paging*, dan mengukur keduanya.
5. **Mengubah data** — pengubahan status di tempat dengan konkurensi optimistis, penghapusan yang dikonfirmasi, ekspor CSV yang aman.
6. **Empat jalur dan satu skrip** — Razor Pages, PHP, Java, Flask, *progressive enhancement* dengan `fetch()`, satu himpunan pemeriksaan.

Urutannya mengikuti bab: kontrol klasik lebih dulu, karena grid modern paling mudah dipahami sebagai jawaban atas keterbatasannya. Buku: Subbab 13.1.

Posisi Kita Saat Ini

Bab 9 memperkenalkan ASP.NET: Web Forms (2002, hanya Windows dan .NET Framework) dan ASP.NET Core dengan Razor Pages, yang dijalankan buku ini. Bab 10 membandingkan PHP, JSP, dan ASP.NET; Bab 11 membahas formulir; Bab 12 mengerjakan soal lintas ketiganya.

Sebuah formulir memasukkan **satu** rekaman ke basis data. Pemilik Profitina Laundry ingin melihat **nota-nota**: yang minggu ini, yang menunggu di Keputih, yang termahal, yang salah ketik tadi pagi.

> **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.

Setiap kendali grid adalah tautan atau formulir kecil, dan masing-masing berakhir sebagai permintaan HTTP biasa. Di grid-lah kecerobohan terhadap banyak data menjadi mahal. Buku: Subbab 13.1.

Tujuh Pekerjaan

Tabel 13.1 Tujuh pekerjaan sebuah tabel data, dalam Web Forms dan dalam *grid* bab ini.

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

Ketujuh pekerjaan Tabel 13.1 dikerjakan di satu halaman. Buku: Subbab 13.2.1.

Besarnya Masalah

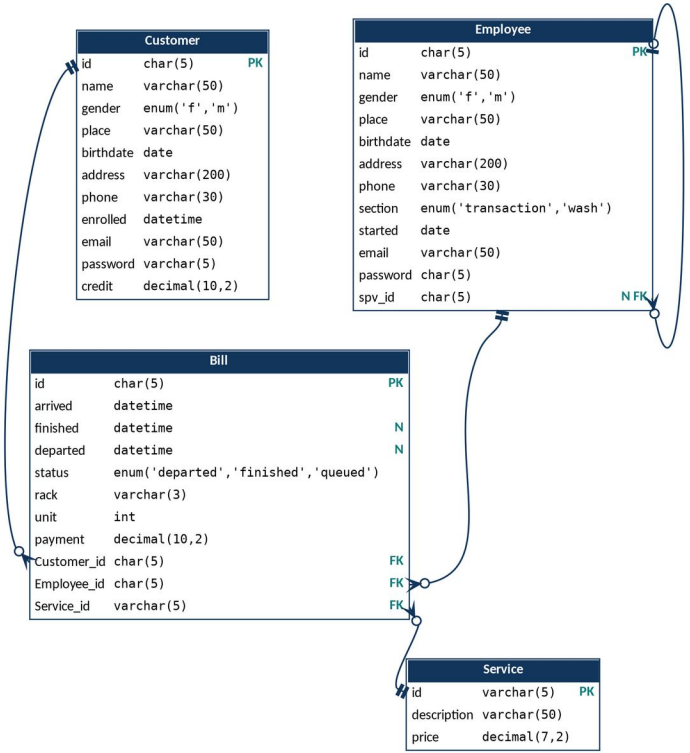
Tabel 13.2 Ke-2.953 nota basis data contoh menurut gerai dan status, dihitung dengan `GROUP BY`.

	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**

Sebagai satu tabel HTML, *grid* ini berukuran sekitar 1,2 MB (kira-kira 400 *byte* per baris); 25 baris sekali tampil, ia menjadi 119 halaman yang masing-masing sekitar 14 KB.

Pada Tabel 13.2 hanya 57 nota yang masih ada di toko; itulah sebabnya saringan bukan kemewahan. MyLaundry hanya memiliki empat nota; pada empat baris, pembagian halaman dan pengurutan hanyalah hiasan. Buku: Subbab 13.2.2 dan Tabel 13.2.

Rancangan MyLaundry



Gambar 13.1 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.

Studi kasusnya: "Buatlah program (web) untuk MyLaundry ... Fitur yang paling terlihat oleh pelanggan: daftar layanan dan harganya." Pada Gambar 13.1 bacalah kaki gagak sebagai "banyak"; `finished` dan `departed` boleh NULL selama cucian masih di toko. Buku: Subbab 13.2.3 dan 13.3.1.

Tabel Bill

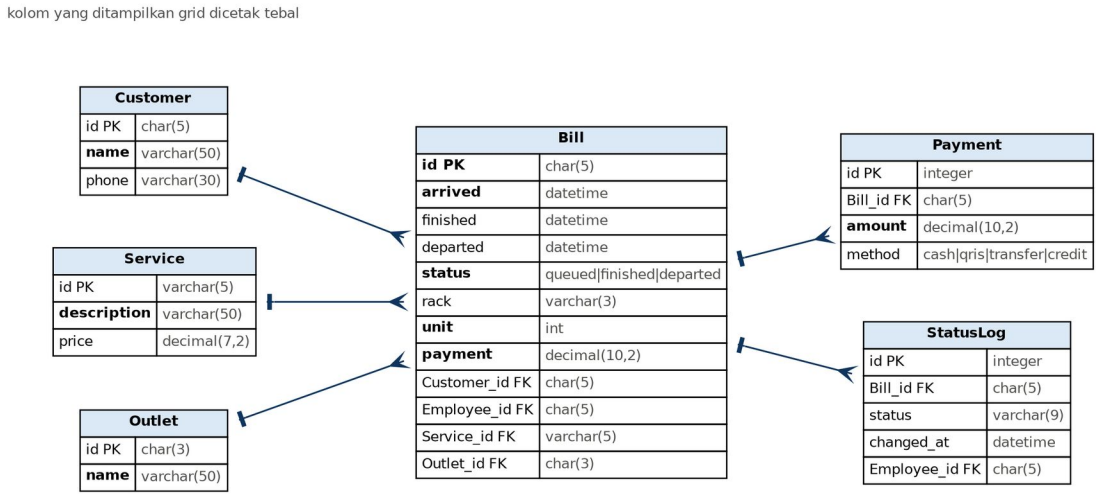
Kode 13.1 Basis data dan tabel Bill-nya, dari `MyLaundryDB.sql`. `status` adalah `enum` tiga kata, dan nota baru bernilai `queued` kecuali dinyatakan lain.

```
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)
);
```

Pada Kode 13.1, MySQL mengurutkan `enum` menurut posisi nilainya; karena disusun alfabetis, urutannya sama dengan urutan SQLite atas kata-kata yang sama sebagai teks. phpMyAdmin, yang menampilkan tabel Service dengan Edit, Copy, dan Delete di setiap baris, sendiri adalah tabel data. Program konsol membaca tabel yang sama dengan data reader. Buku: Subbab 13.3.1 dan 13.3.2, Gambar 13.3 dan 13.4.

Nota-nota Profitina Laundry



Gambar 13.2 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 aturan skema pada Gambar 13.2 mengikat grid: CHECK constraint mengaitkan status dengan stempel waktu `finished`/`departed`; setiap foreign key ke Bill adalah `ON DELETE RESTRICT`; dan `payment` adalah jumlah yang ditagihkan, sedangkan yang dibayar adalah jumlah baris Payment. Buku: Subbab 13.3.3.

GridView Terkecil

GridView me-*render* sebuah sumber data sebagai `<table>` HTML; tidak ada yang muncul sampai ia **diikat** dengan `DataBind()`. Kode 13.2 adalah seluruh halamannya.

Kode 13.2 `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>
```

Kontrol server diubah menjadi objek di server dan pada akhirnya diminta me-render dirinya sebagai HTML. `AutoGenerateColumns` *menyala secara bawaan: satu kolom per kolom data, berjudul namanya. Buku: Subbab 13.4.1 dan 13.4.2.*

Mengisi dan Mengikat *Grid*

Kode 13.3 `Page_Load` dalam `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.

```
    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();
```

Pada Kode 13.3, `Fill` milik data adapter membuka dan menutup koneksi sendiri, sehingga `con.Close()` yang terlalu awal tidak berbuat apa-apa. `DefaultView` adalah DataView yang dapat mengurutkan dan menyaring di memori. Yang sampai ke peramban adalah `<table>` biasa. Buku: Subbab 13.4.2.

Daftar Harga, Tergambar

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 13.3 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 13.3.

Gambar 13.3 kecil dan sederhana, tetapi begitulah rupa bagian dalam sebagian besar web bisnis hingga kini: hampir setiap halaman administrasi adalah gambar ini dengan lebih banyak kolom. Buku: Subbab 13.2.3 dan 13.4.2.

Mendeklarasikan Kolom

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`

Halaman kedua, `02DeleteSelect.aspx`, mematikan `AutoGenerateColumns` dan mendeklarasikan tiga `BoundField` serta dua `ButtonField` dari Tabel 13.3. Nama perintah `Delete` dan `Select` menjadi event `RowDeleting` dan `SelectedIndexChanged`. Buku: Subbab 13.5.1.

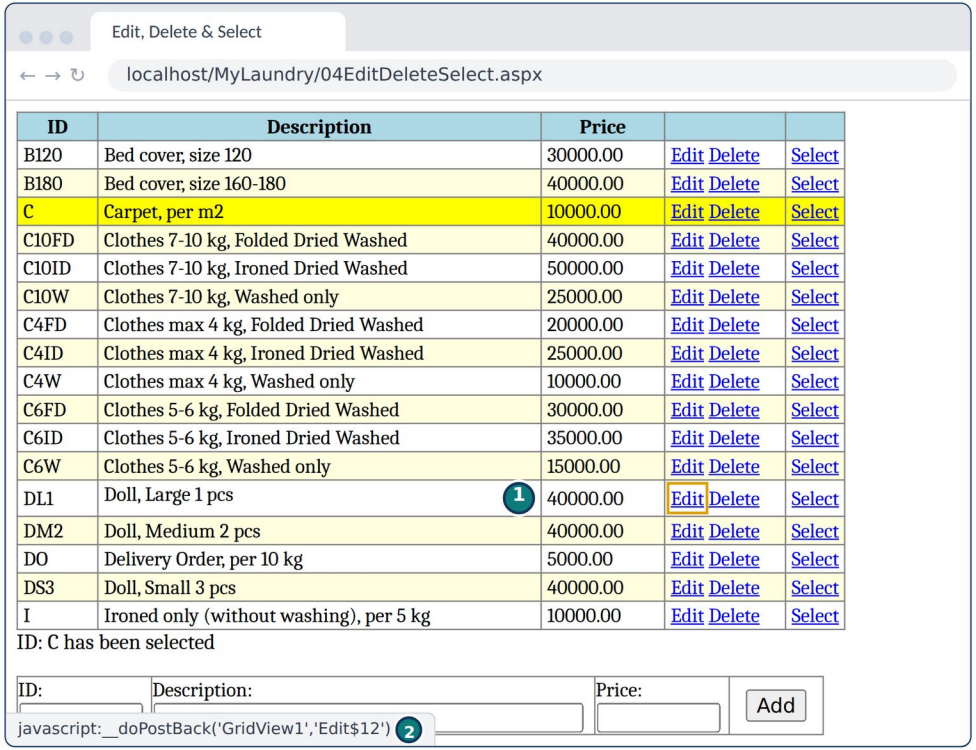
Handler `DeleteRow`

Kode 13.4 Awal `DeleteRow` dalam `02DeleteSelect.aspx.cs`: membuang sebuah baris dari *view*, lalu dari basis data. Tidak dijalankan dalam pembangunan buku ini.

```
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();
```

Bacalah Kode 13.4 perlahan: id diambil dari apa yang ditampilkan grid di sel pertama, dan `DELETE` dibangun dengan menyambung string. Keduanya keliru, dan Contoh Terkerjakan 1 menunjukkan sebabnya. Buku: Subbab 13.5.1 dan 13.5.3.

Tautan yang Memanggil C#



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

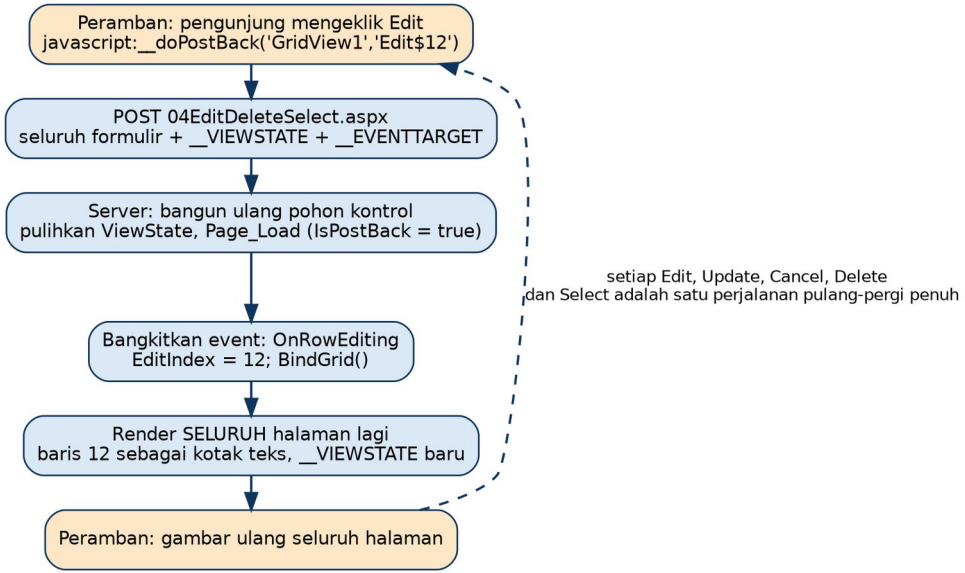
ID: C has been selected

ID:	Description:	Price:	Add

Gambar 13.4 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.

Pada Gambar 13.4 tautan itu tidak menuju alamat mana pun: `__doPostBack` menulis kontrol dan argumennya (`Edit$12`, perintah Edit pada baris 12, dihitung dari 0) ke dua field tersembunyi, `__EVENTTARGET` dan `__EVENTARGUMENT`, lalu mengirim satu-satunya formulir halaman. Buku: Subbab 13.5.2.

Satu Klik, Satu *Postback*



Gambar 13.5 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.

Gambar 13.5: ASP.NET membangun ulang pohon kontrol, memulihkan state dari `__VIEWSTATE`, menjalankan `Page_Load` dengan `IsPostBack` bernilai true, membangkitkan event, lalu me-render ulang seluruh halaman. Buku: Subbab 13.5.2.

Harga Sebuah *Postback*

Bobot. *View state* sebuah *grid* memuat cukup data untuk menggambar ulang setiap baris, dan setiap klik mengirimnya kembali.

Tanpa alamat. Setiap tindakan adalah POST ke URL yang sama: *grid* yang disaring, diurutkan, dan dibagi halaman tidak dapat ditandai (*bookmark*) atau dibagikan, dan tombol Back menanyakan pengiriman ulang formulir.

Salinan di sesi. `02DeleteSelect` menyimpan DataSet di sesi: satu kueri dihemat per klik, memori terpakai per pengunjung, dan basi begitu orang lain mengubah tabelnya.

> **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*.

Rancangan ini cemerlang pada 2002, karena memungkinkan pemrogram desktop menulis event handler tanpa harus mempelajari HTTP. Buku: Subbab 13.5.2 dan 13.5.3.

Contoh Terkerjakan 1: Data Menjadi Kode

Langkah 1. `DeleteRow` membangun pernyataannya dengan menyambung *string*:

```
cs = "DELETE FROM Service WHERE id='" + deletedID + "'";
```

Langkah 2. Seseorang memakai formulir *Add* di halaman keempat, yang *memang* memakai parameter, untuk menambahkan layanan ber-id lima karakter `'OR'1`. Nilai itu tersimpan dengan setia.

Langkah 3. Seminggu kemudian seorang rekan menghapus layanan itu di halaman ini. Pernyataannya menjadi `DELETE FROM Service WHERE id=''OR'1'`: "id kosong, atau benar". Setiap baris cocok.

Data yang disimpan dengan aman telah menjadi kode dalam perjalanan keluar: ***injeksi SQL tingkat kedua*** *(second-order SQL injection). Satu-satunya pertahanan yang andal sama dengan untuk injeksi tingkat pertama: jangan pernah menyisipkan nilai ke dalam teks SQL; ikatlah ia. Buku: Subbab 13.5.3.*

Contoh Terkerjakan 1: Diselamatkan Kebetulan

Kebetulan 1. `BoundField` meng-*encode* HTML apa yang ditampilkannya, sehingga sel berisi `'OR'1` dan pernyataan sambungan tidak cocok dengan apa pun. Matikan `HtmlEncode` (“perbaikan” yang lazim untuk `&`), dan ia cocok dengan semuanya.

Kebetulan 2. Empat layanan dipakai oleh nota, sehingga *foreign key* membuat MySQL menolak seluruh pernyataan. Pada tabel yang tidak dirujuk apa pun, setiap baris lenyap.

Kesalahan kedua. `Cells[0].Text` adalah teks hasil *render* yang sudah di-*encode*: id jujur yang mengandung `&` atau `'` pun sampai ke SQL dalam keadaan berubah. Kunci baris seharusnya diambil dari *data key grid*, bukan dari tampilannya.

> **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.

Kode yang keamanannya bersandar pada kebetulan tidaklah aman. Soal Latihan 13.1 menuliskan dua baris perbaikannya. Buku: Subbab 13.5.3 dan 13.27.

Templat dan *Data Key*

Kode 13.5 Dari *grid* `04EditDeleteSelect.aspx`: `DataKeyNames`, enam *event handler*, dan kolom Description dengan templat butir dan templat ubah. Tidak dijalankan dalam pembangunan buku ini.

```
        <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="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>
```

Pada Kode 13.5, `DataKeyNames="id"` menyimpan kunci setiap baris di view state, sehingga kunci berasal dari data, bukan dari tampilan. `<%# … %>` adalah ekspresi pengikatan data; Description berupa `Label` dalam keadaan biasa dan `TextBox` dalam mode ubah, sedangkan ID tidak memiliki templat ubah. Buku: Subbab 13.6.1.

Event yang Mengubah

Tabel 13.4 *Event* GridView yang penting untuk pengubahan. Masing-masing adalah satu *postback* (Gambar 13.5); *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)

Pengubahan hanyalah soal satu bilangan, `EditIndex` milik grid: baris yang sedang diubah, atau -1 bila tidak ada. `OnRowEditing` mengaturnya lalu mengikat ulang; `OnRowCancelingEdit` mengembalikannya ke -1. Setiap handler pada Tabel 13.4 diakhiri dengan mengikat ulang grid. Buku: Subbab 13.6.1 dan 13.6.2.

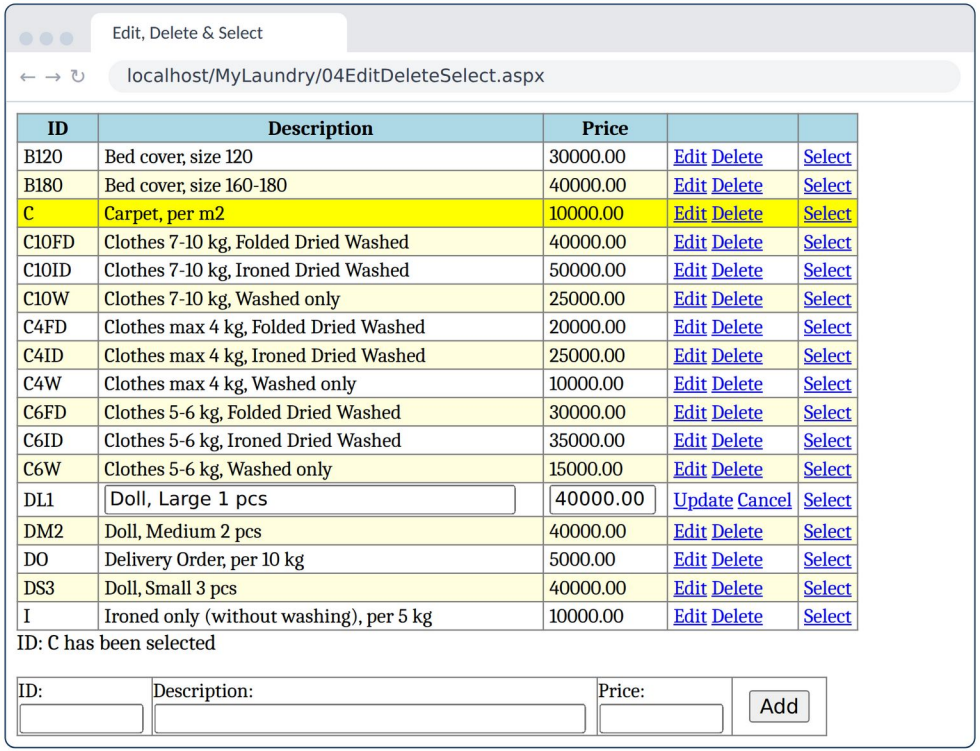
Mengubah Sebuah Baris

Kode 13.6 Awal `OnRowUpdating` dalam `04EditDeleteSelect.aspx.cs`: menemukan kotak teks, mengambil kunci dari `DataKeys`, dan menjalankan `UPDATE` berparameter. Tidak dijalankan dalam pembangunan buku ini.

```
    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);
```

Kode 13.6 mengikat setiap nilai; blok `using` menutup koneksi, perintah, dan adapter bahkan setelah eksepsi. Setelah pengubahan, phpMyAdmin menampilkan DL1 sebagai Doll, Large 1 pcs (special). Buku: Subbab 13.6.2 dan Gambar 13.9.

Baris DL1 dalam Mode Ubah



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 13.6 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 13.5, sehingga tetap berupa teks biasa.

Gambar 13.6: Update dan Cancel telah menggantikan Edit dan Delete pada baris yang indeksnya sama dengan `EditIndex`. Buku: Subbab 13.6.2.

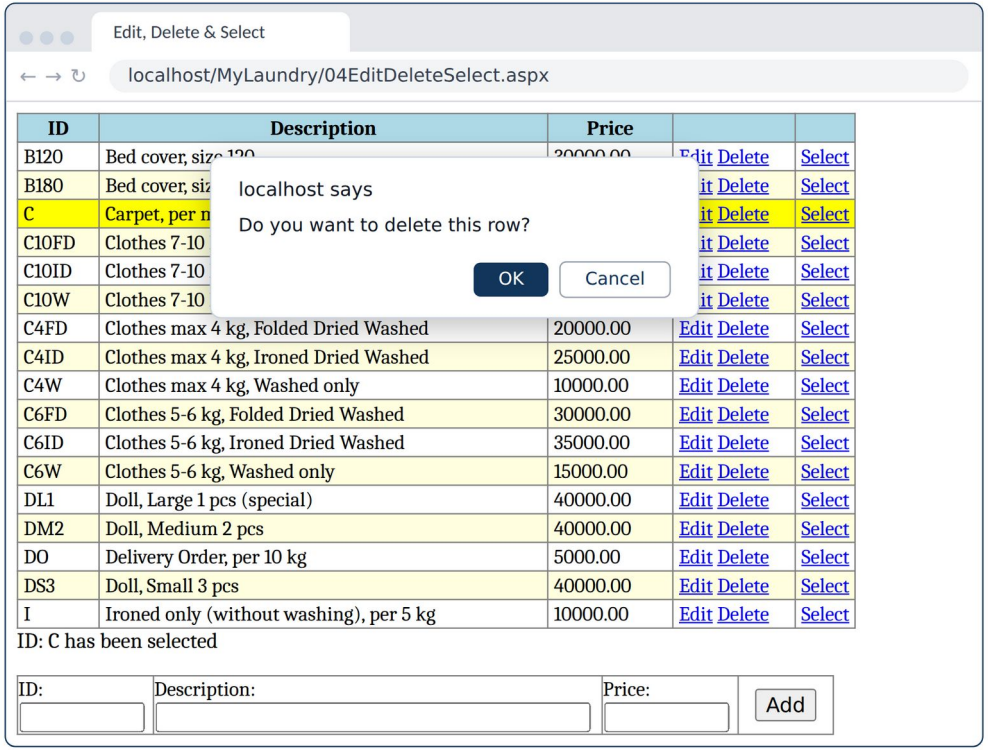
Menghapus, Setelah Bertanya

Kode 13.7 `OnRowDataBound` dan awal `OnRowDeleting` dalam `04EditDeleteSelect.aspx.cs`: `confirm()` pada setiap tautan Delete, dan penghapusan menurut kunci baris dengan parameter. Tidak dijalankan dalam pembangunan buku ini.

```
    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"] =
                "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);
```

Pada Kode 13.7, `Controls[2]` adalah tautan Delete, kontrol ketiga di sel command field. Dialog itu kemudahan bagi orang jujur, bukan kendali keamanan: siapa pun dapat mengirim postback tanpa melewatinya. Buku: Subbab 13.6.2.

Peramban yang Bertanya



ID	Description	Price		
B120	Bed cover, siz… 120	30000.00	Edit Delete	Select
B180	Bed cover, siz…		it Delete	Select
C	Carpet, per m…		it Delete	Select
C10FD	Clothes 7-10		it Delete	Select
C10ID	Clothes 7-10		it Delete	Select
C10W	Clothes 7-10		it 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 (special)	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 13.7 Dialog konfirmasi peramban, yang dipasang pada tautan Delete oleh `OnRowDataBound`. Dialog ini digambar oleh peramban, bukan oleh halaman, sehingga rupanya berbeda di setiap peramban.

Gambar 13.7: Cancel menghentikan postback; OK menghapus. Formulir Add mengembalikan DL1; ID kosong menulis `alert` dengan `Response.Write`, yang merusak HTML dan membuka celah XSS. Buku: Subbab 13.6.2–13.6.3, Gambar 13.11–13.13.

Mendeklarasikan Data: `SqlDataSource`

Kode 13.8 Perintah-perintah sumber data `dsBills` dalam `LaundryBills/Bills.aspx`: saringan opsional, dan satu `UPDATE` yang mengatur status beserta stempel waktunya sekaligus. Tidak dijalankan dalam pembangunan buku ini.

```
        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">
```

Pada Kode 13.8, `(@outlet = '' OR b.Outlet_id = @outlet)` bernilai benar untuk setiap baris di bawah All outlets. Kode di belakangnya menolak langkah yang dilompati dan hanya menghapus nota queued yang belum dibayar. Keterbatasannya: 2.953 baris diambil untuk menampilkan 25, tidak ada baris StatusLog, state tersembunyi di view state. Buku: Subbab 13.7.

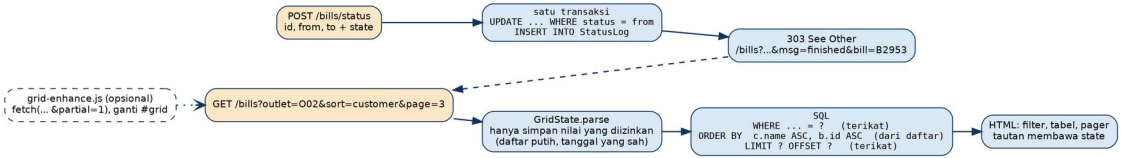
Dari GridView ke Razor Pages

Tabel 13.5 Dari GridView ke *grid* Razor Pages: enam baris pertama dari sebelas.

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=002&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

ASP.NET Core tidak memiliki GridView: kontrol itu tidak punya pijakan di sana. Bacalah kolom kanan Tabel 13.5 sebagai rancangan grid modern. Buku: Subbab 13.8.1 dan Tabel 13.6.

Dua Jenis Permintaan



Gambar 13.8 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()`.

Gambar 13.8: pembacaan adalah GET yang seluruh state-nya ada di query string; perubahan adalah POST yang membawa nota, status lama dan barunya, serta state grid, lalu dijawab dengan pengalihan 303 kembali ke sebuah pembacaan. Buku: Subbab 13.8.2.

Satu Kontrak untuk Empat *Grid*

Tabel 13.6 Skema alamat *grid*: tujuh parameter *state grid*.

Parameter	Nilai yang diizinkan	Bawaan (tidak ditulis di alamat)	Nilai lain
`outlet`	id dari tabel Outlet: `001`, `002`, `003`	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

Keempat grid menjawab alamat-alamat ini (Tabel 13.6); `edit`, `delete`, `msg`, `bill`, dan `partial` melengkapi daftarnya. Nilai yang buruk sekadar tidak dipakai: `?sort=b.id;DROP TABLE Bill` menampilkan urutan bawaan. Buku: Subbab 13.8.3 dan Tabel 13.7.

Mengurai *State*

Kode 13.9 `GridState.Parse` dalam `Data/GridState.cs`: setiap properti *state* baru telah lolos uji, atau bernilai bawaannya.

