Aplikasi Berbasis Web II • Pertemuan 4

Struktur Kontrol, Perulangan & Optimasi Database

Membangun View yang Bersih dan Efisien dengan Laravel 11 Blade Engine.

Dosen: Rizki Muliono, S.Kom, M.Kom | **Teknik Informatika, Universitas Medan Area.**

Agenda Engineer: Evolusi Kode View

Refactoring Logika

Mengganti struktur kontrol PHP standar yang berantakan dengan direktif sintaks Blade yang elegan (`@if`, `@isset`, `@empty`).

Penguasaan Iterasi

Mengelola perulangan data dengan `@forelse` dan memanfaatkan metadata cerdas dari variabel `$loop`.

Optimasi Arsitektur

Menghindari N+1 Problem Problem yang mematikan kinerja aplikasi dengan strategi Eager Loading (`with()`).

Arsitektur Clean Code: Separation of Concerns

Controller / Model (Logika Bisnis & Database)

Standard PHP (Jangan Lakukan Ini): Menggabungkan query database, perhitungan logika, dan HTML dalam satu file yang sama (Spaghetti Code).

View / Blade (