```
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);
}
```

`GridState` adalah record C# yang tak dapat diubah (Kode 13.9); `int.TryParse` gagal untuk `"abc"`, `""`, dan `"99999999999"`. Argumen fungsi membuat satu metode dapat membaca `Request.Query` untuk GET dan `Request.Form` untuk POST. Buku: Subbab 13.9.1.

Membangun Setiap Tautan

Kode 13.10 `Href` dan `SortHref` dalam `Data/GridState.cs`: alamat untuk *state* ditambah pasangan tambahan, dan aturan pengurutan yang diharapkan setiap pengguna *spreadsheet*.

```
    /// <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();
```

`Params()` *meninggalkan setiap nilai bawaan, sehingga satu state memiliki satu alamat. Pada Kode 13.10,* `Uri.EscapeDataString` *meng-encode setiap nilai. Formulir POST membawa state dalam field tersembunyi, dan pengalihan dibangun ulang dengan* `Parse` *yang sama: alamat* `back=` *yang dikirim formulir akan menjadi open redirect. Buku: Subbab 13.9.1 dan 13.9.2.*

Saringan sebagai Nilai Terikat

Kode 13.11 `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
```

Untuk `outlet=002, status=queued`, *Kode 13.11 menghasilkan* `1 = 1 AND b.Outlet_id = ? AND b.status = ?` *dan* `['002', 'queued']`. *Pengikatan membuat nilai tidak berbahaya; daftar putih membuatnya bermakna. Buku: Subbab 13.10.1.*

Contoh Terkerjakan 2: “Sampai” Berarti Sehari Penuh

Langkah 1. `arrived` menyimpan tanggal dan jam, `'2026-09-30 16:26:26'`. `b.arrived <= '2026-09-30'` yang naif membandingkan teks, dan *string* yang lebih panjang terurut *sesudah* yang lebih pendek: setiap nota tanggal 30 hilang.

Langkah 2. Bandingkan dengan awal hari *berikutnya*, sebagaimana dinyatakan Persamaan (13.1):

$$\text{from} \le \text{arrived} < \text{to} + 1\,\text{hari} \tag{13.1}$$

Langkah 3. Untuk September 2026: `'2026-09-01' <= arrived < '2026-10-01'` memilih 256 nota; bentuk naif memilih 246, dan sepuluh yang terbuang semuanya masih harus dikerjakan.

Program yang menambahkan satu hari (`date.fromisoformat(...) + timedelta(days=1)`), sehingga SQL-nya sama di SQLite dan MySQL; `2026-02-30` berbentuk benar tetapi dibuang. Buku: Subbab 13.10.2.

Kolom Tidak Dapat Diikat

Kode 13.12 Keluaran `bench/order_by_param.py`: diikat, `'c.name'` hanyalah *string*; ditulis, `c.name` mengurutkan menurut nama.

```
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')]
```

Kode 13.13 `SORTS` dalam `python/grid.py`: delapan kunci, masing-masing dipetakan ke SQL yang 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",
}
```

Kode 13.12: placeholder hanya mewakili nilai. Dengan Kode 13.13, pengunjung memilih kunci dan program memilih SQL-nya. Buku: Subbab 13.11.1–13.11.2.

Daftar Putih dan Pemutus Seri

Kode 13.14 `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
```

Kode 13.15 Keluaran nyata `bench/tie_breaker.py`: sepuluh nota yang diurutkan menurut jumlah, tanpa dan dengan id sebagai pemutus seri. Empat nota terakhir berbeda.

```
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
```

Pada Kode 13.15 empat dari tujuh belas nota seri di Rp 135.000, dan SQL tidak menjanjikan urutan apa pun di antara yang seri: sebuah nota dapat muncul di dua halaman sementara yang lain tidak pernah muncul. Kode 13.14 membuat urutannya total. Aturannya: kueri berhalaman harus mengakhiri ORDER BY-nya dengan kolom unik. Buku: Subbab 13.11.2 dan 13.11.3.

Pembagian Halaman Adalah Aritmetika

N baris yang cocok, ukuran halaman s, halaman p dihitung dari 1. Banyaknya halaman adalah Persamaan (13.2):

$$P = \max\left(1, \lceil \frac{N}{s} \rceil\right) \tag{13.2}$$

Offset, yaitu banyaknya baris yang dilewati sebelum halaman p, adalah Persamaan (13.3):

$$o = (p - 1)\, s \tag{13.3}$$

Baris yang ditampilkan, dicetak sebagai “Showing 26–50 of 2,953 bills”, adalah Persamaan (13.4):

$$\text{first} = o + 1, \text{last} = \min(p\, s, N) \tag{13.4}$$

$\lceil x \rceil$ adalah bilangan bulat terkecil yang tidak kurang dari x; $\max$ mempertahankan satu halaman kosong bila tidak ada yang cocok. Buku: Subbab 13.12.1.

Contoh Terkerjakan 3: 2.953 Nota dalam Halaman

Tabel 13.7 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

Baris keempat Tabel 13.7: $o=(119-1)\times 25=2950$*, baris pertama yang ditampilkan adalah* 2951*, dan yang terakhir* $\min(119\times 25,2953)=\min(2975,2953)=2953$*: halaman terakhir berisi tiga nota. Soal Latihan 13.2 mengerjakan hal yang sama untuk 996 nota Keputih. Buku: Subbab 13.12.1.*

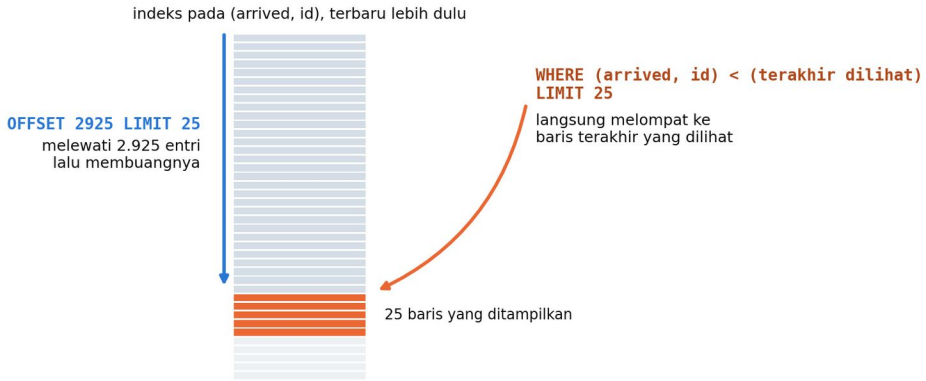
Satu Halaman, Dua Kueri

Kode 13.16 `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}
```

Pada Kode 13.16 kedua kueri memakai klausa WHERE yang sama, sehingga hitungan dan baris selalu sepakat; penjepitan mengubah `?page=999` menjadi halaman terakhir. PDO milik PHP mengemulasi prepared statement untuk MySQL dan akan mengirim `LIMIT '25'`, sehingga grid PHP mengikat limit dan offset dengan `PDO::PARAM_INT`. Buku: Subbab 13.12.2.

OFFSET Berjalan, *Keyset* Mencari



Gambar 13.9 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.

`LIMIT 25 OFFSET 2925` *tidak dapat melompat ke baris 2.926: tabel tidak memiliki nomor baris, sehingga basis data menghasilkan baris-baris terurut lalu menghitung 2.925 di antaranya. OFFSET melewatkan keluaran, bukan pekerjaan (Gambar 13.9). Buku: Subbab 13.13.1 dan 13.13.2.*

Dua Model Biaya dan Satu Syarat

OFFSET berbiaya tetap t_0 ditambah c untuk setiap baris yang dilewati, Persamaan (13.5):

$$T_{\text{offset}}(p) \approx t_0 + c\,(p-1)\,s \tag{13.5}$$

Keyset paging melanjutkan sesudah baris terakhir yang ditampilkan, (a_0, i_0), dengan syarat Persamaan (13.6), dan pencarian indeks berbiaya Persamaan (13.7):

$$a < a_0 \vee (a = a_0 \wedge i < i_0) \tag{13.6}$$

$$T_{\text{keyset}} \approx t_0 + c' \log N \tag{13.7}$$

Dalam kata-kata: tiba lebih awal, atau tiba pada detik yang sama persis dengan id yang lebih kecil. Klausa kedua itulah sebabnya pemutus seri wajib di sini. Untuk grid menaik, kedua < *menjadi* >*. Buku: Subbab 13.13.1 dan 13.13.2.*

Baris yang Sama, Ditulis agar Dapat Dicari

Bentuk OR, `WHERE (arrived < ? OR (arrived = ? AND id < ?))`, dan *row value*, `WHERE (arrived, id) < (?, ?)`, memilih baris yang sama dengan Persamaan (13.8):

$$a \le a_0 \wedge (a < a_0 \vee i < i_0) \tag{13.8}$$

Bukti per kasus. $a<a_0$: kurung bernilai benar. $a=a_0$: kurung menjadi $i<i_0$. $a>a_0$: keduanya salah.

Barisnya sama; perencana kuerinya berbeda. `arrived <= ?` *di depan adalah rentang pada kolom pertama indeks, yang menjadi seek; SQLite membaca bentuk OR sebagai scan. Buku: Subbab 13.13.3.*

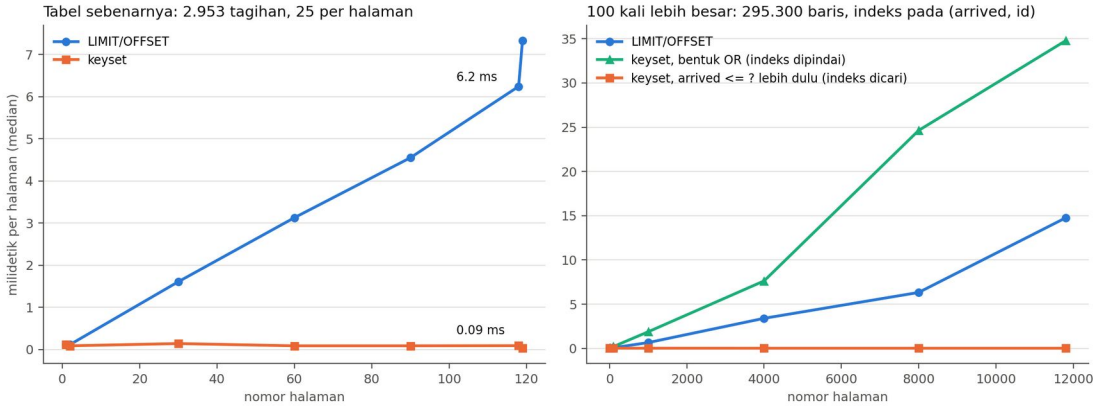
Diukur pada Nota-nota Laundry

Kode 13.17 Baris-baris dari Bagian 1 dan 2b keluaran nyata `bench/paging_bench.py` (SQLite 3.45.1). Median dalam milidetik; "same rows? yes" berarti halaman *keyset* sama dengan halaman OFFSET.

```
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 2b: 295,300 rows (100 copies) = 11,812 pages, with an index on (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
    8000   199975      6.327        24.636         0.023         0.023  yes
   11812   295275     14.773        34.795         0.023         0.023  yes
```

Kode 13.17, Bagian 1: OFFSET tumbuh sekitar 2,1 µs per baris yang dilewati, keyset bertahan di sekitar 0,1 ms, dan mengambil seluruh 2.953 baris memakan 9,504 ms, sekitar 84 kali satu halaman. Bagian 2b, 295.300 baris dengan indeks: OFFSET mencapai 14,773 ms; bentuk OR melakukan scan dan lebih buruk; row value dan bentuk awalan melakukan SEARCH dalam 0,023 ms. Buku: Subbab 13.13.4.

Hasil Pengukuran, Tergambar



Gambar 13.10 Milidetik per halaman terhadap nomor halaman, dari Kode 13.17. 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.

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.

Pada Gambar 13.10, tanpa indeks (Bagian 2a) setiap metode mengurutkan 295.300 baris per halaman dan OFFSET naik hingga 399 ms: untuk grid besar, indeks bukan pilihan. Buku: Subbab 13.13.4.

Mana yang Dipakai *Grid*

OFFSET, dan memang tepat: halaman nyata terburuknya berbiaya 6 ms, dan hanya OFFSET yang memberi nomor halaman, tautan *Last*, dan "page 60 of 119".

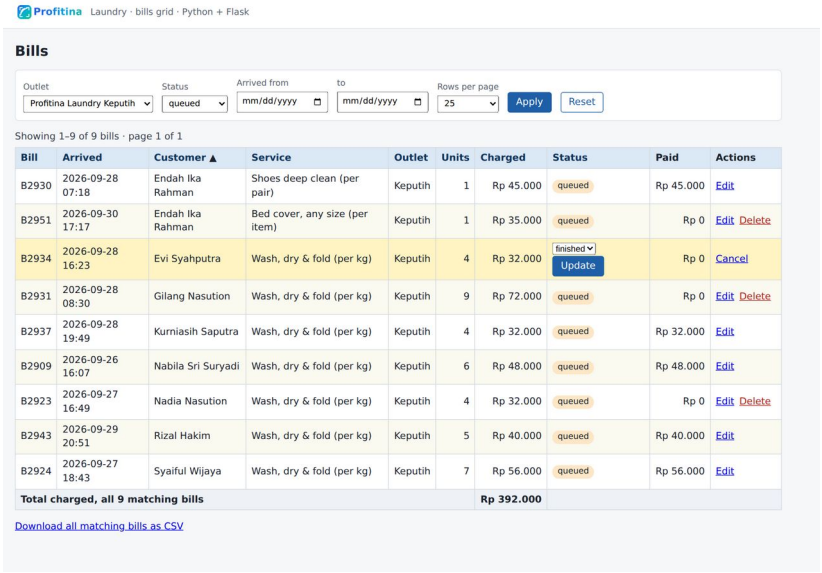
Keyset cocok untuk umpan (*feed*), *infinite scroll*, tabel yang sangat besar, dan tabel yang berubah selagi dibaca orang. Kode 13.18 adalah versi *grid* Flask.

Kode 13.18 `fetch_after()` dalam `python/grid.py`: *keyset paging* dengan syarat ramah-indeks, di bawah saringan yang sama dengan halaman bernomor.

```python
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 2.953 nota menurut nomor halaman dan melalui tautan Older →, lalu memeriksa bahwa kedua penelusuran sepakat, nota demi nota. Buku: Subbab 13.13.5.*

Mengubah Status di Tempat



Gambar 13.11 *Grid* Flask pada `/bills?outlet=002&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`.

Sebuah nota bergerak `queued → finished → departed`, satu langkah sekali; setiap langkah mencatat waktunya dan menulis satu baris StatusLog. Tautan Edit pada Gambar 13.11 adalah grid saat ini ditambah `edit=B2934`: `EditIndex` versi modern. Buku: Subbab 13.14.1.

Baris Ubah

Kode 13.19 Baris ubah dalam `python/templates/_grid.html`: *state grid* dalam *field* tersembunyi, id nota dan status *saat ini*, 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>
```

Kode 13.19 juga mengirim status baris ketika digambar (`from`). Buku: Subbab 13.14.1.

Contoh Terkerjakan 4: Dua Kasir, Satu Nota

Kode 13.20 `advance()` dalam `python/grid.py`: satu langkah yang sah, bersyarat pada status yang masih `old`, dengan baris StatusLog-nya dalam transaksi yang sama.

```python
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
```

Kedua kasir melihat B2934 queued. POST pertama mengubah satu baris. POST kedua menjalankan `UPDATE … WHERE id = ? AND status = ?` *yang sama dengan* `queued`*, cocok dengan* ***nol baris****, mendapati notanya ada, lalu mengembalikan* `conflict` *(Kode 13.20). Inilah konkurensi optimistis;* `BEGIN IMMEDIATE` *memegang kunci tulis SQLite sejak awal. Buku: Subbab 13.14.2.*

Post/Redirect/Get

Kode 13.21 Dua *handler* POST dalam `python/app.py` dan pengalihan yang mereka pakai bersama. `code=303` adalah *See Other*: "ambillah hasilnya dengan GET".

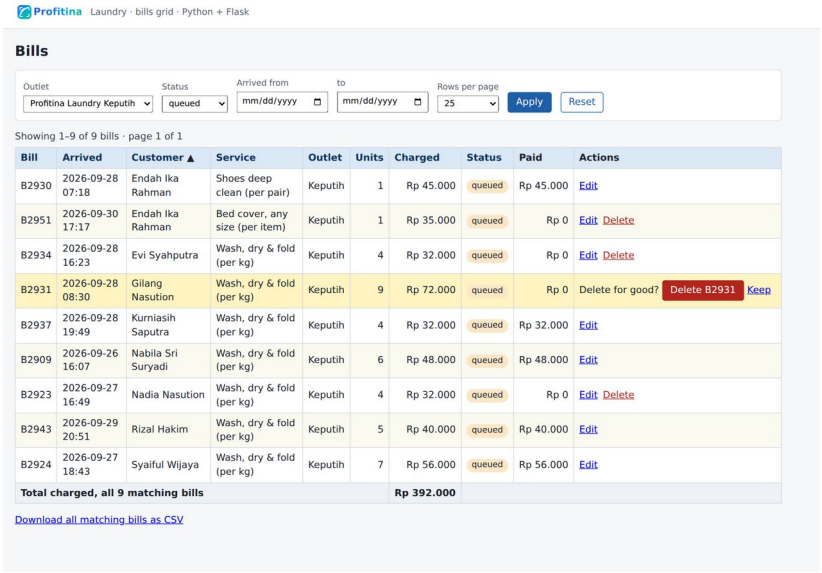
```
def back_to_grid(code, bill_id):
    """Post/Redirect/Get: rebuild the grid's address from validated values only."""
    state = current_state(request.form)
    bill = bill_id if grid.BILL_RX.match(bill_id) else ""
    pairs = state.params() + [("msg", code), ("bill", bill)]
    return redirect("/bills?" + 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)
```

Pada Kode 13.21 alamat membawa kode pesan dan id nota, bukan kalimat; memuat ulang hanya mengulangi GET. Buku: Subbab 13.14.3.

Bertanya Tanpa Skrip



Gambar 13.12 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.

Aturannya: ***nota hanya boleh dihapus selama masih queued dan belum memiliki pembayaran*** *(delapan nota). Tautan Delete menuju grid ditambah* `delete=B2931`*, dan server menggambar pertanyaannya sebagai baris dengan formulir sungguhan (Gambar 13.12); ia bekerja tanpa skrip dan bagi pembaca layar. Buku: Subbab 13.15.1 dan 13.15.2.*

Penghapusan, Diperiksa dalam Satu Transaksi

Kode 13.22 `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"
```

Memeriksa dan menghapus dalam langkah terpisah adalah check-then-act race: sebuah pembayaran dapat masuk di antaranya. Kode 13.22 menghapus baris StatusLog lebih dulu karena `ON DELETE RESTRICT` *akan menolak menghapus nota selama baris itu ada. Buku: Subbab 13.15.3.*

CSV yang Aman Dibuka

Kode 13.23 `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.

```python
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"
```

Kode 13.23: tanda petik dan CR LF (RFC 4180); BOM untuk Excel; `=`, `+`, `-`, `@` di awal diberi apostrof (CSV injection). Buku: Subbab 13.16.

Razor Pages: Menolak POST Lintas Situs

Kode 13.24 *Middleware* dalam `Program.cs` yang menolak POST dari situs lain, memakai *header* `Sec-Fetch-Site` dan `Origin`.

```
// 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;
        }
    }
    await next();
});
```

Kode 13.24 menjawab formulir di `evil.example` dengan 403 (CSRF); token antiforgery adalah penjaga kedua. Tanpa NuGet, grid memakai penyimpanan di memori. Buku: Subbab 13.17.1–13.17.2.

Razor Pages: *Page Model*

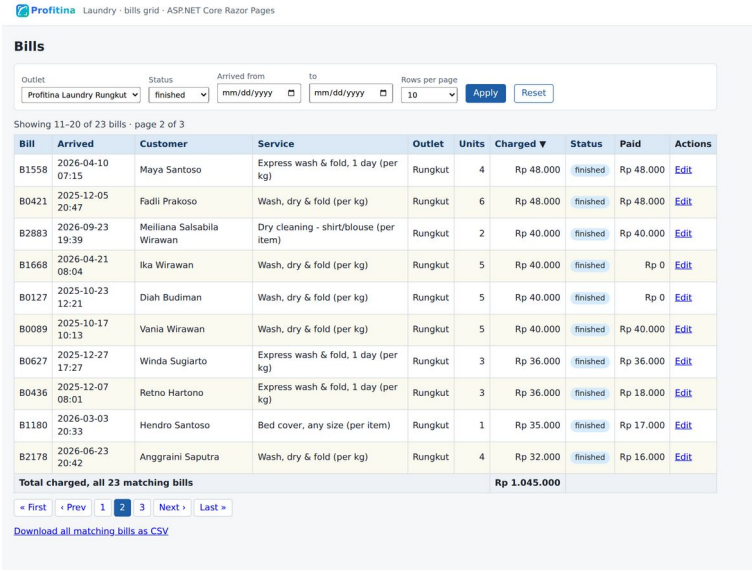
Kode 13.25 `OnGet` dan `OnPostStatus` dalam `Pages/Bills.cshtml.cs`: mengisi properti yang dibaca templat (atau hanya mengembalikan *grid* bila `partial=1`), atau mengubah sebuah nota.

```
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);
}
```

Bandingkan Kode 13.25 dengan `Page_Load`: tanpa `IsPostBack`, tanpa langkah pengikatan, tanpa view state. `@page "/bills/{handler?}"` memetakan `POST /bills/status` ke `OnPostStatus`, dan `BackToGrid` menjawab 303. Razor meng-encode setiap `@value`. Buku: Subbab 13.17.3 dan 13.17.4.

Grid Razor Pages Berjalan



Gambar 13.13 *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.

Gambar 13.13 disaring, diurutkan, dan dibagi halaman, dan seluruh state-nya ada di bilah alamat: tandai, bagikan, muat ulang. Buku: Subbab 13.17.5.

Progressive Enhancement dengan `fetch()`

Kode 13.26 `load()` dalam `static/grid-enhance.js`: mengambil fragmen *grid* dengan `partial=1`, menukarnya di tempat, menjaga bilah alamat tetap selaras, dan kembali ke pemuatan halaman penuh bila ada kegagalan apa pun.

```
    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
      });
  }
```

Grid bekerja sepenuhnya dengan tautan dan formulir; Kode 13.26 hanya membuatnya lebih baik. `pushState` *menjaga alamat yang nyata dan dapat dibagikan;* `popstate` *menangani Back; tautan Delete mendapat* `confirm()`*; apa pun yang janggal menjadi pemuatan halaman biasa. Playwright memeriksa delapan perilaku pada keempat grid. Buku: Subbab 13.21.*

Satu Himpunan Pemeriksaan untuk Empat Server

Kode 13.27 Baris-baris dari keluaran nyata `tests/run_checks.sh java`: 44 pemeriksaan atas pembacaan, injeksi, alur kerja status, penghapusan, POST lintas situs, dan basis data sesudahnya.

```
PASS  an injected sort column is ignored (default order)
PASS  an injected outlet is ignored (no filter)
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 POST from another origin is refused with 403
PASS  a cross-site POST (Fetch Metadata) is refused with 403
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
```

Pemeriksaan pada Kode 13.27 menghitung jawaban yang benar dengan SQL-nya sendiri dari basis data yang masih asli, sehingga server lulus hanya bila sepakat dengan perhitungan yang independen. PHP, Java, dan Flask lulus 44 dari 44; Razor Pages 40 dari 40 (tanpa berkas basis data); halaman Web Forms diurai Roslyn, tidak dijalankan. Buku: Subbab 13.22.

Profitina dalam Praktik

Riwayat adalah *grid* atas tabel yang terus tumbuh: pada *dashboard* skor mingguan, indeks yang melayani urutan *grid* membuat minggu ke-100 secepat minggu pertama.

Ekspor adalah fitur, dan data pribadi ikut di dalamnya: ekspor CSV mengikuti saringan layar, urutan yang stabil, dan tanpa rumus hidup; kewajiban UU PDP diperlakukan sebagai fitur.

Umpan adalah rumah *keyset paging*: kiriman terus datang di atas, dan pembaca tidak boleh melihat satu kiriman dua kali atau melewatkannya.

Dua basis data, satu SQL, dan **seketika, tetapi tidak pernah bergantung pada triknya:** kiriman yang gagal terkirim dipulihkan.

Profitina berjalan di atas FastAPI dengan PostgreSQL di `app.profitina.com`; studi kasus yang dapat dijalankan tetap Profitina Laundry. Buku: Subbab 13.23.

Kesalahan yang Sering Terjadi

`"ORDER BY " + sort`, atau `ORDER BY ?` yang diam-diam mengurutkan menurut konstanta; mengurutkan kueri berhalaman tanpa pemutus seri yang unik.

Pembagian halaman di memori; *keyset* hanya dengan bentuk OR; `arrived <= '2026-09-30'` untuk “sampai tanggal 30”; `LIMIT '25'` dari PDO dengan MySQL.

Kunci yang dibaca dari teks tampilan; memercayai basis data sendiri (injeksi tingkat kedua); mengandalkan `confirm()` atau tautan yang disembunyikan.

Pengubahan yang mengabaikan apa yang dilihat pengguna; perubahan dan baris audit dalam transaksi terpisah; menjawab POST dengan halaman; mengalihkan ke alamat dari formulir; teks pesan di alamat.

CSV tanpa tanda petik, BOM, atau penjaga rumus; “peningkatan” yang tanpanya *grid* tidak dapat bekerja.

Buku mendaftar setiap kesalahan beserta perbaikannya. Buku: Subbab 13.24.

Ringkasan: Lima Hal untuk Diingat

1. Tabel data mengerjakan tujuh pekerjaan (Tabel 13.1); GridView mengerjakannya dengan *postback* dan *view state*, membagi halaman di memori, dengan *state* yang tidak terlihat di bilah alamat.

2. *Grid* modern adalah tautan dan formulir: pembacaan adalah GET dengan seluruh *state* di alamat; perubahan adalah POST yang dijawab dengan 303 (Gambar 13.8).

3. Nilai saringan diikat; kunci pengurutan berasal dari daftar putih dan diakhiri pemutus seri yang unik; “sampai” berarti sebelum tengah malam berikutnya (Persamaan (13.1)).

4. Pembagian halaman adalah aritmetika (Persamaan (13.2), Persamaan (13.3)); OFFSET berjalan, *keyset* mencari bila syaratnya memungkinkan indeks dicari (Persamaan (13.8)).

5. Pengubahan bersifat bersyarat dan transaksional, penghapusan sempit dan dikonfirmasi oleh server, ekspor diberi tanda petik dan aman dari rumus; satu skrip meningkatkan keempat *grid* tanpa dibutuhkan oleh satu pun.

Buku: Subbab 13.25; Soal Latihan 13.1–13.12 beserta penyelesaiannya di Subbab 13.26 dan 13.27.

Pertemuan Berikutnya

Bab 14 — Validasi dan Manajemen State. Pemeriksaan di peramban adalah kesopanan; pemeriksaan di server adalah hukum.

Enam *validator* Web Forms, lalu *data annotation* di Razor Pages; *view state*, *cookie*, sesi, TempData, dan *cache*, dipilih dengan bertanya *siapa yang boleh melihat ini?* dan *berapa lama ia harus hidup?*

Sebelum itu: kerjakan Soal Latihan 13.2, 13.4, 13.6, dan 13.10, lalu matikan JavaScript dan pakailah *grid*-nya (Soal Latihan 13.11).

`Clock.Actor` *dan* `LAUNDRY_ACTOR` *hari ini menggantikan sementara sebuah proses masuk (login); Bab 14 mulai menggantikannya. Buku: Bab 14.*

Kuliah 14 — Validasi dan Manajemen State

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 14

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini kita menghentikan data buruk di pintu masuk dan membuat protokol yang pelupa menjadi ingat, di meja depan Profitina Laundry. Buku: Bab 14, Subbab 14.1–14.15.

Cakupan Pertemuan Ini

1. **Mengapa memvalidasi, dan di mana** — tiga jenis data buruk; peramban membantu, server memutuskan.
2. **Validator Web Forms** — *required*, *compare*, *range*, *regular expression*, *custom*, *summary*; formulir Laundry; *user control*.
3. **Validasi di Razor Pages** — *data annotation*, atribut khusus untuk nomor ponsel Indonesia, `ModelState`, pemeriksaan peramban yang *unobtrusive*.
4. **HTTP lupa** — peta mekanisme *state*, dipilih menurut cakupan dan masa hidup.
5. ***State* klien dan server** — *view state*, *hidden field*, *query string*, *cookie*; sesi, TempData, *application state*, *cache*.
6. **Empat *track*, keamanan, dan UU PDP** — satu meja depan dalam Razor Pages, PHP, Java, dan Flask, serta di mana data pribadi boleh tinggal.

Validasi menyangkut satu permintaan; manajemen state menyangkut banyak permintaan. Keduanya terus bertemu, karena setiap state yang kembali dari peramban adalah masukan. Buku: Subbab 14.1.

Posisi Kita Saat Ini

Bab 13 mengisi sebuah *grid* dengan nota-nota Laundry. Bab-bab sebelumnya meninggalkan dua janji: *bagaimana jika orang yang mengetik keliru, atau berbohong?* dan *bagaimana kotak teks Web Forms mengingat nilainya, padahal HTTP tidak mengingat apa pun?*

Validasi memutuskan, untuk setiap nilai yang tiba, apakah nilai itu boleh masuk ke sistem. **Manajemen *state*** membawa informasi dari satu permintaan ke permintaan berikutnya.

> **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*.

Cookie gerai, id pelanggan tersembunyi, dan bilangan di query string semuanya tampak seperti ingatan, dan semuanya adalah masukan. Buku: Subbab 14.1.

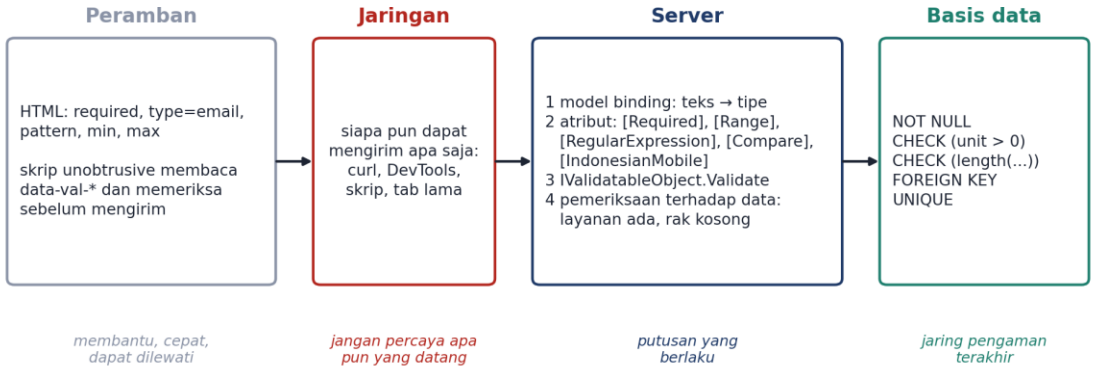
Tiga Jenis Data Buruk

Tabel 14.1 Tiga jenis data buruk, dengan contoh dari meja depan Profitina Laundry.

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

*Pada **Tabel 14.1** hanya nilai tak berguna yang dapat dikenali sendirian; dua jenis lainnya memerlukan pengetahuan server. Peramban dapat membantu, tetapi tidak pernah memutuskan. Buku: Subbab 14.2.1.*

Empat Tempat yang Dilalui Sebuah Nilai



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:* `curl -d "Input.Name=" http://127.0.0.1:9401/customers/new` *mengirim formulir yang tidak pernah dilihat pemeriksaan peramban mana pun. Constraint basis data adalah jaring pengaman, bukan antarmuka pengguna. Buku: Subbab 14.2.2.*

Sisi Klien dan Sisi Server

> **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.

Di peramban: JavaScript, atau atribut HTML yang dipahami peramban secara bawaan — `required`, `type="email"`, `pattern`, `min`, `max`.

Di server: setelah permintaan tiba dan sebelum apa pun disimpan — putusan yang berlaku.

JavaScript dapat dimatikan, halaman dapat diubah lewat developer tools, atau peramban dilewati sama sekali. Buku: Subbab 14.2.2.

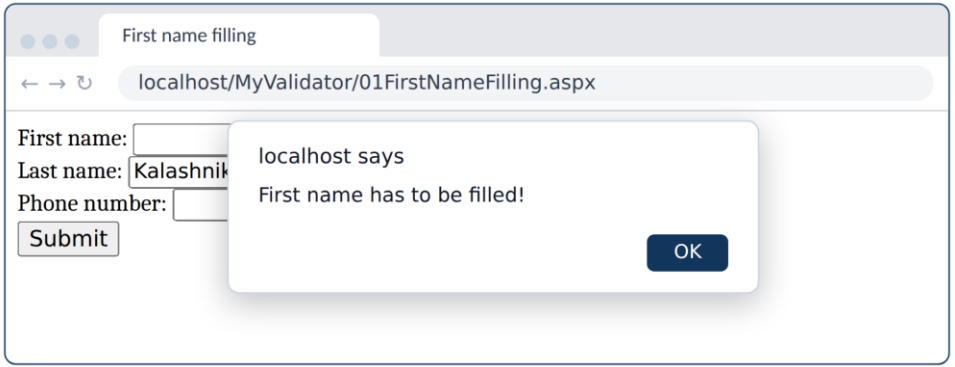
Pemeriksaan yang Ditulis Sendiri

Kode 14.1 Skrip dan tag formulir `01FirstNameFilling.aspx` (Web Forms, tidak dijalankan di sini): `onsubmit` memanggil fungsi yang menolak nama depan kosong.

```
    <script>
        function checkForm() {
            var x = document.forms["myForm"]["fname"].value;
            if (x == "") {
                alert("First name has to be filled!");
                return false;
            }
            return true;
        }
    </script>
    <form id="myForm" runat="server"
        action="01Done.html" onsubmit="return checkForm()"
        method="get">
```

*Pada **Kode 14.1**, nilai `false` dari `checkForm()` membatalkan pengiriman. Tidak ada yang memeriksa nama depan di server: `01Done.html` mengatakan Your data has been saved! kepada siapa saja. Buku: Subbab 14.2.3.*

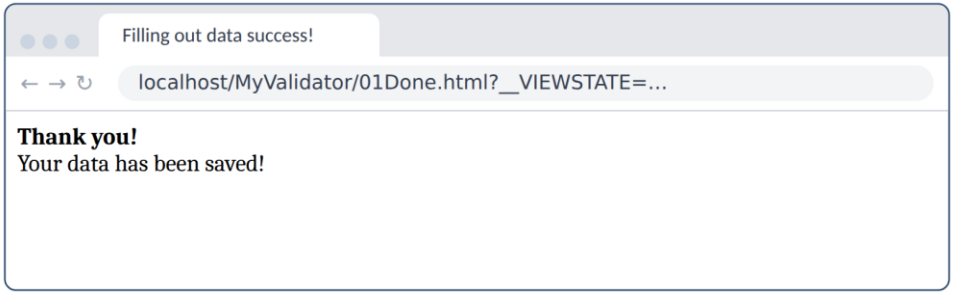
Peramban Menolak



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.

***Gambar 14.2**: dialog milik peramban sendiri, dan tidak ada yang dikirim. Setiap contoh `.aspx` di bab ini dijalankan di Windows; mesin pembangun buku ini adalah Linux. Buku: Subbab 14.2.3.*

State di Alamat



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.

*Formulir memakai GET, sehingga setiap field masuk ke alamat (**Gambar 14.3**), termasuk `__VIEWSTATE`, hidden field yang tidak pernah dideklarasikan halaman. State halaman ikut bepergian bersama permintaan. Buku: Subbab 14.2.3.*

Contoh Terkerjakan 1: Tanpa Peramban Sama Sekali

`tests/http_check.py` menyimpan *cookie* dan *hidden field*, tetapi tidak menjalankan skrip. Program itu mengirim formulir pelanggan yang sepenuhnya salah ke meja depan Razor Pages:

```
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
```

> **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.

422 Unprocessable Content: “dipahami, dan ditolak”. Tempat lahir kembali terisi; kata sandi tidak. Buku: Subbab 14.2.4.

Enam Kontrol Validator

Tabel 14.2 Lima validator Web Forms dan padanannya di Razor Pages; yang keenam, `ValidationSummary`, hanya mendaftar pesan.

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`

*Validator pada **Tabel 14.2** hanya melaporkan ke `Page.IsValid`. Buku: Subbab 14.3.1.*

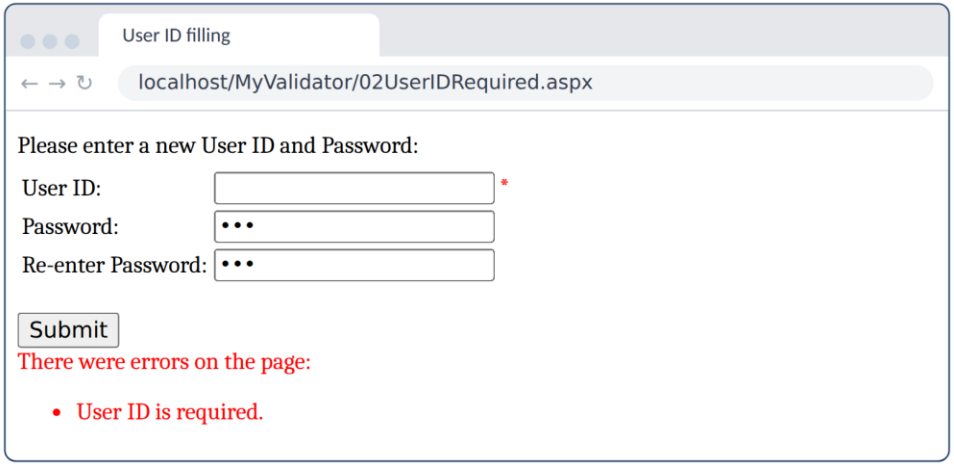
`RequiredFieldValidator`

Kode 14.2 Dari `02UserIDRequired.aspx` (Web Forms, tidak dijalankan di sini): `RequiredFieldValidator` yang teksnya sendiri berupa tanda bintang dan `ErrorMessage`-nya masuk ke `ValidationSummary`.

```
            <td><input type="text" runat="server" id="txtID" />
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
                                            runat="server"
                                            ControlToValidate="txtID"
                                            ErrorMessage="User ID is required.">
                    *
                </asp:RequiredFieldValidator>
    <asp:ValidationSummary ID="ValidationSummary1" runat="server"
                           HeaderText="There were errors on the page:" />
```

*Pada **Kode 14.2**, tanda bintang di antara tag adalah `Text` validator, ditampilkan di samping field; `ErrorMessage` dikumpulkan oleh ringkasan. Validator dapat menjaga `<input runat="server">` biasa juga. Buku: Subbab 14.3.2.*

Id Pengguna yang 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:*.

***Gambar 14.4**: dua tempat untuk satu masalah, tanda di samping field dan kalimat di dalam daftar. Buku: Subbab 14.3.2.*

`CompareValidator`: Tiga Tugas

Kode 14.3 Dari `03PasswordsNotMatch.aspx` (Web Forms, tidak dijalankan di sini): kata sandi kedua harus sama dengan yang pertama.

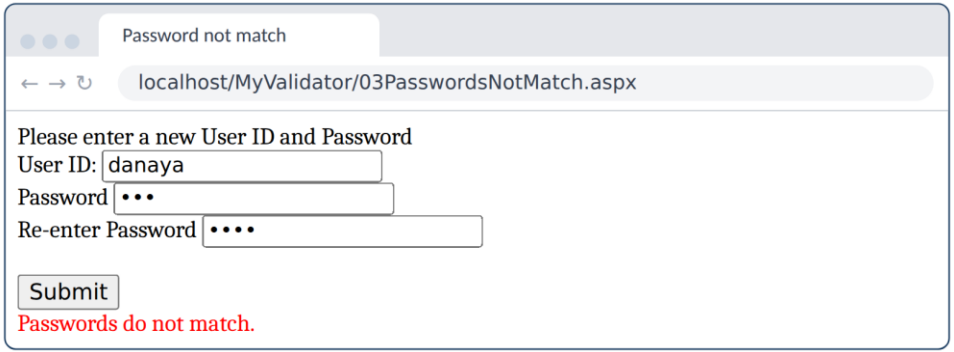
```
        <asp:CompareValidator ID="CompareValidator1" runat="server"
            ControlToValidate="txtRePassword"
            ControlToCompare="txtPassword"
            ErrorMessage="Passwords do not match." />
```

Kode 14.4 Dari `05GreaterThan18.aspx` (Web Forms, tidak dijalankan di sini): perbandingan dengan konstanta, sebagai bilangan bulat.

```
            <asp:CompareValidator id="CompareValidator1" runat="server"
                Operator="GreaterThan" ValueToCompare="18"
                ControlToValidate="TextBox1"
                ErrorMessage="You must be older than 18"
                Type="Integer">
            </asp:CompareValidator>
```

Kode 14.3 *memakai* `ControlToCompare`*;* `Operator="DataTypeCheck"` *memeriksa tipe;* ***Kode 14.4*** *memakai* `ValueToCompare`*. Tanpa* `Type="Integer"`*,* `"9"` *lebih besar daripada* `"18"`*. Jebakan yang sama ada pada* `RangeValidator`*, dan setiap validator kecuali required menerima field kosong. Buku: Subbab 14.3.3 dan 14.3.4.*

Kata Sandi yang Berbeda



Gambar 14.5 Kode 14.3 dengan kata sandi tiga karakter dan ulangan empat karakter: *Passwords do not match.*

***Gambar 14.5**: validator menjaga kotak kedua dan operatornya secara bawaan `Equal`. `RangeValidator` 30 sampai 40 menjawab 17 dengan cara yang sama. Buku: Subbab 14.3.3–14.3.4 dan Gambar 14.6.*

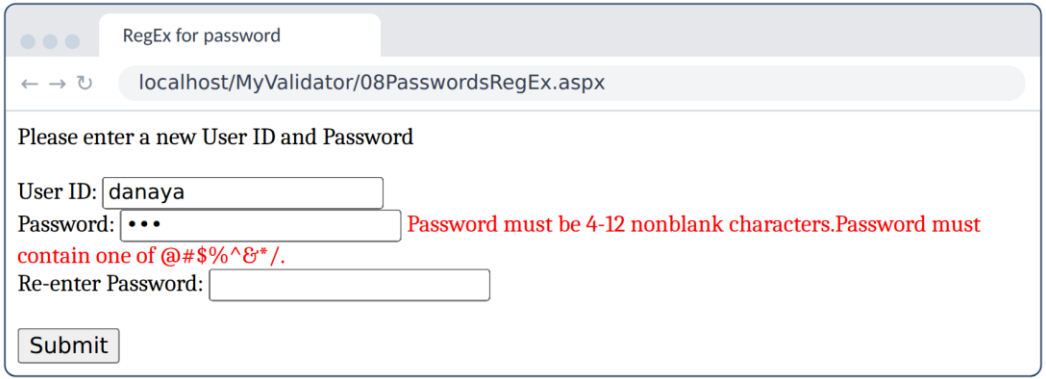
`RegularExpressionValidator`

Kode 14.5 Kotak kata sandi `08PasswordsRegEx.aspx` (Web Forms, tidak dijalankan di sini): dua *regular expression*, dan komentar yang menjelaskan pemeriksaan *required* di sampingnya.

```
            <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. -->
```

*Pada **Kode 14.5**, `[^\s]{4,12}` berarti "empat sampai dua belas karakter bukan spasi" dan `.*[@#$%^&*/].*` berarti "salah satu simbol ini di suatu tempat". Pola harus cocok dengan seluruh nilai. Regular expression memeriksa bentuk, bukan kebenaran: `1234@1234.com` tidak ada. Buku: Subbab 14.3.5.*

Dua Validator Bicara



Gambar 14.6 Kode 14.5 dengan kata sandi `abc`: terlalu pendek dan tidak mengandung simbol yang disyaratkan, sehingga dua validator bersuara.

***Gambar 14.6**: `Display="Dynamic"` tidak menyisakan ruang kosong bagi pesan required, yang tidak punya apa-apa untuk dikatakan. Buku: Subbab 14.3.5.*

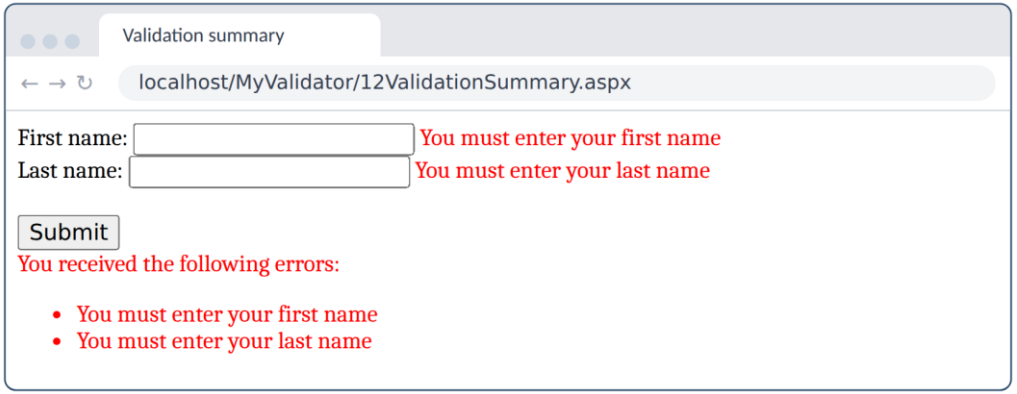
`CustomValidator` di Server

Kode 14.6 `ValidateNumber` dan validatornya dalam `11ServerValidation.aspx` (Web Forms, tidak dijalankan di sini): aturan "habis dibagi 5", dijalankan di server.

```
        void ValidateNumber(object source, ServerValidateEventArgs args) {
            try {
                int num = int.Parse(args.Value);
                args.IsValid = ((num % 5) == 0);
            } catch (Exception ex) {
                args.IsValid = false;
            }
        }
            <asp:CustomValidator id="CustomValidator1"
                runat="server" ControlToValidate="TextBox1"
                ErrorMessage="Number must be divisible by 5"
                OnServerValidate="ValidateNumber">
```

*Versi yang hanya di peramban, `ClientValidationFunction`, meloloskan 13 bila skripnya dilewati: `Page.IsValid` tetap `true`. **Kode 14.6** menolak 3 dan `abc` di server. Rancangan yang benar memasang kedua atribut, dengan aturan yang sama ditulis dua kali. Buku: Subbab 14.3.6 dan Gambar 14.8–14.9.*

`ValidationSummary`



Gambar 14.7 `12ValidationSummary.aspx` setelah pengiriman kosong: setiap validator menampilkan pesannya di samping *field*-nya, dan ringkasan mengulang keduanya di bawah *You received the following errors:*.

***Gambar 14.7**: `DisplayMode` dapat berupa `BulletList`, `List`, atau `SingleParagraph`; `ShowMessageBox="true"` juga menampilkan alert. Ringkasan mengatakan bahwa ada masalah; pesan di field mengatakan apa masalahnya. Buku: Subbab 14.3.7.*

Formulir Laundry, Cara Web Forms

Kode 14.7 Aturan telepon `CustomerNew.aspx` (Web Forms, tidak dijalankan di sini): paruh peramban dalam JavaScript, dan `CustomValidator` yang menyebut kedua paruhnya.

```
        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);
        }
        <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="*" />
```

*Seluruh formulir **Kode 14.7** memakai validator required, range (`Type="Date"`), custom, regular expression, dan compare, masing-masing menampilkan tanda bintang, dengan ringkasan di atas. Kata sandi bersifat opsional: tanpa validator required. Buku: Subbab 14.3.8.*

Kode di Balik Formulir

Kode 14.8 Dari `CustomerNew.aspx.cs` (Web Forms, tidak dijalankan di sini): paruh server aturan telepon, dan tidak ada yang disimpan kecuali `Page.IsValid` bernilai `true`.

```
    // 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);
    }
```

*Tanpa `if (!Page.IsValid) return;` pada **Kode 14.8**, halaman akan menampilkan tanda bintang merah dan tetap menyimpan pelanggan. `Page_Load` mengatur rentang tanggal pada setiap permintaan, karena "sebelum hari ini" berubah setiap hari. Buku: Subbab 14.3.8.*

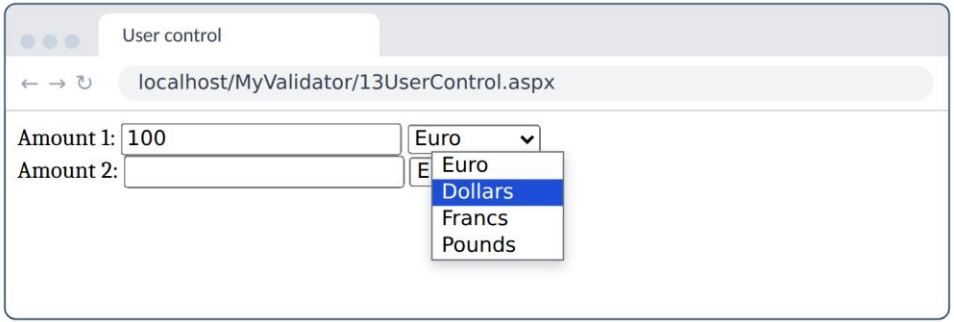
Contoh Terkerjakan 2: Kontrol yang Mengingat

Kode 14.9 `OldRate` dan `Select` dalam `MoneyField.ascx.cs` (Web Forms, tidak dijalankan di sini): kurs mata uang saat ini disimpan di `ViewState`.

```
    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;
```

100 dalam Euro (kurs 1,0), pilih Pounds (0,62): $100/1 \times 0{,}62 = 62$, *dan* `OldRate` *menjadi 0,62. Pilih Dollars (0,88):* $62/0{,}62 \times 0{,}88 = 88$. ***Kode 14.9*** *benar hanya karena* `ViewState["rate"]` *mengingat pound itu. Buku: Subbab 14.3.9.*

Dua *Field* Uang di Satu Halaman



Gambar 14.8 `13UserControl.aspx` di peramban: dua *field* uang yang saling bebas; yang pertama sedang diganti dari Euro ke mata uang lain.

*Pada **Gambar 14.8** setiap instance memegang potongan view state-nya sendiri, dikunci oleh `ID`-nya. Di ASP.NET Core, potongan yang dapat dipakai ulang adalah partial view, tag helper, atau view component, tanpa view state tersembunyi. Buku: Subbab 14.3.9.*

Aturan yang Ditulis sebagai Atribut

Kode 14.10 Dari `CustomerInput.cs`: aturan sebagai *data annotation* pada model yang dikirim formulir.

```
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 = "Enter a mobile phone number.")]
    [IndonesianMobile]
    public string? Phone { get; set; }
    [Compare(nameof(Password), ErrorMessage = "The two passwords do not match.")]
    [DataType(DataType.Password)]
    public string? ConfirmPassword { get; set; }
```

Kode 14.10 *menerjemahkan halaman Web Forms hampir kata demi kata:* `[Required]`*,* `[StringLength]`*,* `[RegularExpression]`*,* `[Compare(nameof(Password))]`*;* `[IndonesianMobile]` *adalah buatan kita. Proyek Razor Pages dijalankan di .NET 10 tanpa paket NuGet. Buku: Subbab 14.4.1 dan 14.4.2.*

Empat Rincian, dan Aturan tentang Hari Ini

Kode 14.11 `Validate` dalam `CustomerInput.cs`: aturan `IValidatableObject` yang memerlukan tanggal hari ini, sebuah `CustomValidator` sisi server.

```
    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) });
        }
    }
```

Properti *nullable*: tanggal kosong menjadi `null` dan `[Required]` bersuara; `DateOnly` biasa diam-diam menjadi `0001-01-01`.

Spasi dianggap kosong, dan **`null` lolos** dari setiap atribut kecuali `[Required]`, sehingga kata sandi opsional berjalan.

Kode 14.11 *menyebut properti pemiliknya, sehingga pesannya muncul di samping tanggal. Buku: Subbab 14.4.2.*

Contoh Terkerjakan 3: Nomor Ponsel Indonesia

Tabel 14.3 Aturan ponsel Indonesia dijalankan dengan tangan pada enam masukan, satu langkah per kolom.

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`
`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	—

***Tabel 14.3**: buang pemisah, `+62`/`62` menjadi `0`, lalu `08[1-9]` dan 7–10 digit. Buku: Subbab 14.4.3.*

Aturannya dalam Satu Kelas

Kode 14.12 Pola dan `Normalize` dalam `Phone.cs`: apa yang diketik menjadi sepuluh sampai tiga belas digit, atau `null`.

```
    // 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;
    }
```

*`[GeneratedRegex]` mengubah pola menjadi C# saat build, sehingga kesalahan pada pola menggagalkan build. **Kode 14.12** mengembalikan digit, bukan `true`/`false`: validasi dan normalisasi adalah satu pekerjaan. `Format` menyimpan `0812-3456-7890`. Buku: Subbab 14.4.3.*

Atribut Validasi Khusus

Kode 14.13 `IndonesianMobileAttribute`: `IsValid` berjalan di server; `AddValidation` menulis dua atribut `data-val-*` agar peramban dapat menjalankan aturan yang sama.

```
[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!);
    }
}
```

*`value is null ||` pada **Kode 14.13** menjaga "wajib" dan "berbentuk benar" sebagai dua aturan dengan dua pesan. `IClientModelValidator` bersifat opsional: ia hanya menambahkan `data-val="true"` dan `data-val-idmobile` pada `<input>`. Buku: Subbab 14.4.3.*

Model Binding dan `ModelState`

Kode 14.14 `OnPost` dalam `New.cshtml.cs`: model yang tidak valid ditampilkan lagi dengan status 422; model yang valid disimpan dan peramban dialihkan.

```
    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
```

*Sebelum `OnPost` berjalan, `[BindProperty]` memicu model binding (teks menjadi tipe) dan model validation (setiap atribut, lalu `Validate`) ke dalam `ModelState`. `ModelState.IsValid` adalah `Page.IsValid` dengan kewajiban yang sama: Anda harus memeriksanya (**Kode 14.14**). Buku: Subbab 14.4.4.*

Tag Helper Menggambar Formulir

Kode 14.15 Dari `New.cshtml`: ringkasan, nama, dan telepon, masing-masing digambar dari model oleh *tag helper*.

```
<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>
    <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>
```

`asp-for` menulis `name`, `id`, `value`, `type`, dan atribut `data-val-*`; **`asp-validation-for`** adalah tempat satu pesan; **`asp-validation-summary="All"`** adalah `ValidationSummary`.

*Setelah POST yang gagal, formulir pada **Kode 14.15** digambar ulang dari model terikat: begitulah Razor Pages "mengingat" apa yang diketik, tugas yang dulu dikerjakan view state di Web Forms. Buku: Subbab 14.4.5.*

Validasi *Unobtrusive*: Aturan sebagai HTML

```
<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="">
```

HTML sebenarnya dari *input* telepon: `data-val-required` dari `[Required]`, `data-val-idmobile` dari atribut kita. `[Compare]` menghasilkan `data-val-equalto`, `[RegularExpression]` menghasilkan `data-val-regex`.

Halaman tidak berisi JavaScript validasi miliknya sendiri: jQuery Validation dan adaptor unobtrusive Microsoft membaca atribut-atribut itu saat halaman dimuat. Tidak ada yang ditulis dua kali dengan tangan, kecuali aturan khusus. Buku: Subbab 14.4.6.

Paruh Peramban `[IndonesianMobile]`

Kode 14.16 `indonesian-mobile.js`: 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);
```

*Pada **Kode 14.16**, `this.optional(element)` bernilai `true` untuk field kosong, sebagaimana `IsValid` untuk `null`. Track PHP, Java, dan Flask sama sekali tidak memiliki validasi sisi klien: server saja sudah cukup untuk kebenaran. Buku: Subbab 14.4.6.*

Dihentikan di Peramban



Gambar 14.9 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.

***Gambar 14.9**: program tangkapan layar menghitung nol POST. Pesan ketujuh dari server, tentang tanggal lahir di 2031, tidak ada di sini: ia hanya hidup di `Validate`. Validasi klien adalah himpunan bagian validasi server, tidak pernah sebaliknya. Buku: Subbab 14.4.6.*

Satu Baris Draf Nota

Kode 14.17 `BillLineInput`: sebuah layanan, unit bulat dari 1 sampai 999, dan rak dari `A01` sampai `C20`. Tidak ada harga.

```
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?` di **Kode 14.17** menyatakan dua aturan bisnis: unit adalah bilangan bulat (3,2 kg dibulatkan ke atas menjadi 4) dan positif. Tiga aturan lain memerlukan basis data dan ditambahkan di handler dengan `ModelState.AddModelError`. Buku: Subbab 14.4.7.*

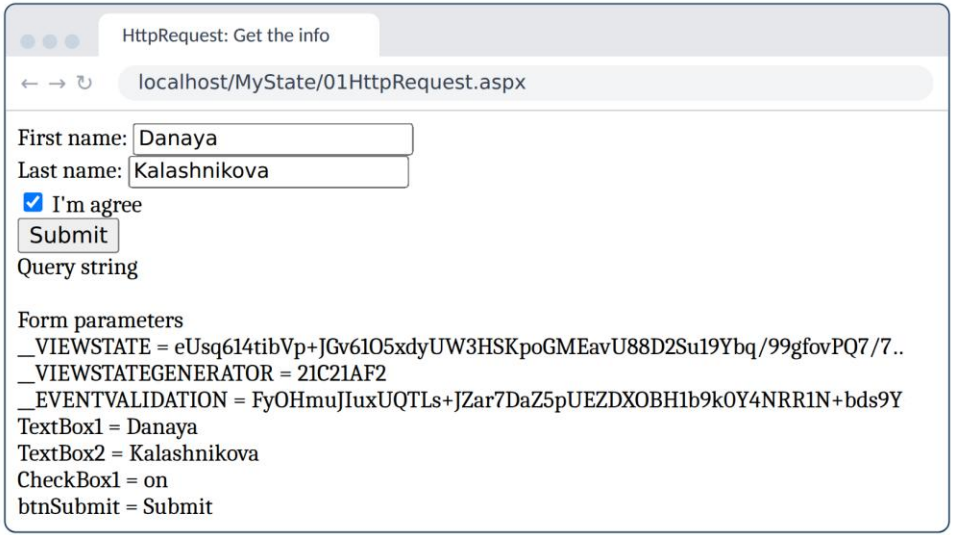
Contoh Terkerjakan 4: Aturan yang Memerlukan Server

```
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.
```

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.

`2.5` gagal saat binding; `S0099`, rak B08 di O01, dan B09 kedua adalah data kontradiktif: berbentuk benar sendiri-sendiri, salah hanya dalam terang apa yang diketahui server. Buku: Subbab 14.4.7.

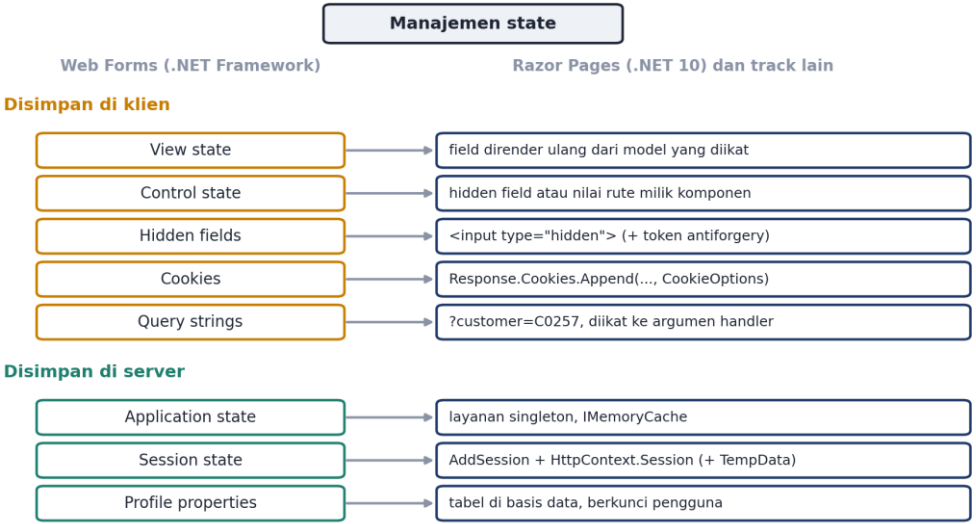
HTTP Lupa



Gambar 14.10 Parameter formulir dari satu *postback*: tiga *field* yang diisi orang, tombol yang ditekan, dan tiga *hidden field* yang ditambahkan Web Forms, `__VIEWSTATE`, `__VIEWSTATEGENERATOR`, dan `__EVENTVALIDATION`.

*Setiap permintaan bertemu instance halaman yang baru, yang dibuang setelah jawabannya dikirim. **Gambar 14.10** adalah seluruh rahasia "ingatan" Web Forms: state ditulis ke jawaban terakhir sebagai hidden field lalu dikirim balik. Tidak ada apa pun di server yang mengingat. Buku: Subbab 14.5.1.*

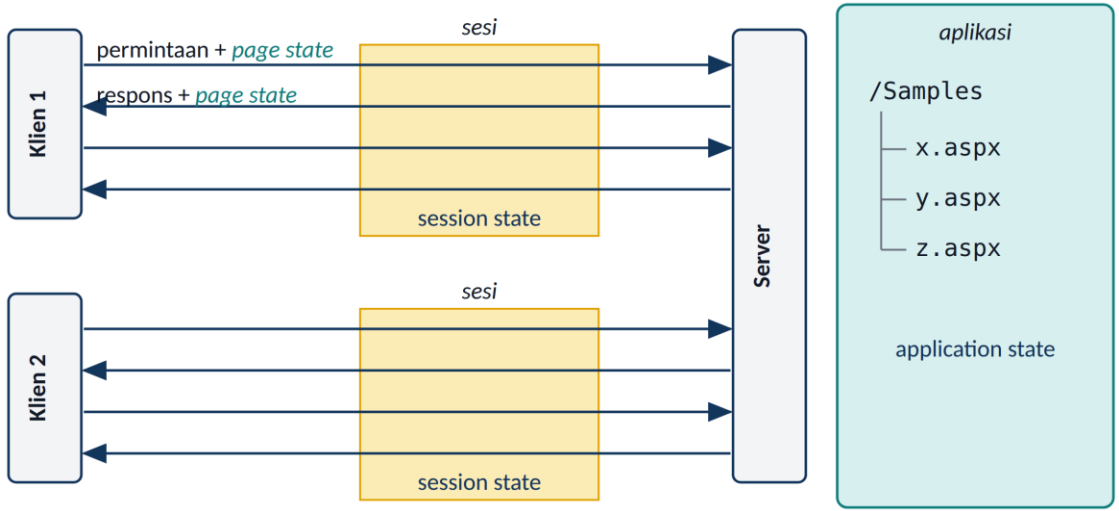
Peta Mekanisme



Gambar 14.11 Pilihan manajemen *state* ASP.NET Web Forms, dikelompokkan menurut tempat data disimpan, dengan alat yang mengerjakan tugas yang sama di Razor Pages sekarang.

***Gambar 14.11**: view state, control state, hidden field, cookie, dan query string menyimpan data di klien; application state, session state, dan profile properties di server. Pengalihan dengan `?name=` yang dicetak lewat `<%= %>` juga celah XSS: pakailah `<%: %>`. Buku: Subbab 14.5.2 dan 14.5.3.*

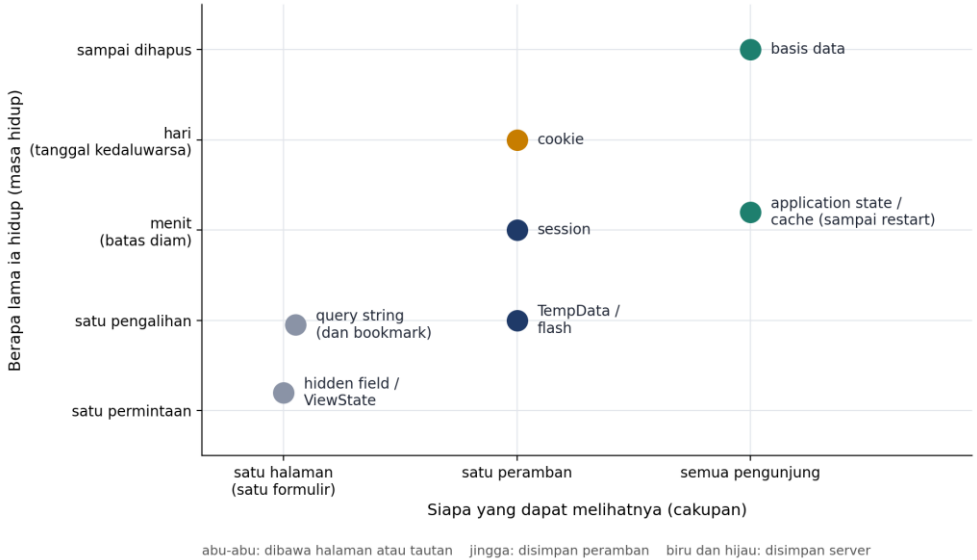
Halaman, Sesi, Aplikasi



Gambar 14.12 *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.

***Gambar 14.12**: **view state** tinggal di hidden field satu halaman; **session state** adalah data global untuk satu pengguna, dihapus setelah 20 menit tanpa aktivitas; **application state** adalah data global untuk semua pengguna, dihapus saat restart. Buku: Subbab 14.5.3.*

Siapa yang Melihatnya, dan Berapa Lama?



Gambar 14.13 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.

*Pada **Gambar 14.13**: draf nota milik satu peramban selama beberapa menit (sesi); gerai konter milik satu komputer selama berminggu-minggu (cookie); "Pelanggan terdaftar" bertahan satu pengalihan (TempData); daftar harga milik semua orang (cache); nota bertahan bertahun-tahun (basis data). Buku: Subbab 14.5.3 dan Tabel 14.4.*

View State, Dilihat dari Peramban

view-source:localhost/MyState/05HiddenField.aspx

```
<!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><title>
    Hidden Field
</title></head>
<body>
    <form method="post" action="./05HiddenField.aspx" id="form1">
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="eov/0rX
 D7PB6hN0Irwk…" />
</div>

<div class="aspNetHidden">
    <input type="hidden" name="__VIEWSTATEGENERATOR" id="__VIEWSTATEGEN
 ERATOR" value="ED5EA427" />
    <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATIO
 N" value="VrW179/mmj8ozmI…" />
</div>
    <div>
            
        <input type="submit" name="Button1" value="No Action Button" id
 ="Button1" /> 
        <input type="hidden" name="HiddenField1" id="HiddenField1" valu
 e="2" />
    </div>
        <strong>         <br />
           Click counter</strong> (Saved in Hidden Field) :&n
 bsp;
        <span id="Label1">2</span>
    </form>
</body>
</html>
```

Gambar 14.14 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`.

*Pada **Gambar 14.14**: view state adalah Base64, **tidak dienkripsi** secara bawaan tetapi dilindungi MAC dari pengubahan; ia memakan bandwidth dua kali; ia dapat dimatikan per mesin, aplikasi, halaman, atau kontrol. Control state tetap hidup saat view state dimatikan. Razor Pages sama sekali tidak memilikinya. Buku: Subbab 14.6.1 dan 14.6.2.*

Hidden Field dan *Query String*

Hidden field ikut dalam formulir POST, dan siapa pun dapat mengubahnya. `customer` tersembunyi milik draf diperiksa terhadap sesi:

```
tampered hidden customer    -> /: This draft has expired. Please open it again.
```

Query string dapat ditandai dan dibagikan: `/bills/draft?customer=C0257`, diperiksa terhadap tabel pelanggan. Tidak ada harga di formulir mana pun (aturan BR-1).

> **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`.

Setiap formulir POST Razor Pages juga membawa `__RequestVerificationToken`, token antiforgery. Buku: Subbab 14.6.3 dan 14.6.4.

Atribut Sebuah *Cookie*

Tabel 14.4 Atribut-atribut *cookie*, dan *cookie* gerai Laundry.

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`

***Tabel 14.4**: `Secure` mencegah penyadapan di Wi-Fi terbuka, `HttpOnly` mencegah skrip sisipan, `SameSite` mencegah POST lintas situs membawa cookie Anda. Dengan `Expires` atau `Max-Age`, cookie bertahan melewati restart; tanpanya, ia mati bersama peramban. Buku: Subbab 14.6.5.*

Cookie Gerai

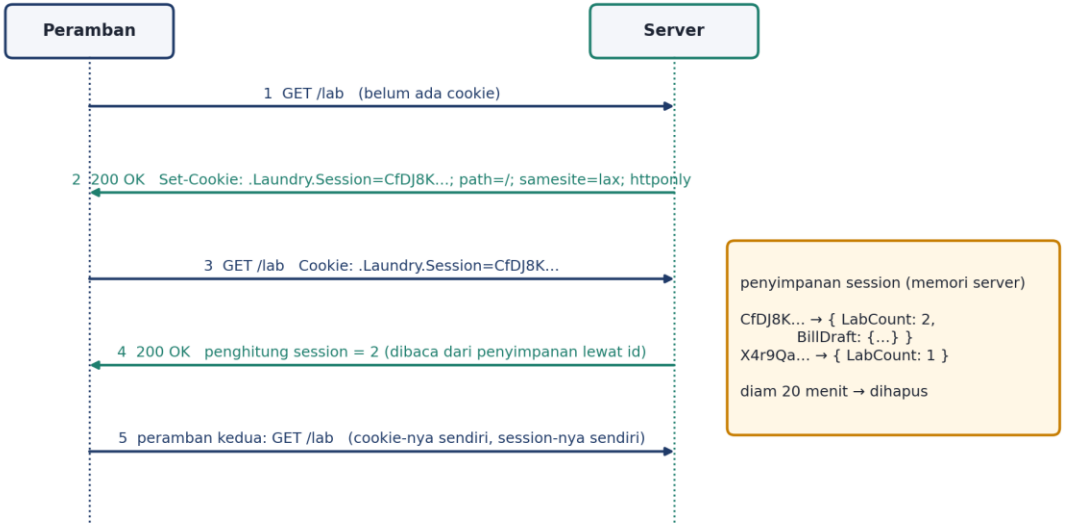
Kode 14.18 `Read` dan `Write` dalam `OutletCookie.cs`: 30 hari, `HttpOnly`, `SameSite=Lax`, `Secure` di HTTPS; diterima hanya bila menyebut gerai yang nyata.

```
    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 = "/",
        });
```

*Gerai adalah nilai cookie yang sempurna: milik perangkat, hidup berminggu-minggu, tidak rahasia, dan diperiksa setiap kali dibaca (**Kode 14.18**). Peramban menerima `Set-Cookie: outlet=O02; max-age=2592000; path=/; samesite=lax; httponly`. Buku: Subbab 14.6.5.*

Cara Kerja Sesi



Gambar 14.15 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.

***Gambar 14.15**: peramban hanya memegang kunci acak yang tak dapat ditebak; server memegang datanya. Permintaan tanpa kunci yang dikenal mendapat sesi baru yang kosong; sesi yang menganggur dihapus. Id sesi adalah kredensial. Buku: Subbab 14.7.1.*

Sesi di ASP.NET Core

Kode 14.19 `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));
}
```

*`AddDistributedMemoryCache()`, `AddSession(…)`, dan `app.UseSession()` sebelum `MapRazorPages()`. Sesi menyimpan byte dan string, sehingga **Kode 14.19** memakai JSON, dan objek yang diubah harus di-`Set` lagi, atau perubahannya hilang. Buku: Subbab 14.7.2.*

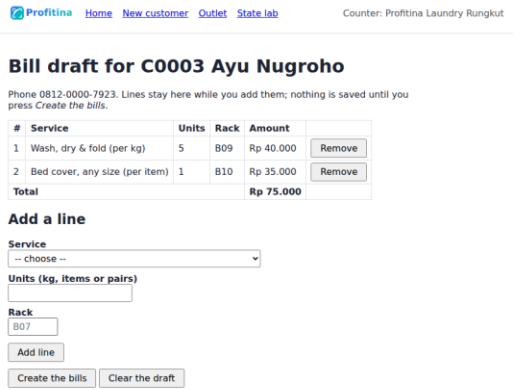
Draf Nota di dalam Sesi

Kode 14.20 Dari `OnPost` dalam `Draft.cshtml.cs`: pelanggan tersembunyi harus sesuai dengan sesi, dan tiga aturan yang memerlukan basis data ditambahkan ke `ModelState`.

```
    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("/");
        }
                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.");
```

*Draf bekerja seperti keranjang belanja. Pada **Kode 14.20**, tab yang basi, sesi yang kedaluwarsa, atau field yang diubah ditolak. Setelah perubahan, handler menulis draf kembali lalu mengalihkan; `Confirm` menghitung harga setiap baris saat itu dari daftar harga dan membuang draf. Buku: Subbab 14.7.3.*

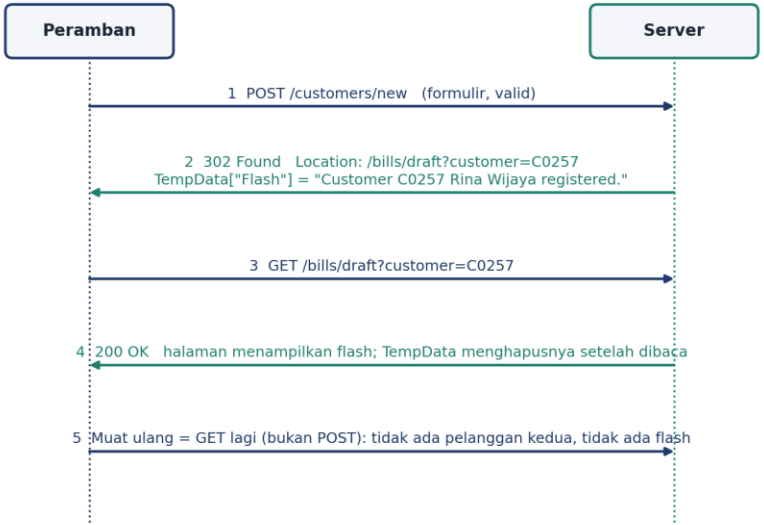
Draf di Layar



Gambar 14.16 Draf nota untuk pelanggan C0003 di gerai O01 (Rungkut): dua baris di rak B09 dan B10, jumlahnya dihitung server, dan totalnya Rp 75.000.

***Gambar 14.16**: jumlahnya dihitung server; kasir dengan developer tools tidak dapat memberi temannya potongan harga. Buku: Subbab 14.7.3.*

Post/Redirect/Get dengan Pesan *Flash*



Gambar 14.17 Post/Redirect/Get dengan pesan *flash*. POST menyimpan pesan dan menjawab 302; GET berikutnya menampilkan pesan lalu menghapusnya; memuat ulang hanya mengulang GET.

***Gambar 14.17**: halaman yang digambar langsung setelah POST akan membuat tombol muat ulang menghasilkan pelanggan C0258. GET memerlukan pesannya, dan HTTP lupa: **TempData** menyimpannya sampai dibaca sekali. Buku: Subbab 14.7.4.*

TempData, Diperiksa

```
confirm                    302 -> /
  flash                    Created B2954, B2955 for C0257 Rina Wijaya: Rp 75.000.
  reload                   flash gone
```

Dua nota, karena baris ketiga sudah dihapus; B2954 dan B2955 melanjutkan nota terakhir data contoh, B2953; muat ulang tidak menampilkan apa-apa.

Di sini TempData tinggal di sesi (`AddSessionStateTempDataProvider()`); Flask menyebutnya `flash()`, sedangkan PHP dan Java menyimpan sebuah daftar di sesi.

Layout meng-escape pesan dengan `@`, dan pesan hanya berisi nomor nota, nama, dan total, tidak pernah telepon atau alamat. Buku: Subbab 14.7.4.

Application State dan *Cache*

Kode 14.21 `LabCounter`: *application state* sebagai *singleton*.

```
/// <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);
}
```

Kode 14.22 `PriceList`: daftar harga, di-*cache* selama sepuluh menit.

```
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);
}
```

***Kode 14.21**: `Interlocked` adalah `Application.Lock()` versi modern. **Kode 14.22**: cache hanya data publik. Buku: Subbab 14.7.5.*

Contoh Terkerjakan 5: Lab *State*

```
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
```

app naik pada setiap permintaan dari setiap peramban; **session** dan **cookie** mulai lagi dari 1 untuk peramban B; **hidden** hanya bergerak pada POST; **query** hanya ketika tautan diikuti.

Setelah restart, app dan session mulai lagi dari 1, sedangkan penghitung cookie melanjutkan, karena peramban menyimpannya. Soal Latihan 14.8 memainkan lab ini dengan dua peramban. Buku: Subbab 14.7.6 dan 14.7.7.

Satu Meja Depan, Empat *Track*

Tabel 14.5 Keempat *track* bab ini, tugas demi tugas.

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`

***Tabel 14.5**: tugasnya sama, hanya namanya berubah. Buku: Subbab 14.8–14.11.*

Sesi Flask Adalah *Cookie*-nya

```
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": []
  }
}
```

Flask menandatangani seluruh `session` dengan `SECRET_KEY` dan mengirimnya sebagai *cookie*: tahan ubah, seperti MAC *view state*, tetapi **dapat dibaca**, di sini diurai tanpa kunci apa pun.

Cukup untuk id dan nomor rak; keliru untuk telepon, kata sandi, atau rahasia. PHP mengubah `Input.Name` menjadi `$_POST['Input_Name']`; Java menulis penyimpanan sesinya sendiri. Buku: Subbab 14.8–14.10.

Keamanan dan Data Pribadi

Id sesi adalah kredensial: panjang dan acak, hanya di *cookie* `HttpOnly`, `Secure`, `SameSite`, kedaluwarsa saat menganggur, diperbarui saat *login*.

CSRF: `SameSite=Lax` ditambah token *antiforgery* di sesi dan *hidden field*; Razor Pages mengerjakan keduanya secara otomatis.

Jangan memercayai *state* klien; *escape* saat keluaran; *hash* kata sandi (PBKDF2-HMAC-SHA256, 600.000 iterasi), jangan pernah menampilkan atau mencatatnya.

UU PDP No. 27/2022: tujuan, minimisasi, akurasi, keamanan, retensi terbatas, hak subjek data, transparansi; memilih mekanisme *state* berarti memilih di mana data pribadi tinggal.

Data pribadi di sesi lenyap setelah 20 menit tanpa aktivitas; di cookie persisten ia menetap di komputer konter bersama selama berminggu-minggu. Buku: Subbab 14.12 dan Tabel 14.7.

Profitina dalam Praktik

Validasi di tepi, selalu di server: setiap alamat yang dipindai, setiap kiriman, setiap tautan komunitas (penyaring *Link Safety*).

***State* dipilih menurut cakupan dan masa hidup:** siapa yang masuk di sesi, tren skor mingguan di basis data, batas per *tier* di aplikasi.

Antarmuka optimistis, server yang jujur: kiriman muncul seketika, dan dipulihkan bila server menolak.

Data pribadi sebagai fitur: *hash* Argon2id, persetujuan eksplisit, *login* yang dibatasi lajunya, ekspor dan penghapusan akun.

Studi kasus yang dapat dijalankan tetap Profitina Laundry; yang terbawa adalah bentuk disiplinnya. Buku: Subbab 14.13.

Kesalahan yang Sering Terjadi

Memvalidasi hanya di peramban; melupakan `Page.IsValid` atau `ModelState.IsValid`; validator *compare* atau *range* tanpa `Type`.

Mengandalkan pola untuk menolak masukan kosong; properti masukan yang tidak *nullable*; memercayai *hidden field*, *cookie*, atau *query string*.

`<%= Request.QueryString["name"] %>`; menampilkan halaman langsung setelah POST yang berhasil; mengubah objek sesi tanpa `Set`.

`UseSession()` **sesudah** `MapRazorPages()`; `count++` pada *state* bersama; rahasia di sesi Flask, *cookie*, atau *view state*.

***Cookie* tanpa** `HttpOnly`, `Secure`, `SameSite`; mengharapkan `Input.Name` di PHP; id sesi di dalam alamat.

Setiap kesalahan disertai perbaikannya di buku. Buku: Subbab 14.14.

Ringkasan: Lima Hal untuk Diingat

1. Data buruk bersifat tak berguna, tak terautentikasi, atau kontradiktif (**Tabel 14.1**); peramban adalah kesopanan, server adalah hukum.

2. Validator Web Forms (**Tabel 14.2**) dan *data annotation* Razor Pages (**Kode 14.10**) menyatakan aturan yang sama; `Page.IsValid` dan `ModelState.IsValid` wajib diperiksa.

3. Atribut khusus berjalan di kedua sisi (**Kode 14.13**, **Kode 14.16**); aturan yang memerlukan basis data tinggal di *handler*.

4. HTTP lupa; pilihlah mekanisme menurut cakupan dan masa hidup (**Gambar 14.13**); *cookie* memerlukan `HttpOnly`, `Secure`, `SameSite` (**Tabel 14.4**).

5. Sesi menyimpan data di server di balik id acak (**Gambar 14.15**); Post/Redirect/Get dan TempData membuat muat ulang tidak berbahaya (**Gambar 14.17**).

Buku: Subbab 14.15; Soal Latihan 14.1–14.12 beserta penyelesaiannya di Subbab 14.16 dan 14.17.

Pertemuan Berikutnya

Bab 15 — ADO.NET dan Lapisan Data. Bagaimana program .NET berbicara dengan basis data, dari *connection string* pertama sampai `Commit()` terakhir.

Kerja terhubung (koneksi, perintah, parameter, DataReader) dan terputus (DataAdapter, DataSet, DataTable); transaksi; *repository* dalam lima *track*.

Sebelum itu: kerjakan Soal Latihan 14.1, 14.3, 14.8, dan 14.11.

`SqliteLaundryStore` *hari ini menunggu sebuah paket NuGet; Bab 15 membahas persis lapisan itu. Buku: Bab 15.*

Kuliah 15 — ADO.NET dan Lapisan Data

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 15

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini kita membuka kotak yang selama ini dianggap sudah ada oleh setiap halaman: dari mana data datang, dan bagaimana beberapa operasi tulis menjadi satu tindakan. Buku: Bab 15, Subbab 15.1–15.20.

Cakupan Pertemuan Ini

1. **Apa itu ADO.NET** — satu antarmuka, satu penyedia per basis data, dan dua cara bekerja: terhubung dan terputus.
2. **Model terhubung** — koneksi, perintah, parameter, DataReader; buka selambat mungkin, selalu tutup.
3. **Transaksi** — sebuah nota, log statusnya, dan pembayaran pertamanya sebagai satu tindakan; tingkat isolasi.
4. **Model terputus** — DataSet, keadaan dan versi baris, DataView, DataAdapter.
5. ***Stored procedure* dan *SQL injection*** — satu tempat untuk sebuah unit kerja; perekatan diperagakan dan dicegah.
6. **Lapisan data** — satu antarmuka *repository*, diimplementasikan dalam C#, PHP, Java, dan Python; EF Core.

Tiga pertanyaan menata bab ini: apa itu ADO.NET dan mengapa ia punya dua paruh; bagaimana beberapa operasi tulis menjadi satu tindakan; di mana SQL seharusnya tinggal dan siapa yang boleh memasukkan teks ke dalamnya. Buku: Subbab 15.1.

Posisi Kita Saat Ini

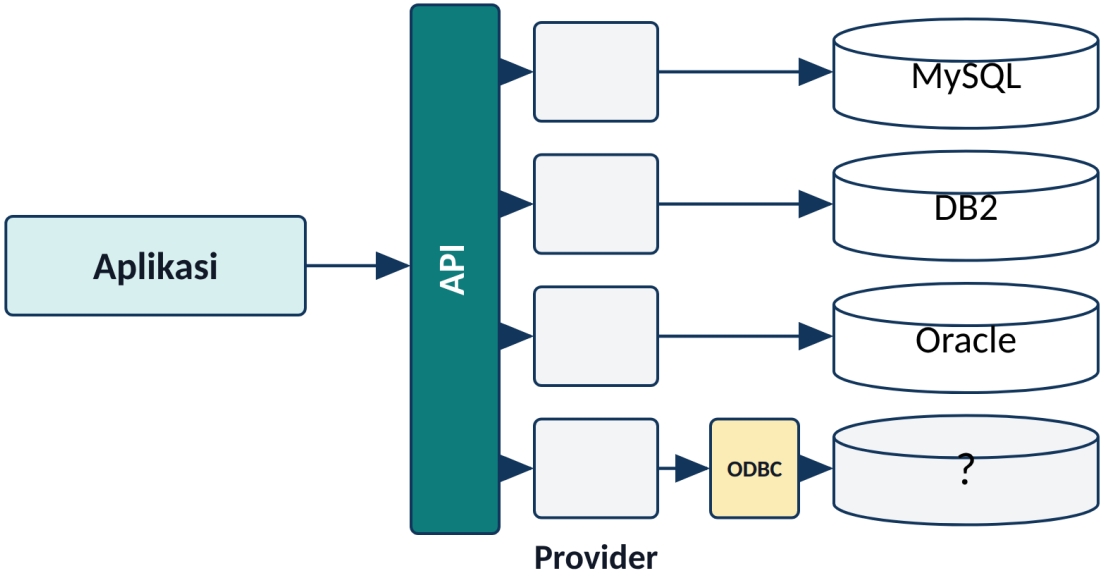
Bab 5 mengajarkan SQL dan PDO; Bab 7 JDBC dan *Data Access Object*; Bab 13 dan 14 menampilkan data di halaman dan menyimpan *state* di antara permintaan. Di semua bab itu halaman-halaman begitu saja *memiliki* datanya.

Apa yang dijalankan di sini: seluruh isi `System.Data` berjalan di .NET 10 SDK; penyedia SQLite, MySQL, dan SQL Server serta EF Core adalah paket NuGet dan tidak dijalankan. Lapisan PHP, Java, dan Python berjalan terhadap basis data Laundry yang sebenarnya.

> **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.

Segala hal tentang koneksi, perintah, reader, DataSet, dan transaksi tinggal di dalam lapisan itu, sehingga semuanya dapat diubah tanpa menyentuh satu halaman pun. Buku: Subbab 15.1 dan 15.1.1.

Akses Data Universal



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.

*Pada **Gambar 15.1** aplikasi hanya berbicara dengan API; setiap penyedia menerjemahkan untuk satu basis data; jembatan seperti ODBC menjangkau sumber yang tidak memiliki penyedia sendiri. Buku: Subbab 15.2.1.*

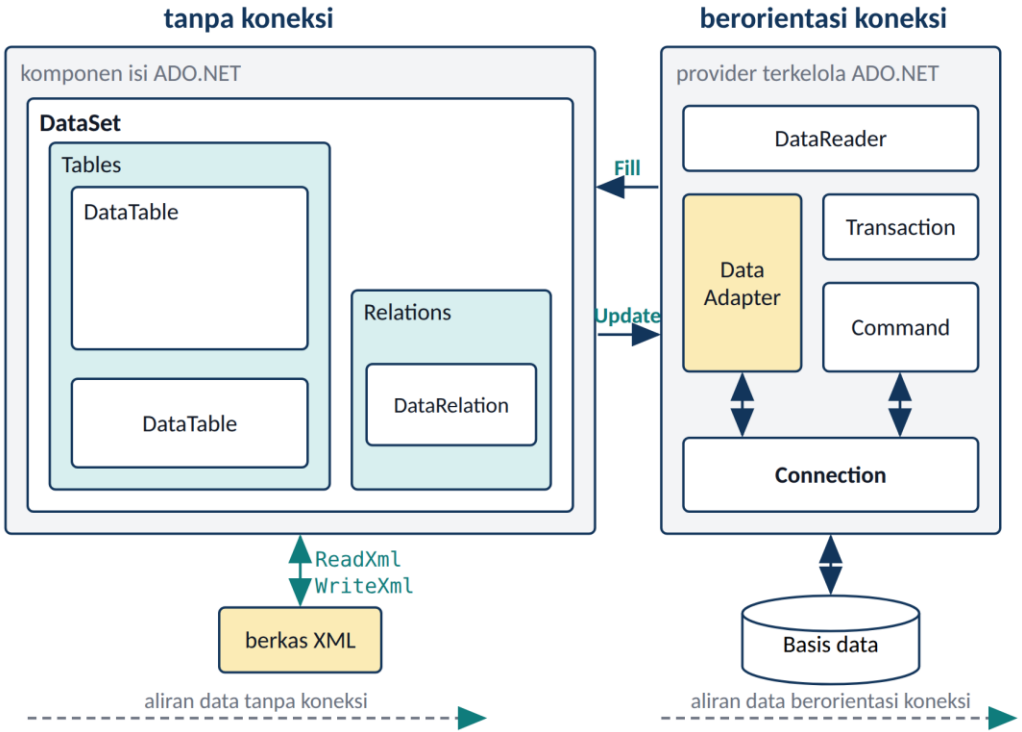
Apa Itu ADO.NET

> **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.

Leluhur: ODBC (1992, antarmuka C), OLE DB (1996, COM), ADO (1996, Recordset). ADO.NET (2002) adalah rancangan ulang, bukan ADO versi baru: ia menambahkan DataSet tanpa koneksi dan XML.

Connection string yang diawali `provider=SQLOLEDB` *adalah string OLE DB: ketiga leluhur itu masih akan Anda jumpai di kode lama. Buku: Subbab 15.2.1–15.2.2 dan Tabel 15.1.*

Dua Paruh


tanpa koneksi
berorientasi koneksi
komponen isi ADO.NET
DataSet
Tables
DataTable
DataTable
Relations
DataRelation
Fill
Update
provider terkelola ADO.NET
DataReader
Data Adapter
Transaction
Command
Connection
ReadXml
WriteXml
berkas XML
Basis data
aliran data tanpa koneksi
aliran data berorientasi koneksi

Gambar 15.2 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.

*Simpanlah **Gambar 15.2** di kepala untuk sisa pertemuan hari ini: penyedia terkelola di satu sisi, DataSet di sisi lain, dan `Fill` serta `Update` milik DataAdapter sebagai satu-satunya jembatan. Buku: Subbab 15.2.3.*

Terhubung dan Tanpa Koneksi

DEFINISI. Dalam model **berorientasi koneksi** program membiarkan koneksi tetap terbuka selama bekerja dan membaca hasilnya baris demi baris melalui DataReader. Dalam model **tanpa koneksi** sebuah DataAdapter membuka koneksi sebentar untuk *mengisi* DataSet di memori utama, program bekerja pada DataSet selama yang ia mau, lalu *adapter* menyambung sebentar lagi untuk *memperbarui* sumbernya.

Lamanya setiap model memegang koneksi adalah Persamaan (15.1):

$$T_{\text{connected}} = t_{\text{query}} + t_{\text{use}}, \qquad T_{\text{disconnected}} = t_{\text{fill}} + t_{\text{update}} \tag{15.1}$$

Koneksi itu langka dan dikelola dalam pool, sehingga model yang memegangnya lebih singkat melayani lebih banyak pengunjung. Di halaman web t_{use} sangat kecil bila baris dibaca ke dalam list lebih dulu. Buku: Subbab 15.2.3.

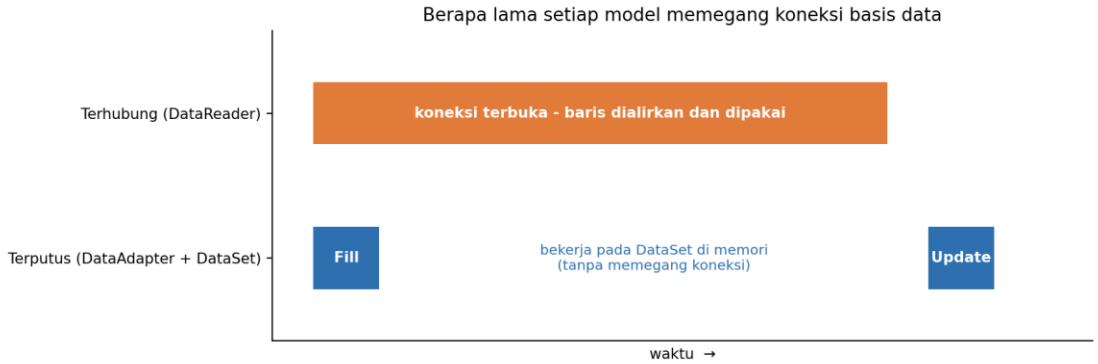
Model yang Mana untuk Tugas yang Mana

Tabel 15.1 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

***Tabel 15.1**: baca sekali dari baris pertama sampai terakhir dengan DataReader; pakailah DataSet untuk menyimpan sementara, merelasikan, mengikat, atau mengolah data tanpa memegang koneksi. DataAdapter sendiri memakai DataReader di dalam `Fill`. Buku: Subbab 15.2.3 dan 15.2.4.*

Berapa Lama Koneksi Dipegang



Gambar 15.3 Berapa lama setiap model memegang koneksi basis data. Model terhubung memegangnya selama baris-baris dipakai; model terputus hanya selama mengisi dan menulis kembali.

***Gambar 15.3**: reader memegang koneksinya dari baris pertama sampai terakhir, termasuk rendering apa pun di perulangan yang sama; adapter memegangnya hanya selama `Fill` dan `Update`. Buku: Subbab 15.2.3.*

Penyedia Kini Berupa Paket

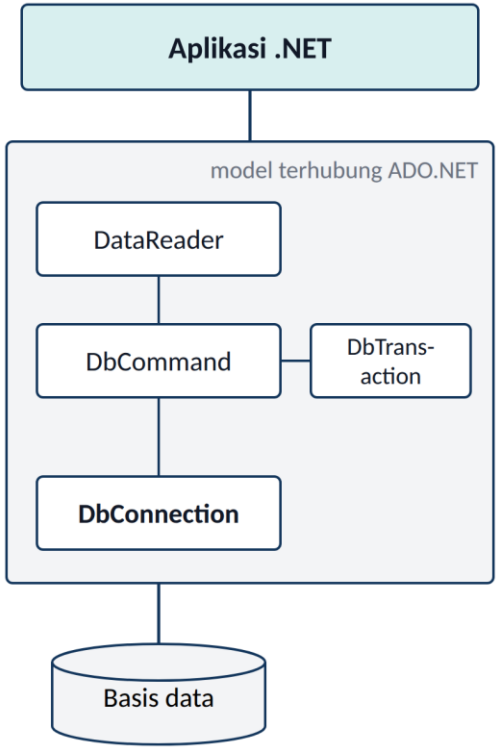
```
dotnet add package Microsoft.Data.Sqlite          # SQLite: the Laundry database file
dotnet add package MySqlConnector                  # MySQL 8 or MariaDB (XAMPP)
dotnet add package Microsoft.Data.SqlClient        # SQL Server, SQL Server Express, Azure SQL
dotnet add package Microsoft.EntityFrameworkCore.Sqlite   # EF Core on SQLite (Section 15.16)
```

`System.Data` dan `System.Data.Common` tetap di *framework*; `System.Data.SqlClient` menjadi `Microsoft.Data.SqlClient`; penyedia SQLite, MySQL, dan PostgreSQL adalah paket NuGet.

> **PERINGATAN:** `sa` (SQL Server) dan `root` (MySQL) dapat melakukan apa saja terhadap setiap basis data. Aplikasi harus menyambung dengan *login* yang hanya dapat menjangkau satu basis data dan hanya melakukan apa yang dilakukan aplikasi.

SQL Server Developer dan Express gratis; SSMS mengelolanya di Windows; Northwind adalah contoh klasiknya. Kode Laundry tidak memerlukan server: salinan SQLite dan CSV dari basis datanya. Buku: Subbab 15.2.5, 15.3, dan Tabel 15.3.

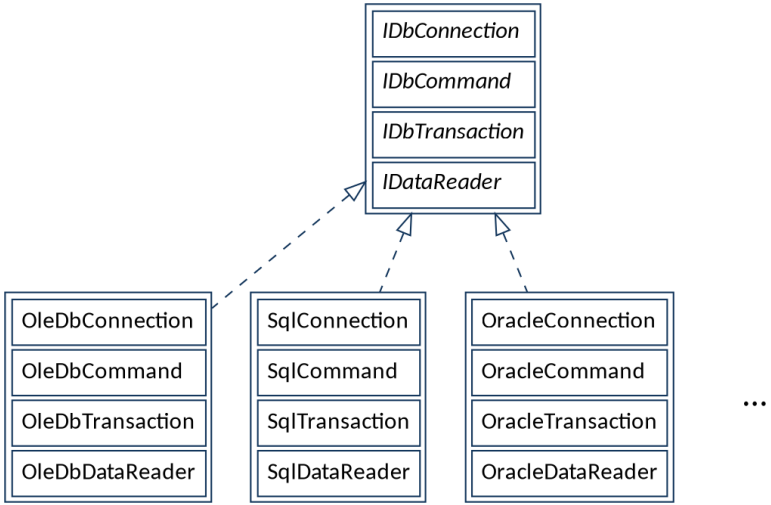
Model Terhubung



Gambar 15.4 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.

***Gambar 15.4**: masing-masing berupa antarmuka (`IDbConnection`), kelas per penyedia (`SqlConnection`), dan kelas dasar `Db`…. Buku: Subbab 15.4.1.*

Antarmuka dan Penyedia



Gambar 15.5 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.

***Gambar 15.5**: memprogram terhadap antarmuka, atau terhadap kelas dasar Db..., itulah yang membuat ADO.NET "universal"; hanya baris yang membuat koneksi yang menyebut nama penyedia. Buku: Subbab 15.4.1.*

Pola Akses Terhubung

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
```

Pseudokode 15.1*: koneksi yang tidak ditutup tidak kembali ke pool; halaman yang membocorkan satu koneksi per galat, pada hari yang buruk, akan menghabiskan pool dan menghentikan seluruh situs. Buku: Subbab 15.4.2.*

EmployeesReader, seperti Dulu Ditulis

Kode 15.1 Isi blok `try` dalam `EmployeesReader.OleDb.cs`: buat, buka, perintah, *reader*, perulangan. Tidak dijalankan dalam buku ini (Windows, SQL Server, dan OLE DB).

```
            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();
            }
```

*Pada **Kode 15.1** setiap variabel bertipe antarmuka kecuali `new OleDbConnection(connStr)`; `CreateCommand()` mengembalikan perintah milik penyedia koneksi itu sendiri; `GetValues` mengisi larik seukuran `FieldCount`. Blok `finally` menutup koneksi. Buku: Subbab 15.4.3 dan Gambar 15.14.*

EmployeesReader, seperti Ditulis Sekarang

Kode 15.2 `EmployeesReader.SqlClient.cs`: .NET 10, `await using` yang menutup *reader*, perintah, dan koneksi, serta *connection string* dari lingkungan. Tidak dijalankan dalam buku ini.

```
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)}");
}
```

***Kode 15.2**: `await using` mengerjakan tugas blok `finally`; panggilan `…Async` membebaskan thread selagi basis data bekerja; kata sandi tinggal di lingkungan atau `dotnet user-secrets`, tidak pernah di kode sumber. Buku: Subbab 15.4.3.*

Connection String

Tabel 15.2 *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`

Jangan pernah ada kata sandi di kode sumber; **bangun *string* dengan kelas *builder***, bukan penyambungan; **biarkan *pooling* menyala**, sehingga "buka selambat mungkin, tutup secepat mungkin" hanya berbiaya mikrodetik.

***Tabel 15.2**: pasangan `kunci=nilai` yang dipisahkan titik koma; kuncinya berbeda antarpenyedia. Buku: Subbab 15.4.4.*

Tiga Cara Menjalankan Perintah

Tabel 15.3 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`

*Pada **Tabel 15.3**, bilangan yang dikembalikan `ExecuteNonQuery` layak diperiksa: 0 dari `UPDATE … WHERE id = @id AND status = 'queued'` berarti ada orang lain yang mengubah baris itu lebih dulu. Buku: Subbab 15.4.5.*

Parameter

> **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.

Sebuah parameter memiliki **nama**, **nilai**, **tipe** (`DbType`), dan **arah**: `Input`, `Output`, `InputOutput`, `ReturnValue`.

Penanda: `@name` untuk SqlClient, Sqlite, dan MySqlConnector; `?` menurut posisi untuk OLE DB, ODBC, dan JDBC; `?` atau `:name` di PDO dan Python.

Tiga langkah: penanda di SQL, satu objek parameter per penanda, nilai diberikan sebelum eksekusi. Buku: Subbab 15.4.6 dan Tabel 15.6.

Satu Pembantu untuk Setiap Perintah

Kode 15.3 `AdoBillRepository.Command`: setiap nilai menjadi `DbParameter`, dan `null` menjadi `DBNull.Value`, NULL milik basis data.

```
    /// <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;
    }
```

Kode 15.3 *tidak menyebut penyedia mana pun:* `CreateCommand()` *dan* `CreateParameter()` *mengembalikan jenis milik koneksi itu sendiri.* `null` *di C# berarti "tidak ada objek"; NULL basis data adalah nilai tersendiri,* `DBNull.Value`*.* `Scalar` *dan* `Execute` *membungkus dua metode eksekusi lainnya. Buku: Subbab 15.4.6.*

DataReader

- **ExecuteReader()** mengembalikan objek DataReader

```
IDataReader ExecuteReader()
IDataReader ExecuteReader(CommandBehavior behavior);
```

- DataReader membaca hasilnya secara berurutan, baris demi baris



Gambar 15.6 DataReader atas tabel hasil sebuah `SELECT`. `Read()` memajukan kursor satu baris dan melaporkan apakah masih ada baris yang dapat dituju.

***Gambar 15.6**: kursor yang mulai sebelum baris pertama; `Read()` maju dan melaporkan apakah ada baris. Ia tidak dapat mundur atau melompat. Indexer mengembalikan `object`; getter bertipe mengonversi; `IsDBNull` bertanya lebih dulu. Buku: Subbab 15.4.7.*

DataReader, Dijalankan Sungguhan

Kode 15.4 Perulangan `AdoBasics/ReaderDemo.cs` atas para pegawai Laundry, dengan `DataTableReader`, `IDataReader` yang sudah tersedia di `System.Data`.

```
            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));
            }
```

*Pada **Kode 15.4**, `GetOrdinal` dijalankan sekali sebelum perulangan, dan `IsDBNull` menjaga `GetString`: pemilik usaha, E0001, tidak memiliki penyelia. `DataTable.Load(reader)` bekerja dengan setiap penyedia. Buku: Subbab 15.4.7.*

Contoh Terkerjakan 1: Tiga Baris, Satu Tindakan

Lima kilogram cuci-kering-lipat seharga Rp 8.000 per kg; dibayar Rp 20.000 sekarang. Server menghitung tagihan, Persamaan (15.2), dan sisanya, Persamaan (15.3):

$$\text{tagihan} = \text{harga} \times \text{unit} = 8000 \times 5 = 40000 \tag{15.2}$$

$$\text{sisa} = \text{tagihan} - \sum_{p \in \text{pembayaran nota}} \text{jumlah}_p = 40000 - 20000 = 20000 \tag{15.3}$$

> **DEFINISI. Transaksi** adalah sekelompok pernyataan yang diperlakukan basis data sebagai satu tindakan yang tak terbagi, dengan empat sifat **ACID**: atomisitas, konsistensi, isolasi, ketahanan. Transaksi berakhir dengan **commit** atau **rollback**.

Tulis `Bill` dan `StatusLog`, lalu gagal pada `Payment`, dan pembukuan menyatakan Rp 40.000 masih terutang sementara laci kas berisi Rp 20.000 lebih banyak daripada yang dapat dijelaskan. Buku: Subbab 15.5.1.

Unit Kerja: Aturan Lebih Dulu

Kode 15.5 Awal `AdoBillRepository.CreateBillWithFirstPayment`: satu koneksi, satu transaksi, harga dibaca dan aturan diperiksa di dalamnya.

```
    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)}.");
```

*Pada **Kode 15.5** tagihan berasal dari daftar harga menurut Persamaan (15.2), tidak pernah dari formulir. Membuang (dispose) transaksi yang belum di-commit maupun di-rollback berarti **me-rollback-nya**. Buku: Subbab 15.5.2.*

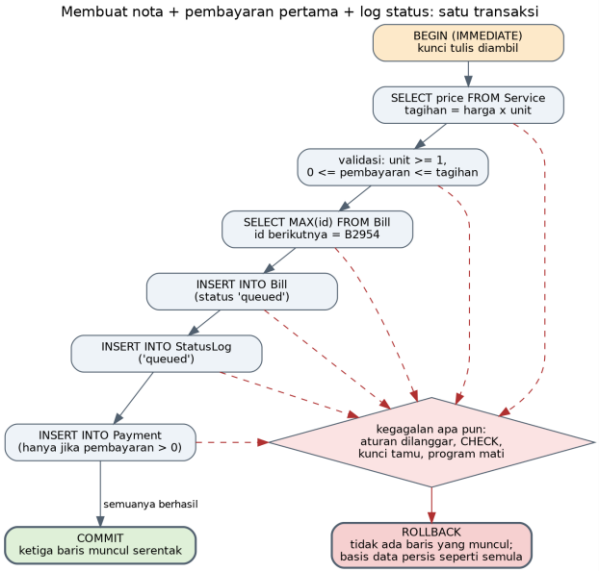
Unit Kerja: Tulis, lalu *Commit*

Kode 15.6 Sisa `CreateBillWithFirstPayment`: id berikutnya dibaca di dalam transaksi, log dan pembayaran ditulis, lalu `Commit`, atau `Rollback` dan satu kalimat.

```
            // 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");
            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);
        }
```

*Setiap perintah pada **Kode 15.6** menerima `tx`; `INSERT` untuk nota tidak ditampilkan di sini. `SELECT MAX(id)` mengembalikan `B2953`, sehingga nota baru adalah `B2954`. `DbException` menjadi `DataLayerException`, sehingga halaman tidak pernah melihat tipe milik penyedia. Buku: Subbab 15.5.2.*

Semuanya, atau Tidak Sama Sekali



Gambar 15.7 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`.

***Gambar 15.7**: setiap kegagalan sesudah `BEGIN` berujung pada `ROLLBACK`; hanya jalur yang melewati setiap langkah yang sampai ke `COMMIT`. `TransactionScope` dan `SaveChanges()` milik EF Core adalah dua cara lain menuliskan hal yang sama. Buku: Subbab 15.5.2 dan 15.5.4.*

Unit Kerja secara Umum

Pseudokode 15.2 Sebuah unit kerja. Semua yang ditulis `work` menjadi terlihat pada 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
```

Pseudokode 15.2 *diimplementasikan lima kali dalam bab ini: di atas ADO.NET, di atas DataSet, serta dalam PHP, Java, dan Python. Buku: Subbab 15.5.2.*

Tingkat Isolasi

- 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 15.8 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.

***Gambar 15.8**: isolasi dibeli dengan kunci, dan kunci mengurangi konkurensi. `ReadCommitted` adalah bawaan yang lazim. Buku: Subbab 15.5.3.*

Tiga Anomali di Laundry

Tabel 15.4 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`

*Phantom pada **Tabel 15.4** adalah masalah id berikutnya. SQLite hanya mengizinkan satu pihak menulis pada satu waktu: `BEGIN IMMEDIATE` langsung mengambil kunci tulis. InnoDB milik MySQL memerlukan `FOR UPDATE`. Kunci auto-increment menghindari soal ini. Buku: Subbab 15.5.3.*

Contoh Terkerjakan 2: Dua Kasir, Dua Koneksi

Kode 15.7 Keluaran nyata `python/demo_isolation.py`: apa yang dilihat koneksi B selagi koneksi A berada di dalam transaksi.

```
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)
```

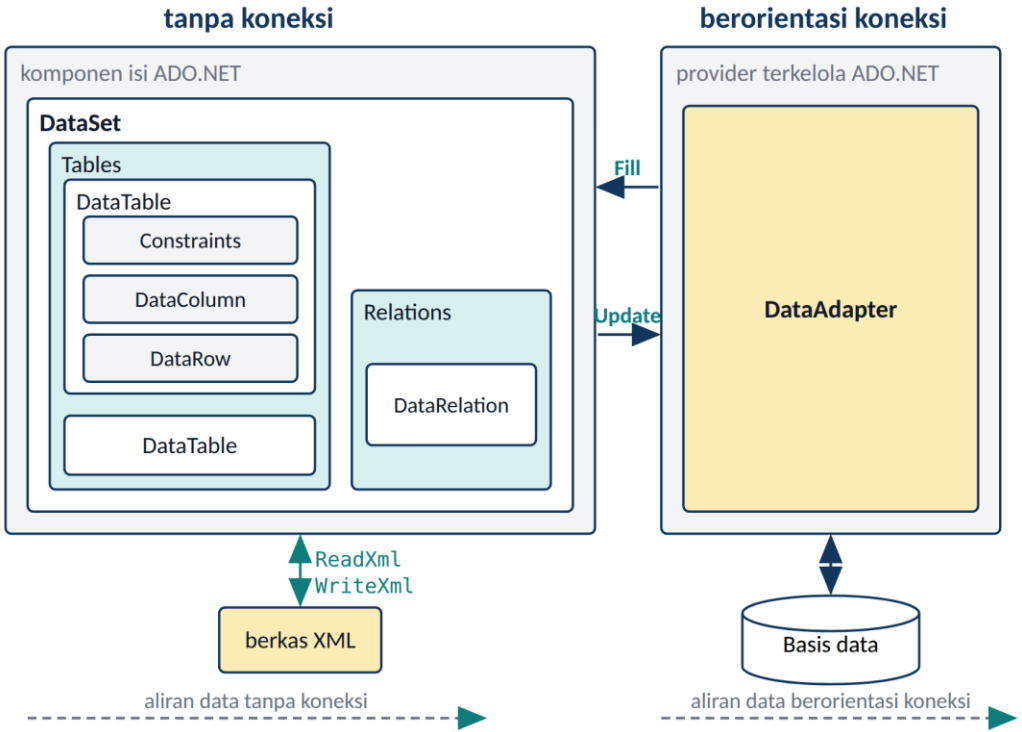
Langkah 1. A menyisipkan B2954 di dalam transaksi: A melihat 2954 nota, B masih 2953.

Langkah 2. B tidak dapat mulai menulis selama A memegang kunci; setelah lima detik, `database is locked`.

Langkah 3. Setelah `COMMIT` B melihat 2954; penghapusan A yang belum di-*commit* tidak pernah terlihat, dan `ROLLBACK` membatalkannya.

Kode 15.7 *adalah isolasi dan atomisitas yang diamati dari luar. Soal Latihan 15.6 menelusuri dua kasir yang sama langkah demi langkah. Buku: Subbab 15.5.3.*

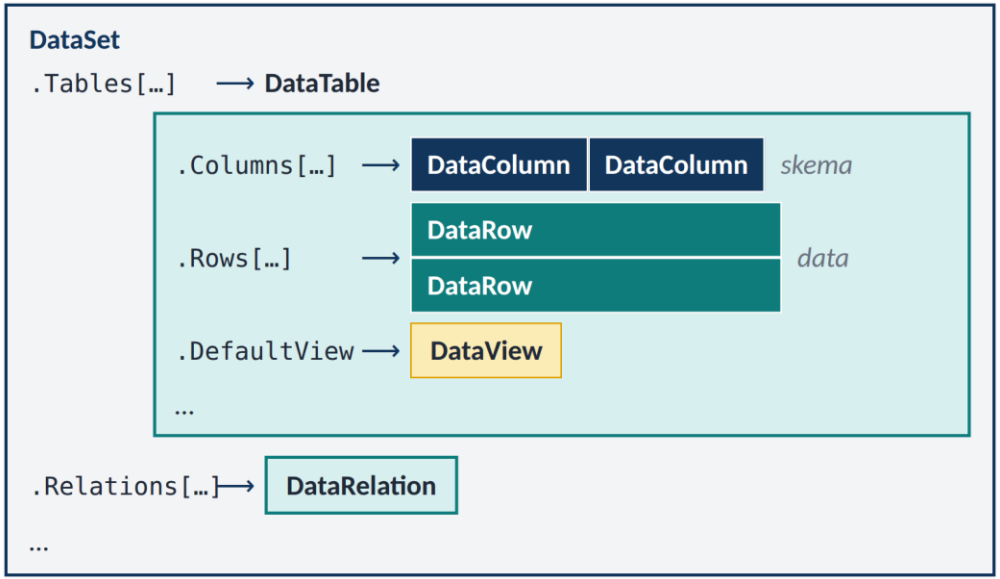
Model Terputus



Gambar 15.9 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.

***Gambar 15.9**: data disimpan di memori utama; koneksi hanya dipakai sebentar untuk mengisi dan memperbarui. Harganya: konflik perubahan. Buku: Subbab 15.6.1.*

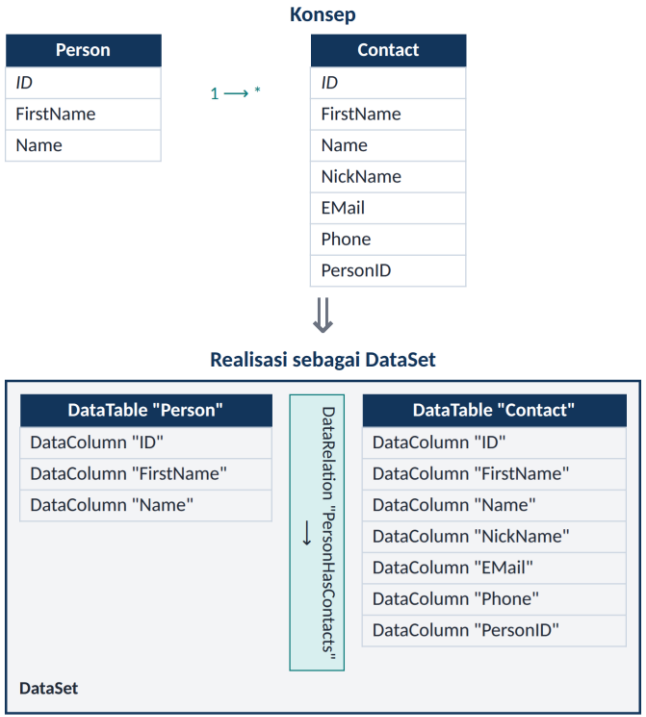
Struktur Sebuah DataSet



Gambar 15.10 Struktur DataSet. `.Tables[…]` menjangkau sebuah DataTable; `.Columns[…]` DataColumn-nya (skema); `.Rows[…]` DataRow-nya (data); `.DefaultView` DataView-nya; `.Relations[…]` DataRelation antartabel.

***Gambar 15.10**: setiap titik dalam `ds.Tables["Bill"].Rows[0]["status"]` melangkah satu kotak ke dalam. Kolom adalah skema, baris adalah data; `PrimaryKey` adalah larik kolom; DataRelation menghubungkan induk dan anak. Buku: Subbab 15.6.2.*

Orang dan Kontaknya



Langkah implementasi: definisikan skema; definisikan data; akses data.

Gambar 15.11 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`.

***Gambar 15.11**: DataSet yang dibangun tanpa basis data sama sekali, dalam tiga langkah: definisikan skema, definisikan data, akses data. Buku: Subbab 15.6.3.*

Sebuah Relasi dan Dua Baris

Kode 15.8 Dari `AdoBasics/MyAdo.cs`: relasi `PersonHasContacts`, satu orang, dan satu kontak.

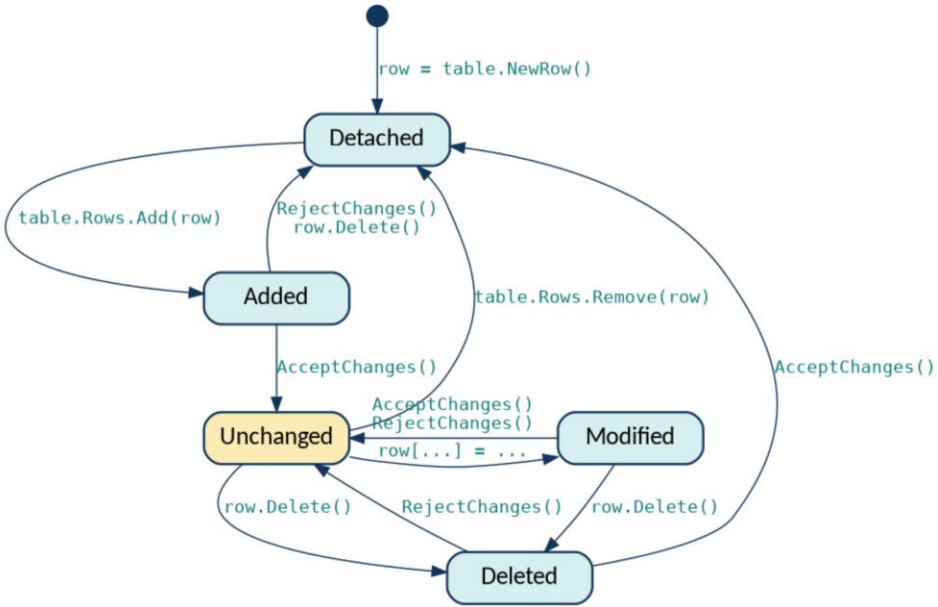
```
        DataColumn parentCol = ds.Tables["Person"]!.Columns["ID"]!;
        DataColumn childCol = ds.Tables["Contact"]!.Columns["PersonID"]!;
        DataRelation rel = new DataRelation("PersonHasContacts", parentCol, childCol);
        ds.Relations.Add(rel);
        DataRow personRow = personTable.NewRow();
        personRow[1] = "Wolfgang";                          // column 1 is FirstName
        personRow["LastName"] = "Beer";
        personTable.Rows.Add(personRow);
        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);
```

*`ID` pada **Kode 15.8** bersifat `ReadOnly`, `Unique`, dan `AutoIncrement` mulai 1; menambahkan relasi juga membuat constraint foreign key. `NewRow` membuat baris yang belum masuk tabel; `GetChildRows("PersonHasContacts")` adalah join di memori. Buku: Subbab 15.6.3.*

Keadaan Sebuah Baris

```
public DataRowState RowState { get; }
```

```
public enum DataRowState {
    Added, Deleted, Detached, Modified, Unchanged
}
```



Gambar 15.12 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*.

***Gambar 15.12**: DataAdapter membaca keadaan ini. Buku: Subbab 15.6.4.*

Contoh Terkerjakan 3: Satu Baris Melewati Setiap Keadaan

Kode 15.9 Keluaran nyata `AdoBasics/RowStates.cs`: satu baris `Service`, keadaannya, dan tiga versi harganya setelah setiap langkah.

```
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)
```

*Pada **Kode 15.9**: baris baru hanya punya nilai proposed; di antara `BeginEdit` dan `EndEdit` keadaannya tidak berubah; baris yang dihapus hanya menyimpan versi aslinya, agar adapter masih dapat menyebut baris mana yang dihapus. `RejectChanges` tidak menghapus `RowError`: panggil `ClearErrors`. Buku: Subbab 15.6.4–15.6.6.*

DataView: Urutkan, Cari, Saring

Kode 15.10 Keluaran nyata `AdoBasics/DataViews.cs`: pengurutan, pencarian, saringan, dan saringan keadaan baris pada satu *view*.

```
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
```

*Pada **Kode 15.10**, `Find` melakukan pencarian biner pada kolom pengurutan dan mengembalikan posisi di dalam view, atau −1. View tidak menyalin baris. `RowFilter` adalah bahasa ekspresi: merekatkan teks pengunjung ke dalamnya sama berbahayanya dengan merekatkannya ke SQL. Buku: Subbab 15.6.7.*

Seluruh Laundry di Memori

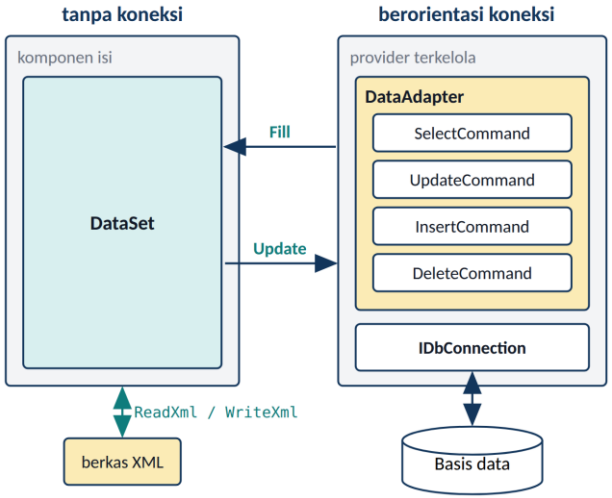
Kode 15.11 Dari keluaran `dotnet run -- laundry`: ketujuh tabel beserta *constraint*-nya, dan nota B0003 yang dicapai melalui relasi-relasinya.

```
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
```

*Pada **Kode 15.11** setiap relasi juga merupakan constraint foreign key; DataTable tidak memiliki `CHECK`, sehingga handler `RowChanging` menegakkan metode pembayaran. `DataTable.Compute` menghasilkan pendapatan September setiap gerai, dan SQLite menyetujuinya. Buku: Subbab 15.6.8.*

DataAdapter

- Sebuah **DataAdapter** menghubungkan DataSet dengan sumber data
  - **Fill**: mengisi DataSet
  - **Update**: menulis kembali perubahan
- DataAdapter memakai objek-objek Command
  - SelectCommand
  - InsertCommand
  - DeleteCommand
  - UpdateCommand



Gambar 15.13 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.

***Gambar 15.13**: `Fill` menjalankan `SelectCommand` dan memuat DataSet; `Update` menjalankan perintah insert, update, dan delete untuk baris yang berubah. `Fill` membuka koneksi yang tertutup lalu menutupnya lagi. Buku: Subbab 15.7.1 dan 15.7.2.*

Perjalanan Pulang-Pergi, dengan Pemeriksaan Konflik

Kode 15.12 Dari `AdapterRoundTrip.MySql.cs`: `Fill`, CommandBuilder, dua peristiwa baris, dan perubahan di memori. Tidak dijalankan dalam buku ini (tanpa MySQL, tanpa NuGet).

```
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);
```

*`UPDATE` yang dibangkitkan pada **Kode 15.12** membandingkan setiap nilai asli; bila tidak ada baris yang cocok, `Update` melempar `DBConcurrencyException`: konkurensi optimistis. `Microsoft.Data.Sqlite` tidak memiliki DataAdapter. DataSet juga membaca dan menulis XML, skema, dan DiffGram. Buku: Subbab 15.7.3–15.7.5 dan 15.8.*

Stored Procedure

> **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 ketika klien memanggilnya dengan `CALL nama(argumen)`.

Kode 15.13 Dari `create_bill_with_payment.mysql.sql`: `catch` milik prosedur, transaksinya, dan id berikutnya yang dibaca di bawah kunci. Tidak dijalankan dalam buku ini.

```
    -- 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;
    -- 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'));
```

*Pada **Kode 15.13**, `EXIT HANDLER … ROLLBACK; RESIGNAL` adalah `catch`, `SIGNAL SQLSTATE '45000'` adalah `throw`, dan `FOR UPDATE` menghentikan phantom. Kelebihan: satu kali pulang-pergi, satu implementasi, hanya `GRANT EXECUTE`. Kekurangan: bahasa yang lebih lemah, dialek per basis data, dan SQLite tidak memilikinya. Buku: Subbab 15.10.1 dan 15.10.2.*

Memanggilnya dari ADO.NET

Kode 15.14 Dari `CallProcedure.MySql.cs`: `CommandType.StoredProcedure`, satu parameter untuk setiap parameter prosedur, dan parameter keluaran untuk id nota baru. Tidak dijalankan dalam buku ini.

```
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);
```

*Pada **Kode 15.14**, `CommandText` berisi nama prosedur, bukan SQL. Stored procedure dan metode repository adalah dua tempat untuk unit kerja yang sama: yang penting adalah adanya tepat satu tempat. Buku: Subbab 15.10.3.*

Contoh Terkerjakan 4: *SQL Injection*

Langkah 1. Program merekatkan teks pengunjung di antara tanda kutip:

```
"SELECT id, name FROM Customer WHERE name = '" + input + "'"
```

Langkah 2. Pengunjung mengetik `x' OR '1'='1`, dan server mengurai:

```
SELECT id, name FROM Customer WHERE name = 'x' OR '1'='1'
```

Langkah 3. Syaratnya adalah Persamaan (15.4), benar untuk setiap baris:

$$(name = \text{'x'}) \vee (\text{'1'} = \text{'1'}) \tag{15.4}$$

*Tanda kutip pertama pengunjung menutup string milik pemrogram, dan tanda kutip terakhir pemrogram menutup string milik pengunjung. Injeksi bukan soal SQL; ia soal **perekatan**. Buku: Subbab 15.11.1.*

Contoh Terkerjakan 4: Akibatnya

Kode 15.15 Keluaran nyata `php/demo/injection.php` pada salinan basis data Laundry: kueri yang direkatkan, kueri yang sama dengan *prepared statement*, dan `UPDATE` yang direkatkan.

```
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

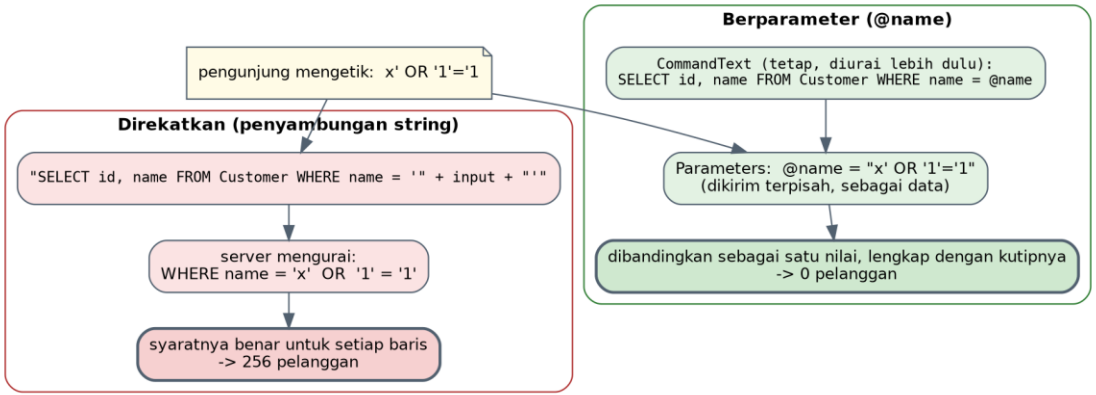
glued UPDATE meant for one customer: customers with zero credit went from 232 to 256
```

`SELECT` yang direkatkan membocorkan seluruh 256 pelanggan; yang di-*prepare* tidak menemukan satu pun, karena tidak ada yang bernama `x' OR '1'='1`.

`UPDATE` yang direkatkan, yang dimaksudkan untuk satu pelanggan, membuat deposit ke-24 pelanggan yang memilikinya menjadi nol.

*Pada **Kode 15.15**, `UPDATE` yang direkatkan membiarkan orang asing memilih baris mana yang berubah. Serangan yang sama bekerja pada ekspresi `DataTable.Select`; menggandakan tanda kutip membantu, tetapi prinsipnya adalah sama sekali tidak membangun ekspresi. Buku: Subbab 15.11.1 dan 15.11.2.*

Direkatkan versus Berparameter



Gambar 15.14 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.

***Gambar 15.14**: dengan parameter, teks tiba lebih dulu di basis data dan maknanya diputuskan; nilai datang sesudahnya, sebagai data, dan hanya dapat dibandingkan. Tidak ada yang perlu di-escape dan tidak ada yang dapat terlupa. Buku: Subbab 15.11.3.*

Apa yang Tidak Dapat Dilakukan Parameter

Identifier — nama tabel dan kolom, arah `ORDER BY`: pakailah *allow-list*, seperti yang dilakukan `Count(table)`.

Pola `LIKE` — `%` dan `_` tetap *wildcard* meski berada di parameter: *escape*-lah keduanya dan nyatakan `ESCAPE '!'`.

Bahasa lain — `RowFilter`, XPath, perintah *shell*: hanya nilai yang lolos pola seperti `^O[0-9]{2}$`.

Pertahanan berlapis — hak akses minimum, tanpa SQL di pesan galat, validasi di batas masuk.

Parameter membawa nilai; tiga hal yang bukan nilai masing-masing memerlukan pertahanannya sendiri. Buku: Subbab 15.11.4 dan 15.11.5.

Repository

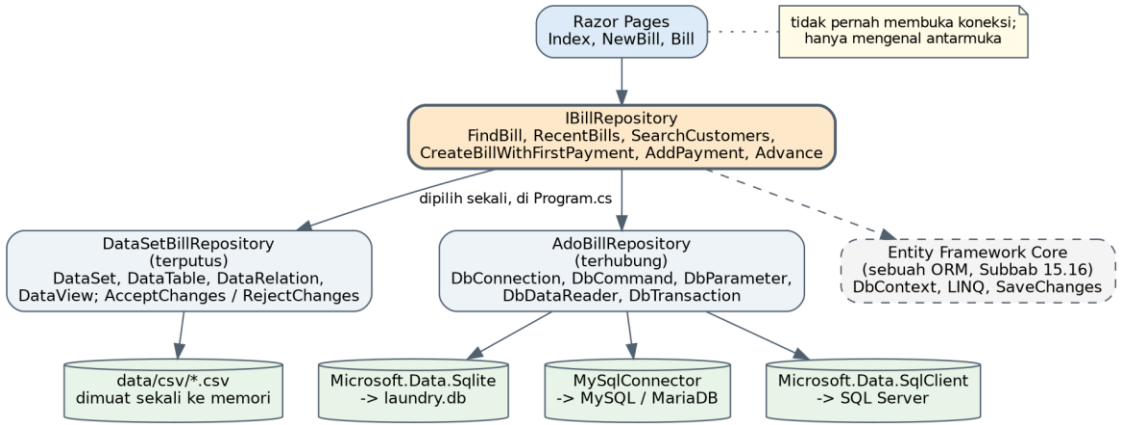
> **DEFINISI.** ***Repository*** adalah objek yang memberi bagian aplikasi lainnya akses ke data tersimpan melalui metode-metode yang dinamai dalam bahasa bisnis, dan menyembunyikan bagaimana dan di mana data disimpan. Kode di atas **lapisan data** tidak pernah melihat koneksi, perintah, DataSet, atau sebaris SQL pun.

Kode 15.16 Anggota-anggota `IBillRepository`: segala hal yang boleh diminta halaman dari penyimpanan.

```
public interface IBillRepository
{
    BillView? FindBill(string id);
    IReadOnlyList<BillView> RecentBills(string? outletId, int limit);
    IReadOnlyList<CustomerRow> SearchCustomers(string text, int limit = 20);
    string CreateBillWithFirstPayment(NewBill bill);
    void AddPayment(string billId, string method, decimal amount, string? reference, string now);
    string Advance(string billId, string employeeId, string now);
    int Count(string table);
}
```

Kode 15.16 *memberi halaman yang tidak dapat berulah, satu program uji untuk setiap implementasi, dan kebebasan mengganti penyimpanan dengan satu baris di* `Program.cs`*.* `BillView.Balance` *adalah Persamaan (15.3). Buku: Subbab 15.12.1.*

Lapisan Data Profitina Laundry



Gambar 15.15 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`.

***Gambar 15.15**: `DataSetBillRepository` menyimpan basis data di memori dengan `AcceptChanges` sebagai commit dan `RejectChanges` sebagai rollback; `AdoBillRepository` menjangkau penyedia mana pun, yang disebut dalam satu berkas kecil yang hanya dikompilasi dengan `-p:UseSqlite=true`. Buku: Subbab 15.12.2 dan 15.12.3.*

Contoh Terkerjakan 5: *Rollback*, Dihitung

Kode 15.17 Sepuluh dari lima belas pemeriksaan `Laundry.Tests` terhadap `DataSetBillRepository`.

```
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
```

*Pada **Kode 15.17** kelebihan bayar ditolak sebelum apa pun ditulis, metode yang buruk ditolak sesudah nota dan lognya ditambahkan, pelanggan yang tidak dikenal ditolak oleh foreign key. Setelah tiga penolakan, jumlah barisnya tepat sama dengan sesudah dua nota yang baik. Buku: Subbab 15.12.4.*

Halaman di Atas *Repository*

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 15.16 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.

***Gambar 15.16**: Razor Pages di atas `IBillRepository` pada port 9501, setelah Post/Redirect/Get. POST yang ditolak menampilkan kalimat dari repository dengan masukan kasir tetap terjaga; `MyADO2.aspx` dari Web Forms menjadi Razor Page tanpa postback dan tanpa view state. Buku: Subbab 15.12.5 dan 15.12.6.*

Empat Jalur Berdampingan

Tabel 15.5 Lapisan data Laundry dalam empat bahasa: enam dari sepuluh baris.

Gagasan	C# / ADO.NET	PHP / PDO	Java / JDBC	Python / `sqlite3`
Koneksi	`DbConnection`	`PDO`	`Connection`	`sqlite3.Connection`
Penanda	`@name`	`?`, `:name`	`?`	`?`, `:name`
Membaca baris (terhubung)	`DbDataReader.Read()`	`foreach` atas `PDOStatement`	`ResultSet.next()`	menelusuri kursor
Memulai	`BeginTransaction()`	`BEGIN IMMEDIATE` / `beginTransaction()`	`setAutoCommit(false)`	`BEGIN IMMEDIATE`
Mengakhiri	`Commit()` / `Rollback()`	`COMMIT` / `ROLLBACK`	`commit()` / `rollback()`	`COMMIT` / `ROLLBACK`
Pengujian di sini	15 pemeriksaan, lulus	15 pemeriksaan, lulus	15 pemeriksaan, lulus	10 pengujian, lulus

***Tabel 15.5**: SQL-nya sama di keempatnya; uji PHP, Java, dan Python berjalan terhadap basis data SQLite yang sama, dan itulah bukti bahwa SQL C#-nya benar. `CachedRowSet` milik Java adalah DataTable-nya. Buku: Subbab 15.13–15.15 dan 15.17.*

Entity Framework Core

Kode 15.18 Dari unit kerja dengan EF Core: LINQ untuk harga dan id berikutnya, objek anak ditambahkan melalui *navigation property*, satu `SaveChanges`. Tidak dikompilasi maupun dijalankan dalam buku ini.

```
        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");
        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;
```

***Kode 15.18**: ORM memetakan kelas ke tabel; LINQ menjadi SQL berparameter; `SaveChanges` mengirim ketiga `INSERT` dalam satu transaksi. Dapper berada di tengah: Anda menulis SQL-nya, Dapper memetakan barisnya. EF Core adalah ADO.NET di bawahnya. Buku: Subbab 15.16 dan Tabel 15.9.*

Profitina dalam Praktik

Satu lapisan data: halaman dan *handler* API meminta "*dashboard* pengguna ini" atau "catat pembelian ini"; mereka tidak pernah memegang koneksi.

Unit kerja: perpanjangan yang dibayar dari saldo, pembelian QRIS yang dikreditkan, kiriman beserta penghitungnya — masing-masing satu transaksi.

Selalu parameter, *allow-list* untuk *identifier*, dan satu SQL untuk PostgreSQL di produksi dan SQLite di pengujian.

UU 27/2022: ekspor data adalah pembacaan lintas setiap tabel; penghapusan akun adalah satu transaksi yang menghapus semuanya atau tidak sama sekali.

Studi kasus yang dapat dijalankan adalah Profitina Laundry; yang terbawa adalah bentuknya. Buku: Subbab 15.18.

Kesalahan yang Sering Terjadi

Merekatkan teks ke SQL atau `RowFilter`; kata sandi di kode sumber; menyambung sebagai `sa` atau `root`; lupa menutup.

Perintah tanpa transaksinya; id berikutnya dibaca di luar transaksi; *foreign key* SQLite dibiarkan mati; `BEGIN` tertunda untuk baca-lalu-tulis.

Meng-*cast* NULL; `HasErrors` tanpa `ClearErrors`; memercayai skema XML hasil inferensi; mencari DataAdapter di `Microsoft.Data.Sqlite`.

Pesan galat basis data diperlihatkan kepada pengunjung; `LIKE` dengan `%` dari pengunjung; kode data dipanggil dari halaman.

Buku memasangkan setiap kesalahan dengan perbaikannya. Buku: Subbab 15.19.

Ringkasan: Lima Hal untuk Diingat

1. ADO.NET adalah satu antarmuka dengan satu penyedia per basis data dan dua paruh (**Gambar 15.2**); Persamaan (15.1) menjelaskan mengapa halaman web kebanyakan memakai paruh terhubung.
2. Buka selambat mungkin, selalu tutup (**Pseudokode 15.1**); nilai hanya bepergian sebagai parameter (**Kode 15.3**).
3. Transaksi menjadikan beberapa operasi tulis satu tindakan (**Gambar 15.7**); id berikutnya memerlukan kunci tulis atau kunci *auto-increment*.
4. DataSet adalah basis data relasional di memori yang mengingat setiap perubahan dalam keadaan dan versi baris (**Gambar 15.12**).
5. *SQL injection* adalah perekatan (Persamaan (15.4)); *repository* menyembunyikan lapisan data di balik bahasa usaha (**Gambar 15.15**).

Buku: Subbab 15.20; Soal Latihan 15.1–15.12 beserta penyelesaiannya di Subbab 15.21 dan 15.22.

Pertemuan Berikutnya

Bab 16 — Evaluasi dan Pembahasan IV: Satu Aplikasi ASP.NET yang Utuh, dan Seluruh Isi Buku.

Bab 13, 14, dan 15 dirangkai dalam satu aplikasi, dan soal-soal yang menjangkau seluruh buku.

Sebelum itu: kerjakan Soal Latihan 15.2, 15.5, 15.6, dan 15.8, lalu jalankan sendiri uji salah satu jalur.

Grid, meja depan yang tervalidasi, dan lapisan data bertemu dalam satu aplikasi. Buku: Bab 16.

Kuliah 16 — Evaluasi dan Pembahasan IV: Satu Aplikasi ASP.NET yang Utuh, dan Seluruh Isi Buku

Pemrograman Web (EF234301)

Buku: Irfan Subakti, *Pemrograman Web: Dari HTML hingga ASP.NET Core, dengan PHP, Java, dan Python*, Bab 16

Dosen: Irfan Subakti — yifana@gmail.com

Hari ini kita membangun satu situs yang utuh, Laundry Counter, sebagai enam soal, mengerjakan tujuh soal baru tentang Bab 13 sampai 15, lalu menengok kembali seluruh isi buku. Buku: Bab 16, Subbab 16.1–16.14.

Cakupan Pertemuan Ini

1. Metode dan peta — tiga keputusan ASP.NET, dan setiap gagasan Web Forms dalam bentuk Razor Pages-nya.

2. Proyek, Soal 16.1–16.6 — basis data, *login*, CRUD, tampilan, bukti, *hosting*.

3. *Grid* data, Soal 16.7–16.8 — pembagian halaman dan pengurutan yang aman; total yang pantas berada di *footer*.

4. Validasi dan *state*, Soal 16.9–16.10 — server yang memutuskan; satu rumah yang tepat untuk setiap nilai.

5. Lapisan data, Soal 16.11–16.13 — $N+1$ pertanyaan, semua atau tidak sama sekali, dan pembaruan yang hilang.

6. Puncak dan seluruh buku — aplikasi Profitina Laundry yang lengkap, dan ke mana melangkah berikutnya.

Setiap soal mengikuti urutan yang sama seperti di Bab 4, 8, dan 12: soalnya, dengan bahasa kita sendiri, rancangan, penyelesaian dalam tiga bahasa, baris demi baris, penyelesaian keliru yang menggoda, dan uji diri. Buku: Subbab 16.1.

Titik Tolak Kita

Bab 1–4: HTTP, HTML, CSS, JavaScript, dan PHP. **5–8:** SQL, PDO, sesi, XML, JSP dan JDBC, serta Laundry Desk. **9–12:** ASP.NET dan Web Forms.

Bab 13: data di halaman. **Bab 14:** validasi dan *state*. **Bab 15:** ADO.NET, terhubung dan terputus. Hari ini paling banyak bertumpu pada ketiganya.

> **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.

Web Forms hanya berjalan di Windows; setiap soal di sini diselesaikan dalam Razor Pages di .NET 10, lalu sekali lagi dalam Java dan Python, sehingga tiga server menjawab 37 pemeriksaan yang sama. Buku: Subbab 16.1.

Tiga Keputusan ASP.NET

1. **Model halaman:** Razor Pages, pewaris Web Forms yang paling dekat dengan upacara paling sedikit; MVC dan *minimal API* adalah alternatifnya.
2. **Batas lapisan data:** satu antarmuka, `ILaundryStore`, dengan `DataSetStore` yang terputus dan `DbLaundryStore` yang terhubung; satu baris di `Program.cs` menukarnya.
3. **Tempat setiap *state*:** *grid* di *query string*, pengguna di *cookie* terenkripsi, pesan di TempData, nota terakhir di sesi, fakta di lapisan data.

> **DEFINISI.** *Page model* adalah kelas di belakang sebuah halaman Razor: *property*-nya memegang apa yang ditampilkan dan diterima halaman, dan *handler*-nya (`OnGet`, `OnPost`) berjalan untuk setiap jenis permintaan. *Model binding* menyalin isian permintaan ke *property* `[BindProperty]`; *model validation* lalu mencatat setiap kegagalan di `ModelState`.

Ketiga keputusan ini ditambahkan pada metode delapan langkah dari Bab 8. Buku: Subbab 16.2.

Dari Web Forms ke Razor Pages

Tabel 16.1 Gagasan Web Forms dari Bab 9 sampai 15, bentuk Razor Pages-nya, dan tempatnya di Laundry Counter.

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`
`RequiredFieldValidator`	`[Required]`	`CustomerForm`, `BillForm`
`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
`Application["key"]`	layanan *singleton*	`ShopClock`, `TeachingSwitches`

Tabel 16.1 adalah jembatan dari Bab 9–15 ke pertemuan hari ini; tabel lengkap di buku berisi tujuh belas baris. Buku: Subbab 16.2 dan Tabel 16.1.

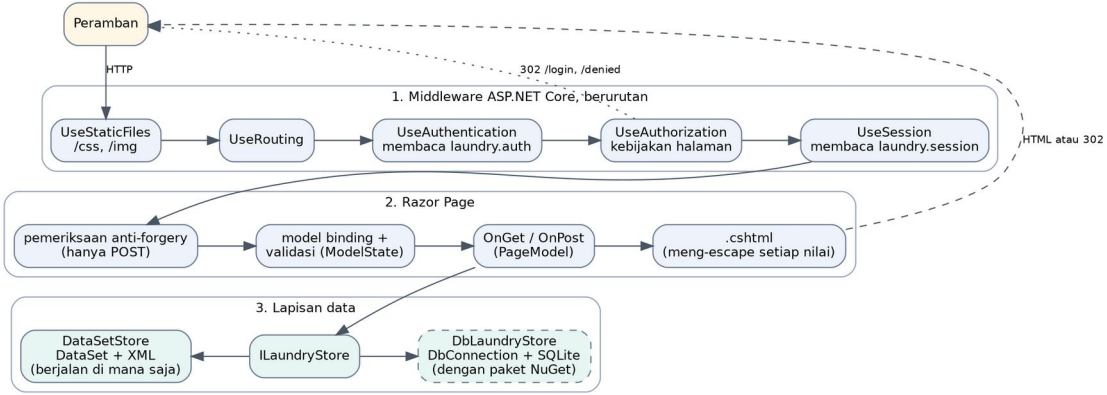
Proyeknya

> **SOAL.** Dengan ASP.NET, bangunlah situs web CRUD dengan *login*, registrasi, dan *logout*. Situs harus berjalan, dan sebaiknya indah. Tulislah laporan berisi rancangan basis data, daftar fitur, tangkapan layar, dan penjelasan halaman demi halaman, serta bagian kerja setiap anggota. Unggah situs ke layanan *hosting* gratis, dan terbitkan kode sumbernya di GitHub.

16.1 basis data; **16.2** registrasi, *login*, *logout*; **16.3** membuat, membaca, mengubah, menghapus; **16.4** indah; **16.5** bukti dan laporan; **16.6** repositori dan *hosting*.

Satu situs dan satu dokumen, diserahkan secara terbuka, jauh lebih mudah diselesaikan sebagai rantai enam soal kecil daripada sebagai satu soal. Buku: Subbab 16.4.

Satu Permintaan Melewati Laundry Counter



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.

Gambar 16.1: berkas statis, routing, cookie login yang diubah menjadi pengguna, kebijakan halaman, sesi; lalu untuk `POST` pemeriksaan anti-forgery, binding dan validasi, handler, dan markup yang di-escape; di bawahnya `ILaundryStore`. Setiap soal tinggal di suatu tempat di jalan ini. Buku: Subbab 16.4.

Sembilan Fitur

Tabel 16.2 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

Tabel 16.2: setiap huruf CRUD muncul setidaknya dua kali, dan setiap fitur adalah alamat yang dapat dipanggil sebuah uji. Buku: Subbab 16.4.

Dua Aturan Bisnis di Server

Jumlah yang ditagihkan disalin ke nota saat nota ditulis, Persamaan (16.1):

$$\text{tagihan} = \text{harga}(\text{layanan}) \times \text{jumlah} \tag{16.1}$$

Sisa tagihan dihitung, tidak pernah disimpan, Persamaan (16.2):

$$\text{sisa} = \text{tagihan} - \sum_{p \in \text{pembayaran nota itu}} \text{jumlah}(p) \tag{16.2}$$

Persamaan (16.1) adalah alasan formulir nota tidak memiliki field harga, dan alasan perubahan harga kemudian tidak menulis ulang sejarah; Persamaan (16.2) tidak pernah dapat berselisih dengan pembayarannya. Buku: Subbab 16.4.

Soal 16.1: Merancang Basis Data

SOAL. Berikan rancangan basis data situs web itu: model data konseptual dan model data fisik.

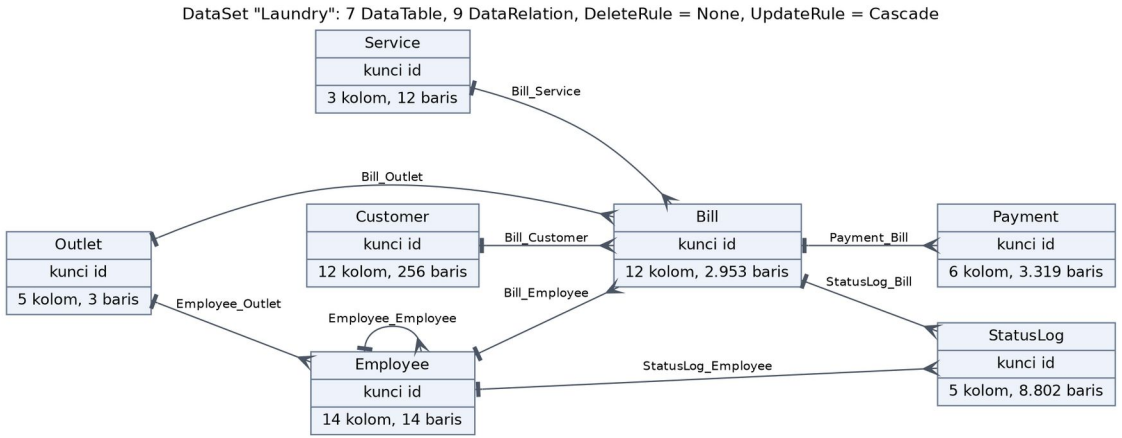
Dengan bahasa kita sendiri: dua gambar dan satu bukti. Bukti adalah bagian yang paling sering dilupakan laporan: tunjukkan bahwa aturan benar-benar ditegakkan dengan mencoba melanggarnya.

Rancangan: tujuh entitas dengan nota di pusatnya; model fisik dua kali, sebagai skema SQL (satu `CHECK` per aturan) dan sebagai `DataSet` ADO.NET.

DEFINISI. `DataRelation` menghubungkan kolom induk dengan kolom anak dan secara bawaan membuat *unique constraint* serta *foreign key constraint*, yang `DeleteRule`-nya salah satu dari `Cascade`, `SetNull`, `SetDefault`, atau `None`.

Selama `EnforceConstraints` bernilai `true`, `DataSet` menolak setiap baris yang melanggar kunci atau relasi, sebagaimana basis data. Buku: Subbab 16.4.1.

Model Fisik sebagai DataSet



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".

Gambar 16.2: tujuh tabel dengan kolom bertipe, panjang, dan kunci, serta sembilan relasi yang dinamai persis seperti foreign key pada skema SQL. Buku: Subbab 16.4.1.

Soal 16.1: Buktinya

Kode 16.1 `Relation` dalam `LaundryData.cs`: aturan hapus setiap relasi dipilih, dan kunci yang berubah merambat ke anak-anaknya.

```
    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
    }
```

```
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
```

Dengan Kode 16.1, empat percobaan ditolak dan tidak ada yang berpindah; percobaan kelima, mengganti kunci C0002, berhasil dan 21 notanya ikut. Java dan Python di SQLite sepakat. Buku: Subbab 16.4.1.

Soal 16.1: Nilai Bawaan yang Menggoda

Menggoda: cukup `ds.Relations.Add(...)`. Aturan hapus bawaannya adalah `Cascade`; jalankan dengan `--cascade`:

```
9 relations, delete rule Cascade: Employee_Outlet, Employee_Employee, Bill_Customer, Bill_Employee,
Bill_Service, Bill_Outlet, Payment_Bill, StatusLog_Bill, StatusLog_Employee
4. delete customer C0001       OK
   bills 2953 -> 2943, payments 3319 -> 3309, log rows 8802 -> 8772
```

Menghapus C0001 diam-diam menghapus sepuluh notanya, pembayarannya, dan tiga puluh baris jejak audit: lima puluh baris sejarah, satu klik.

> **UJI DIRI.** Tabel baru `BillPhoto(id, Bill_id, file)` menyimpan foto noda. Aturan hapus apa yang Anda berikan pada `BillPhoto_Bill`, dan mengapa jawabannya berbeda dengan `Payment_Bill`?

Jawaban: `Cascade` cocok untuk foto, yang tidak bermakna tanpa notanya; pembayaran adalah uang yang benar-benar berpindah tangan, sehingga `Payment_Bill` tetap `None`. Aturannya mengikuti apa baris anak itu. Buku: Subbab 16.4.1.

Soal 16.2: Registrasi, *Login*, *Logout*

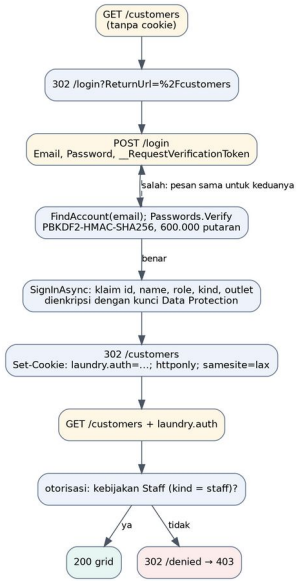
> **SOAL.** Situs web harus memiliki registrasi, *login*, dan *logout*.

Syarat yang tak terucap: setiap halaman yang tidak publik harus *menolak* pengunjung yang belum *login*, atau yang *login* dengan jenis akun yang salah.

Rancangan: staf memakai akun basis data; pelanggan mendaftar. *Login* memberi dua *claim*, `kind` dan `role`, yang diturunkan dari data. Kata sandi disimpan sebagai PBKDF2-HMAC-SHA256 dengan 600.000 iterasi.

Cookie login terenkripsi, `HttpOnly`, dan `SameSite=Lax`, serta berumur satu shift; `AuthorizeFolder` menaruh seluruh folder di bawah kebijakan seperti `Staff`. Buku: Subbab 16.4.2.

Sebuah *Login* dan Permintaan Berikutnya



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.

Gambar 16.3: setelah login, peramban membawa cookie, dan autentikasi mengubahnya kembali menjadi pengguna pada setiap permintaan. Buku: Subbab 16.4.2.

Handler Login

Kode 16.2 `OnPostAsync` dalam `Login.cshtml.cs`: kerja dan pesan yang sama untuk *e-mail* salah dan kata sandi salah, lalu *cookie*, dan pengalihan hanya ke alamat lokal.

```
    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();
        }
        HttpContext.Session.Clear();                          // nothing from before the login survives it
        await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
            new ClaimsPrincipal(new ClaimsIdentity(claims,
CookieAuthenticationDefaults.AuthenticationScheme)));
        TempData["Flash"] = $"Welcome, {a.Name}.";
        string home = a.Kind == "staff" ? "/customers" : "/my-bills";
        return LocalRedirect(Url.IsLocalUrl(ReturnUrl) ? ReturnUrl! : home);   // never to another site
```

Pada Kode 16.2, e-mail yang tidak dikenal tetap menjalankan pemeriksaan PBKDF2 penuh terhadap `Passwords.Dummy`, sehingga waktu tidak membocorkan apa pun. `Session.Clear()` tidak menyisakan apa pun dari sebelum login; `Url.IsLocalUrl` mencegah open redirect. Logout adalah `POST`. Buku: Subbab 16.4.2.

Soal 16.2: Tautan Tersembunyi Tidak Melindungi Apa Pun

Menggoda: sembunyikan tautan “Customers” tanpa kebijakan. Pengunjung yang mengetik alamatnya:

```
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:

```
anonymous GET /customers -> 302 /login
  the page says: (no customer data)
  the menu shows a visitor a Customers link: no
```

> **UJI DIRI.** Seorang penyelia di O02 boleh membuka laporan pendapatan, tetapi hanya untuk gerainya sendiri. Aturan itu milik kebijakan, *page model*, atau *markup*?

Jawaban: di page model atau lapisan data, karena bergantung pada baris yang diminta; menyaring di markup tetap mengirim angka gerai lain ke peramban. Buku: Subbab 16.4.2.

Soal 16.3: Membuat, Membaca, Mengubah, Menghapus

SOAL. Situs web harus menyediakan keempat operasi CRUD: membuat, membaca, mengubah, dan menghapus.

Kode 16.3 Sebagian `ILaundryStore.cs`: apa yang boleh ditanyakan halaman tentang akun dan pelanggan, dan tidak lebih.

```
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
```

Tiga lapisan: halaman, antarmuka Kode 16.3, dan store. Setiap perubahan memakai Post/Redirect/Get. `DeleteCustomer` mengembalikan `false` bila `Rule.None` menolak; `NewBill` menghitung harga dengan Persamaan (16.1); `DataSet` menyimpan di bawah satu kunci dan membatalkan dengan `RejectChanges`. Buku: Subbab 16.4.3.

Soal 16.3: *Overposting*

Menggoda: mengikat seluruh entitas. Seorang kasir mengirim formulir ubah C0005 ditambah satu *field*, `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
```

Di situs yang benar, yang mengikat *input model* berisi hanya *field* yang boleh diubah:

```
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
```

> **UJI DIRI.** Seorang pengunjung menambahkan `Input.Credit=1000000` ke permintaan registrasinya. Apa yang terjadi, dan adakah yang berubah bila `CustomerForm` mempunyai *property* `Credit` tersembunyi?

Jawaban: tidak terjadi apa-apa, karena `CustomerForm` tidak punya `Credit`; bila ada, model binding akan mengisinya. Menyembunyikan input tidak melindungi apa pun; meniadakan property itulah yang melindungi. Buku: Subbab 16.4.3.

Soal 16.4: Indah menurut Aturan

Satu bingkai, tindakan utama paling terlihat, angka rata kanan, label di setiap *field*, tata letak ponsel, fokus yang terlihat, dan teks yang cukup gelap: aturan kontras WCAG, Persamaan (16.3), Persamaan (16.4), dan Persamaan (16.5).

$$c_{\text{lin}} = \begin{cases} c/12.92 & \text{jika } c \leq 0.04045 \\ \left(\frac{c+0.055}{1.055}\right)^{2.4} & \text{selainnya} \end{cases} \tag{16.3}$$

$$L = 0.2126\,R_{\text{lin}} + 0.7152\,G_{\text{lin}} + 0.0722\,B_{\text{lin}} \tag{16.4}$$

$$\text{rasio} = \frac{L_1 + 0.05}{L_2 + 0.05} \tag{16.5}$$

Rasionya berkisar dari 1 sampai 21; tingkat AA meminta sedikitnya 4,5 untuk teks biasa. Warnanya diambil dari logo resmi Profitina, digelapkan sampai lulus. Buku: Subbab 16.4.4.

Contoh Terkerjakan 1: Teks Isi di Atas Putih

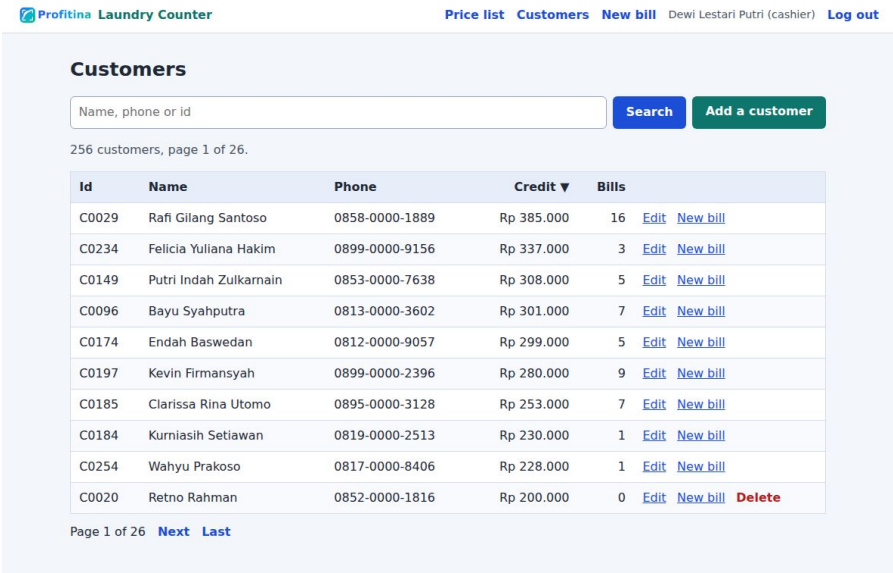
`#1f2937`: kanal $0{,}1216$, $0{,}1608$, $0{,}2157$ menjadi $0{,}0137$, $0{,}0222$, $0{,}0382$ menurut Persamaan (16.3); $L_2 = 0{,}0216$ menurut Persamaan (16.4); rasio $1{,}05 / 0{,}0716 = 14{,}7$.

Kode 16.4 Baris-baris keluaran `P164Contrast.cs`: kontras warna-warna teks `site.css`, dan abu-abu muda yang menggoda.

```
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
status: departed            #166534 on #dcfce7   6.49:1  AA pass
tempting: light grey        #aaaaaa on #ffffff   2.32:1  FAIL
```

Kode 16.4: abu-abu muda `#aaaaaa` yang menggoda hanya 2,32 banding 1, separuh dari yang diminta WCAG; di ponsel murah di bawah terik matahari, teks itu tidak terbaca. Java dan Python mencetak sebelas baris yang sama. Buku: Subbab 16.4.4.

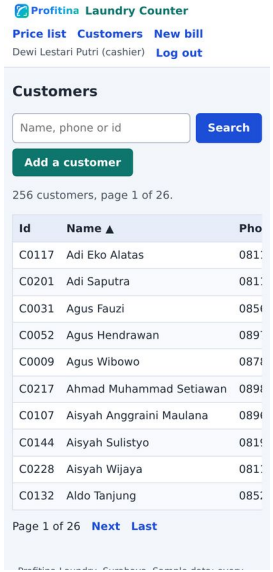
Grid, Diurutkan menurut Deposit



Gambar 16.4 *Grid* pelanggan Laundry Counter ASP.NET, diurutkan menurut deposit, dari yang terbesar. Hanya pelanggan tanpa nota yang mempunyai tombol Delete.

Gambar 16.4: angka rata kanan dengan digit selebar sama; `th[aria-sort]` *menggambar panah dari atribut yang juga dibaca pembaca layar. Buku: Subbab 16.4.4.*

Grid yang Sama di Ponsel



Gambar 16.5 *Grid* yang sama di layar seukuran ponsel: menunya turun ke baris berikut, dan *grid* bergulir di dalam kotaknya sendiri.

Gambar 16.5: sel tabel tidak dipatahkan dan grid bergulir di dalam kotaknya sendiri; menunya membungkus. Buku: Subbab 16.4.4.

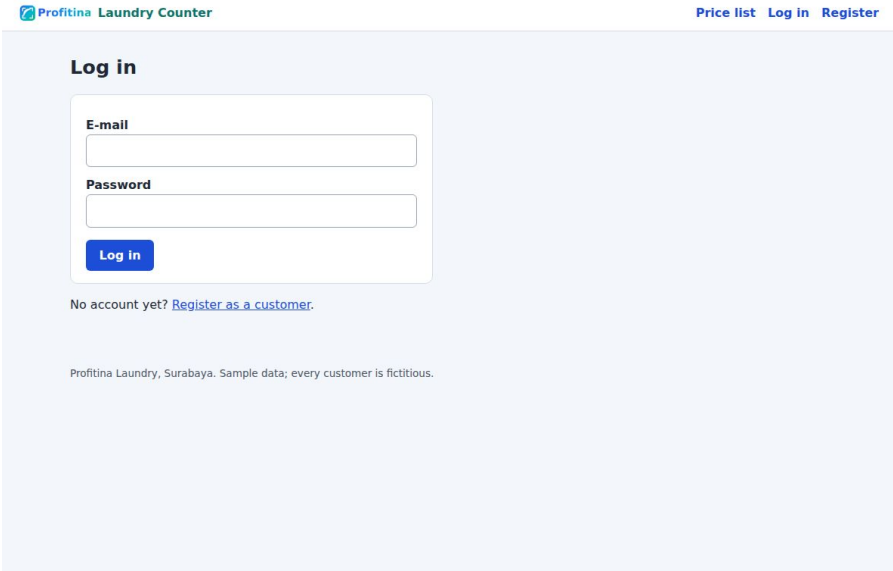
Soal 16.5: Bukti yang Dapat Dijalankan Ulang

> **SOAL.** Tulislah laporan berisi rancangan basis data, daftar fitur, tangkapan layar, penjelasan halaman demi halaman, serta tugas, bagian, dan kontribusi setiap anggota.

```
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  B2   every cookie set at login is HttpOnly and SameSite
PASS  B11  a cashier's edit is saved, but a posted Credit is ignored
PASS  C1   quantity 0 and rack Z99 are refused, each with its own message
PASS  C3   a valid bill is priced by the server: 5 kg x Rp 8.000
PASS  C6   a form without its anti-forgery token is refused
PASS  D4   the bill written earlier keeps the price it was charged
37 of 37 checks passed
```

Uji penerimaan melalui HTTP sungguhan dijalankan terhadap ketiga server dan hasil publish: `37 of 37 checks passed`, kalimat yang dapat diperiksa siapa pun. Tangkapan layar saja tidak dapat menunjukkan penolakan. “Semua mengerjakan semuanya” tidak dapat diperiksa; persentase yang cocok dengan commit dapat. Buku: Subbab 16.4.5 dan Tabel 16.4.

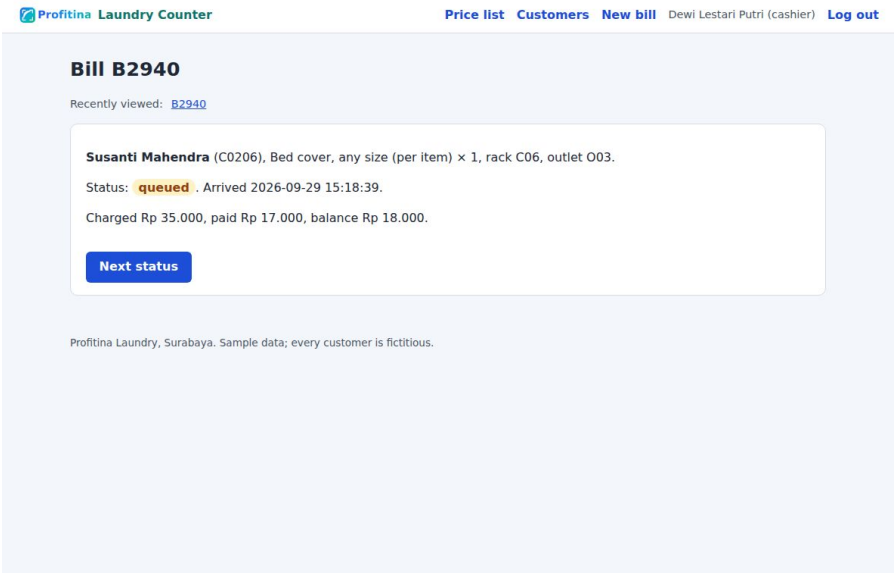
Halaman *Login*



Gambar 16.6 Halaman *login*. Pesan yang sama menjawab surel yang salah dan kata sandi yang salah.

Gambar 16.6: diambil oleh skrip (Playwright), sehingga dapat diambil ulang setelah setiap perubahan; skrip itu juga melaporkan field tanpa label: tidak ada. Buku: Subbab 16.4.5.

Satu Nota Selengkapnya



Gambar 16.7 Satu nota (fitur F7): pelanggan, layanan, rak, status beserta waktunya, dan besaran Persamaan (16.1) serta Persamaan (16.2).

Gambar 16.7: fitur F7, dengan jumlah menurut Persamaan (16.1) dan Persamaan (16.2) yang dihitung di server. Buku: Subbab 16.4.5.

Soal 16.6: Repositori dan *Hosting*

Rancangan: tidak ada hasil *build*, data hidup, atau rahasia di repositori; `dotnet publish -c Release`; pengaturan dari lingkungan (`Laundry__DataDir`); data dan *key ring* di folder yang bertahan melewati *restart*.

```
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
```

Menggoda: menerapkan laptop apa adanya, sehingga pengguna "terlempar keluar" pada setiap *restart*; atau meng-*commit* kunci dan `.env`, sehingga rahasia menjadi publik selamanya.

Key ring mengenkripsi cookie login, token anti-forgery, dan TempData; bila hilang, setiap cookie tidak terbaca. Java menyimpan sesi di memori; Flask memerlukan `LAUNDRY_SECRET` yang tetap. Buku: Subbab 16.4.6.

Soal 16.7: Membagi Halaman dan Mengurutkan dengan Aman

SOAL. Sepuluh pelanggan per halaman, dapat dicari, dapat diurutkan menurut id, nama, telepon, atau deposit; halaman, pencarian, dan urutan tiba di alamat. Setiap kombinasi nilai harus menghasilkan halaman yang masuk akal, tidak pernah galat atau kebocoran.

$$P = \max\left(1, \lceil \frac{n}{s} \rceil\right) = \max\left(1, \lfloor \frac{n+s-1}{s} \rfloor\right) \tag{16.6}$$

$$p' = \min\left(\max\left(p, 1\right), P\right) \tag{16.7}$$

$$\text{offset} = \left(p' - 1\right) s \tag{16.8}$$

Persamaan (16.6) dengan $s = 10$*:* $n = 10$ *memberi 1 halaman,* $n = 11$ *memberi 2,* $n = 256$ *memberi 26. Persamaan (16.7) mengubah halaman 0 menjadi 1 dan halaman 99 menjadi yang terakhir; Persamaan (16.8) membuat halaman 26 melewati 250 baris. Buku: Subbab 16.5.1.*

Nilai Dibandingkan, Nama dari Daftar

Kode 16.5 Dari `DataSetStore.Customers`: kolom pengurutan diambil dari `SortColumns`, dan teks pencarian dibandingkan di C#, tidak pernah diurai.

```
            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));
```

Pada Kode 16.5 urutan yang tidak dikenal menjadi `name`, dan `id ASC` memutus seri. Versi Java dan Python memakai placeholder untuk pencarian serta untuk `LIMIT` dan `OFFSET`. Nilai kuerinya `p`, bukan `page`, yang dicadangkan Razor Pages. Buku: Subbab 16.5.1.

Soal 16.7: Saringan yang Menggoda

```
  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.
```

Direkatkan ke `RowFilter`, `Ma'ruf` merobohkan halaman dan `zzz' OR name <> '` mengembalikan semua orang; di SQL, `ORDER BY` yang dibangun dari teks membocorkan sebuah *hash* satu pertanyaan ya-atau-tidak demi satu: injeksi *blind*.

> **UJI DIRI.** Pencarian `C0258` menemukan pelanggan yang namanya tidak mengandung “C0258”. Bagian mana yang menemukannya, dan mengapa berupa kesamaan, bukan `LIKE`?

Grid yang benar: `sort 'name; DROP'` menjadi `name`, halaman 99 menjadi 26. Jawaban: bagian id, `upper(id) = upper(?)`; id adalah kunci, dan `LIKE` hanya menghasilkan derau. Buku: Subbab 16.5.1.

Soal 16.8: *Grid* dengan *Footer*

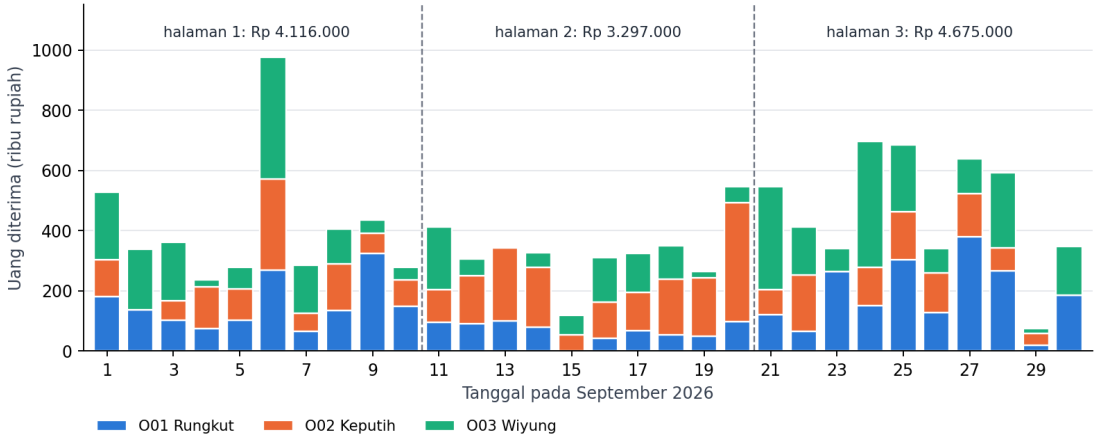
> **SOAL.** Uang yang diterima dalam sebulan, satu baris per hari, satu kolom per gerai dan satu kolom total, sepuluh hari per halaman. Di bawah baris-baris itu *grid* menampilkan total. Total yang mana, dihitung dari apa?

Dengan $r_{d,o}$ uang yang diterima pada hari d di gerai o, D_p hari-hari di halaman p, dan D semua hari, Persamaan (16.9):

$$\text{TotalHalaman}_o = \sum_{d \in D_p} r_{d,o} \quad \text{TotalBulan}_o = \sum_{d \in D} r_{d,o} \tag{16.9}$$

Keduanya sama hanya bila $D_p = D$, sehingga laporan menampilkan keduanya, masing-masing berlabel apa yang dijumlahkannya; total bulan tidak pernah diambil dari baris di layar. Buku: Subbab 16.5.2.

September 2026 dalam Tiga Halaman



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.

Gambar 16.8: ketiga total halaman jelas berbeda, dan bersama-sama membentuk bulannya. Buku: Subbab 16.5.2.

Contoh Terkerjakan 2: Dua Baris *Footer*

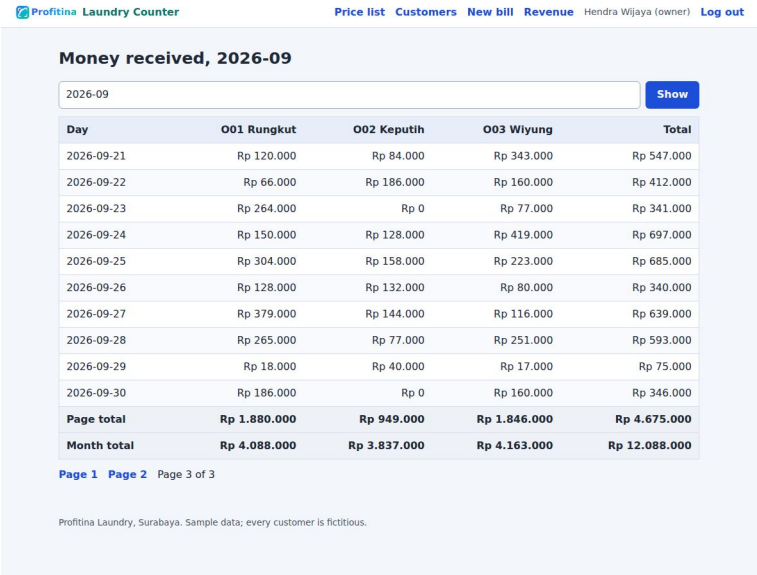
```
Day                   O01           O02           O03         Total
2026-09-21      Rp 120.000     Rp 84.000    Rp 343.000    Rp 547.000
2026-09-28      Rp 265.000     Rp 77.000    Rp 251.000    Rp 593.000
2026-09-29       Rp 18.000     Rp 40.000     Rp 17.000     Rp 75.000
2026-09-30      Rp 186.000           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
```

Dengan tangan, O02 di halaman 3: 84+186+0+128+158+132+144+77+40+0=949 ribu rupiah.

Menggoda: menjumlahkan baris saat digambar (`RowDataBound`): "total bulan" menjadi Rp 4.675.000, kurang dari dua perlima kebenarannya.

Total bulan, Rp 12.088.000, sama dengan `SUM(amount)` yang independen. Rata-rata per hari diambil atas ke-30 hari: 12.088.000/30=402.933 *rupiah. Buku: Subbab 16.5.2.*

Laporan Pendapatan



Gambar 16.9 Laporan pendapatan, halaman 3 September 2026 (fitur F9). Dua baris *footer*, masing-masing menyatakan apa yang dijumlahkannya.

Gambar 16.9: halaman 3 September 2026; bulan yang tidak ada dijawab dengan 400, Bad Request. Buku: Subbab 16.5.2.

Soal 16.9: Validasi yang Tidak Dapat Dimatikan Peramban

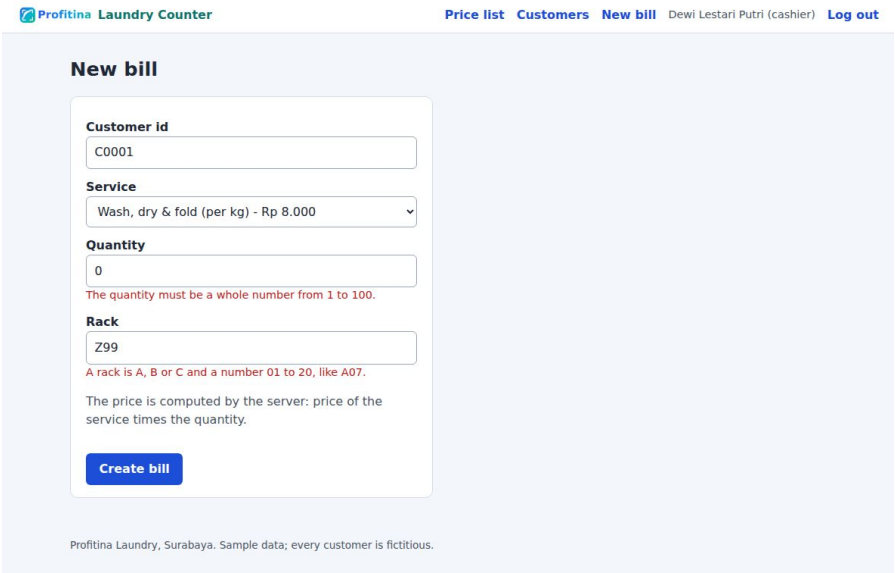
Kode 16.6 Dari `BillForm.cs`: dua atribut yang masing-masing memeriksa satu *field*, dan `Validate`, yang memerlukan data.

```
    [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)$", ErrorMessage = "A rack is A, B or C and
a number 01 to 20, like A07.")]
    public string Rack { get; set; } = "";
    public IEnumerable<ValidationResult> Validate(ValidationContext context)
    {
        ILaundryStore store = context.GetRequiredService<ILaundryStore>();
        if (store.Customer(CustomerId) == null)
            yield return new ValidationResult($"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)]);
    }
```

`min`, `max`, `required`, dan `pattern` di peramban adalah kesopanan; Kode 16.6 adalah hukumnya, dan `if (!ModelState.IsValid) return Page();` adalah satu baris yang dilewati setiap permintaan. `Validate` hanya berjalan bila semua atribut lulus. Buku: Subbab 16.6.1.

Server Menolak Kedua *Field*



Gambar 16.10 Formulir nota baru setelah pemeriksaan peramban dimatikan: *server* menolak kedua isian, masing-masing dengan pesannya sendiri.

Gambar 16.10: pemeriksaan peramban dimatikan dengan `form.noValidate = true`*, lalu kuantitas 0 dan rak* `Z99` *dikirim. Buku: Subbab 16.6.1.*

Soal 16.9: Minus Lima Kilogram

Menggoda: memercayai peramban, dan sebuah skrip 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
```

Dengan pemeriksaan server terpasang:

```
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.
```

UJI DIRI. Layanan ekspres `S0002` hanya dijual sampai 10 kg. Di mana aturannya ditaruh, dan mengapa tidak sebagai atribut?

Jawaban: di `Validate`, karena aturannya menghubungkan dua field; atribut hanya melihat satu property sekali. Persamaan (16.1) bekerja dengan setia, atas masukan yang tak masuk akal. Buku: Subbab 16.6.1.

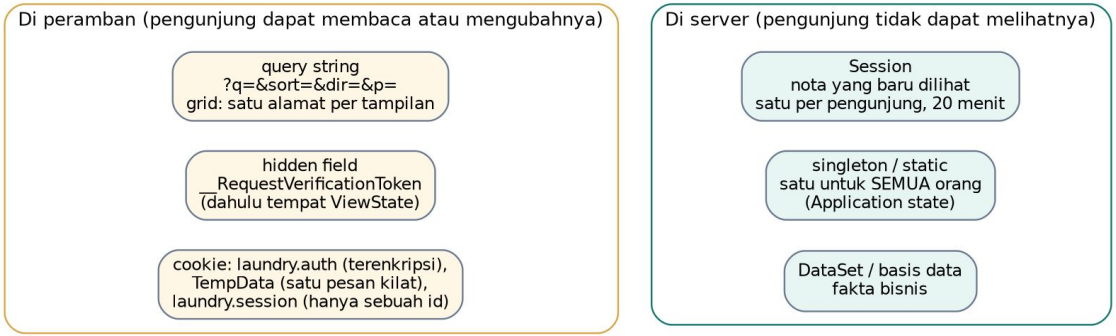
Soal 16.10: Di Mana Setiap Nilai Tinggal?

SOAL. Tampilkan, di atas setiap halaman staf, lima nota terakhir yang dibuka pengguna *ini*. Lalu daftarkan setiap potong *state* yang disimpan Laundry Counter, dan jelaskan mengapa masing-masing tinggal di tempatnya.

DEFINISI. *Session state* adalah kamus kecil yang disimpan server untuk setiap pengunjung, yang hanya memegang kuncinya di dalam *cookie*. *Application state* dipakai bersama oleh semua pengunjung selama proses hidup: sebuah layanan *singleton*, atau *field* `static`.

Satu pengunjung, satu sesi kerja, tidak berbahaya bila hilang, bersifat pribadi: nota terakhir adalah session state, disimpan sebagai satu string seperti `B2954,B0001`. Buku: Subbab 16.6.2.

State Laundry Counter



Gambar 16.11 Tempat tinggal *state* Laundry Counter. Apa yang ada di peramban dapat dibaca atau diubah pengunjung; apa yang ada di *server* tidak.

Gambar 16.11: apa yang ada di peramban dapat dibaca atau diubah pengunjung; apa yang ada di server tidak. Buku: Subbab 16.6.2.

Satu Rumah yang Tepat untuk Setiap Nilai

Tabel 16.3 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

Tabel 16.3: setiap nilai memiliki pemilik, masa hidup, dan tingkat kepercayaan. Buku: Subbab 16.6.2 dan Tabel 16.5.

Soal 16.10: *List* yang *Static*

Menggoda: `static List<string>` bertahan di antara permintaan, sehingga tampak cocok. Ia adalah *application state*, satu daftar untuk semua orang:

```
the cashier opened B0001 and B2950
the owner, who opened nothing, sees: Recently viewed: B2950 B0001
```

Dengan sesi, dua peramban yang sama:

```
the cashier opened B0001 and B2950
the owner, who opened nothing, sees: (no recently viewed bills)
```

> **UJI DIRI.** “Nota yang dibuat hari ini, semua gerai”, langsung diperbarui tanpa kueri: sesi, *singleton*, atau basis data? Apa ruginya bila *restart*?

Jawaban: singleton dengan `Interlocked.Increment`, dibangun ulang dari basis data saat mulai; basis data tetap kebenarannya. Buku: Subbab 16.6.2.

Soal 16.11: Satu Pertanyaan atau Tiga Ribu

> **SOAL.** Hitunglah berapa nota yang masih ada di toko dan berapa seluruh pelanggan masih berutang, dengan Persamaan (16.2) atas semua nota, secara terhubung dan terputus, lalu jelaskan cara bertanya mana yang cepat.

```
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
```

$$T_{N+1} \approx (N+1)\,t_q + N\,t_r \text{ sedangkan } T_1 \approx t_q + N\,t_r \tag{16.10}$$

57 nota di toko; Rp 1.529.000 terutang. Menurut Persamaan (16.10) selisihnya adalah $N\,t_q$; dengan server basis data 0,5 ms per pertanyaan, $2.953 \times 0{,}5$ ms $\approx 1{,}5$ s untuk satu halaman. Buku: Subbab 16.7.1.

Soal 16.11: Jebakan $N+1$

Menggoda: "untuk setiap nota, ambil pembayarannya": benar, wajar, dan sisi lambat dari Persamaan (16.10). Saringan C# `$"Bill_id = '{b["id"]}'"` juga teks yang direkatkan.

Obat: `GetChildRows("Payment_Bill")` dalam model terputus; satu `GROUP BY` atau *join* dalam model terhubung.

UJI DIRI. "My bills" menghitung jumlah bayar setiap nota dengan *subquery* berkorelasi. Apakah ia terkena masalah $N+1$?

Jawaban: tidak; subquery berjalan di dalam basis data, dalam satu kueri, tanpa pulang-pergi per baris. $N+1$ *menyangkut pulang-pergi. Buku: Subbab 16.7.1.*

Soal 16.12: Nota dan Baris Lognya

Baris log menyebut pegawai `E9999`, sehingga *foreign key* menolak operasi tulis kedua. Dengan satu unit kerja:

```
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
```

Menggoda, setiap operasi tulis di-*commit* sendiri-sendiri (`--no-tx`):

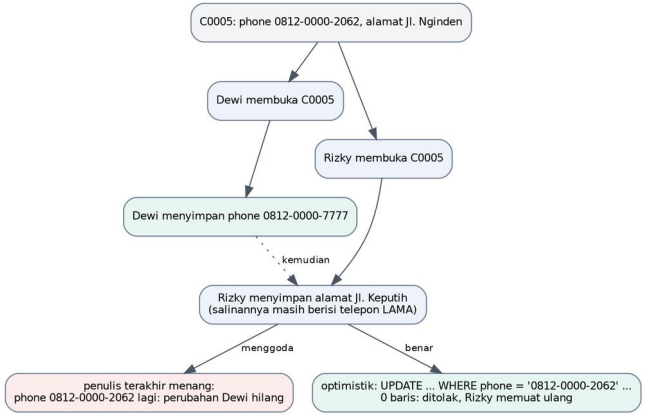
```
before: 2953 bills, 8802 log rows; bills without a log row: 0
error: ForeignKeyConstraint StatusLog_Employee requires the child key values (E9999) to exist in the
parent table.
after:  2954 bills, 8802 log rows; bills without a log row: 1
```

UJI DIRI. Mencatat pembayaran menulis satu baris `Payment`. Apakah itu memerlukan transaksi?

Aturannya: bila beberapa operasi tulis menyatakan satu fakta, taruhlah dalam satu transaksi. Satu pernyataan sudah atomik; pernyataan kedua membuat transaksi menjadi perlu. Buku: Subbab 16.7.2.

Soal 16.13: Pembaruan yang Hilang

SOAL. Dewi dan Rizky membuka pelanggan C0005 pada saat yang sama. Dewi membetulkan telepon lalu menyimpan; semenit kemudian Rizky membetulkan alamat lalu menyimpan. Apa yang terjadi pada koreksi Dewi, dan bagaimana penyimpanan harus ditulis?



Gambar 16.12 Dua kasir mengubah pelanggan yang sama. Menulis balik seluruh salinan menghilangkan perubahan Dewi; memeriksa nilai-nilai asli justru menolak penyimpanan Rizky.

Gambar 16.12: setiap halaman adalah salinan terputus dari barisnya. Buku: Subbab 16.7.3.

Contoh Terkerjakan 3: Konkurensi Optimistis

Dengan o_c nilai kolom c saat halaman dibuka dan v_c nilainya sekarang, penyimpanan berhasil tepat ketika Persamaan (16.11) berlaku:

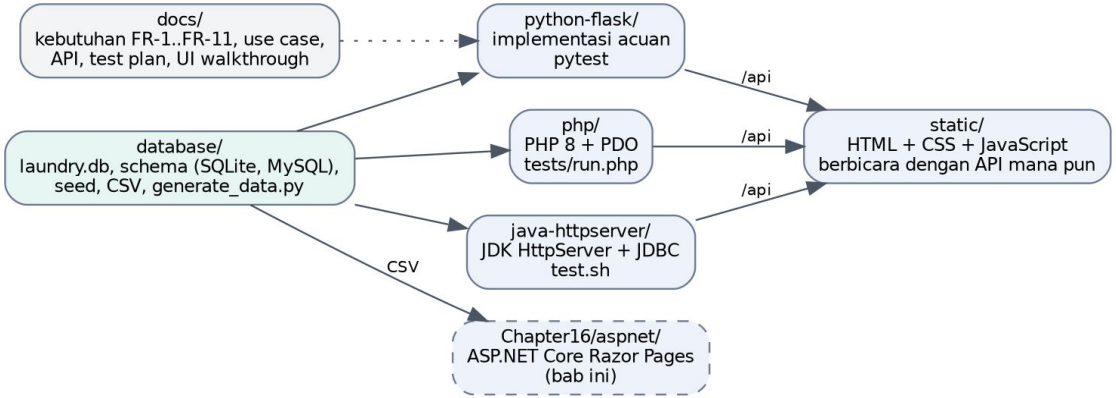
$$\text{penyimpanan berhasil} \Leftrightarrow \forall c : v_c = o_c \tag{16.11}$$

```
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
```

Menggoda, “yang terakhir menyimpan, dialah yang menang”: kedua penyimpanan berhasil, dan nomor telepon kembali ke yang lama, padahal tidak ada yang mengetiknya lagi.

Dalam SQL: `UPDATE … WHERE id = @id AND phone = @old_phone …`*; nol baris berubah berarti “muat ulang”. Kolom* `version` *mendeteksi perubahan apa pun dengan* `WHERE` *yang lebih pendek, dengan harga penolakan yang lebih sering. Buku: Subbab 16.7.3.*

Puncaknya



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.

Gambar 16.13: satu basis data, satu himpunan dokumen (kebutuhan FR-1 sampai FR-11, use case, API, rencana uji), dan jalur dalam Flask, PHP, Java, HTML statis, serta situs ASP.NET bab ini. Buku: Subbab 16.8.

Lebih dari Empat Ratus Pemeriksaan

Flask (`python -m pytest`), **PHP** (`php tests/run.php`), **Java** (`./test.sh`), dan *front end* **statis** (`node tests/run-tests.js`):

```
185 passed, 7 skipped in 52.25s
PASS 100   FAIL 0   MANUAL 3   SKIP 0   (total 103)
RESULT: 107 passed, 0 failed, 4 not automated/skipped  (17.4 s)
PASS 26   FAIL 0   (total 26)
```

Ditambah 37 pemeriksaan yang dilulusi masing-masing dari tiga situs hari ini; SHA-256 basis data contoh tidak berubah sesudahnya.

Bacalah aplikasinya menurut urutan sebuah permintaan: 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. Buku: Subbab 16.8.

Perbandingan Soal 16.1–16.6

Tabel 16.4 Enam soal proyek, dan kekeliruan setiap penyelesaian 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*

Bacalah kolom terakhir Tabel 16.4: penyelesaian yang menggoda bekerja pada contoh biasa dan gagal pada satu masukan yang tidak pernah dikirim orang jujur. Buku: Subbab 16.9.

Perbandingan Soal 16.7–16.13

Tabel 16.5 Tujuh soal baru, dan kekeliruan setiap penyelesaian yang menggoda.

Soal	Pokok	Gagasan utama	Yang dikeliru penyelesaian yang menggoda
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

Pada hampir setiap baris Tabel 16.5, kegagalannya tidak terlihat dari peramban pengguna yang jujur. Buku: Subbab 16.9 dan Tabel 16.7.

Ke Mana Melangkah Berikutnya

***Framework*:** Django, Spring Boot, Laravel, ASP.NET Core MVC; temukan di masing-masing bagian yang sudah Anda kenal — *hashing*, *binding*, validasi, *anti-forgery*, transaksi, sesi.

Penerapan: HTTPS, basis data sungguhan dengan cadangan yang diuji, log dan alamat `/health`, serta *pipeline* yang menerapkan hanya bila uji lulus.

Keamanan: OWASP Top 10 sebagai daftar periksa; dan UU PDP No. 27/2022 untuk data pribadi: kumpulkan hanya yang diperlukan layanan, dan hapus bila tidak ada lagi alasan menyimpannya.

> **JALUR BACAAN.** Untuk web itu sendiri: spesifikasi HTTP, RFC 9110, dibaca sekali dari awal sampai akhir. Untuk keamanan: OWASP Top 10 dan OWASP Cheat Sheet Series. Untuk data: *Patterns of Enterprise Application Architecture* karya Fowler. Lalu dokumentasi *framework* yang Anda pilih.

Pilihlah framework menurut bahasa dan ekosistemnya, bukan menurut tren. Buku: Subbab 16.11 dan Tabel 16.9–16.10.

Profitina dalam Praktik

Akun dan data pribadi: *hash* Argon2id, *login* yang dibatasi lajunya, persetujuan, ekspor, penggantian *e-mail*, penghapusan akun.

Satu basis data, dua mesin, PostgreSQL dan SQLite, dijaga oleh sebuah penjaga; **dasbor dalam satu permintaan**, pelajaran dari Persamaan (16.10).

Penerapan dengan gerbang: sebuah rilis dianggap berhasil hanya setelah pemeriksaan kesehatannya menjawab dan lebih dari seribu uji lulus.

Kegagalan dikatakan terus terang: bila model tidak tersedia, aplikasi mengatakannya; kiriman yang gagal dipulihkan.

Produk memberi saran dan penggunanya yang memutuskan; versi yang dapat dijalankan dari setiap gagasan adalah kode Profitina Laundry. Buku: Subbab 16.12.

Kesalahan yang Sering Terjadi

Aturan hapus bawaan; tautan tersembunyi alih-alih kebijakan; mengikat lebih dari yang boleh diubah formulir; warna dipilih dengan mata.

Bukti yang tidak dapat dijalankan ulang; menerapkan laptop apa adanya; teks pengunjung di dalam kueri atau ekspresi.

***Footer* dijumlahkan dari layar**; validasi hanya di peramban; data per pengguna di *field* `static`.

Satu pertanyaan per baris; dua operasi tulis untuk satu fakta tanpa transaksi; menulis kembali salinan yang basi; nilai kueri bernama `page`.

Setiap kesalahan menunjuk ke soal yang memperlihatkan perbaikannya. Buku: Subbab 16.13.

Ringkasan: Lima Hal untuk Diingat

1. Aplikasi yang utuh adalah delapan langkah Bab 8 ditambah tiga keputusan (Tabel 16.1): Razor Pages, antarmuka untuk lapisan data, satu rumah untuk setiap nilai.

2. Aturan tinggal di server: harga menurut Persamaan (16.1), kebijakan pada folder, validasi melalui `ModelState` (Soal 16.2, 16.3, 16.9).

3. *Grid* menjepit dan membagi halaman dengan aritmetika (Persamaan (16.7)), mengambil nama dari daftar, dan melabeli setiap total (Persamaan (16.9)).

4. Lapisan data bertanya sekali (Persamaan (16.10)), menulis satu fakta dalam satu transaksi, dan hanya menyimpan yang masih seperti semula (Persamaan (16.11)).

5. Bukti adalah uji yang dapat dijalankan ulang siapa pun; penerapan menjaga data dan kuncinya, dan rahasia tetap di luar repositori.

Buku: Subbab 16.14; Soal Latihan 16.1–16.12 beserta penyelesaiannya di Subbab 16.15 dan 16.16.

Kata Penutup

Buku ini dimulai dengan satu permintaan dan berakhir dengan sebuah toko: pelanggan, nota, pembayaran, dan orang-orang, di tiga gerai dan dalam empat bahasa, teruji dan diterapkan.

Teknologinya akan berubah, seperti Web Forms memberi jalan kepada Razor Pages; 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 bekerja?

Ajukan pertanyaan-pertanyaan itu pada setiap halaman yang Anda tulis, dan Anda akan menulis halaman yang dapat dipercaya orang untuk cuciannya, dan untuk jauh lebih banyak lagi. Buku: Subbab 16.10 dan 16.14.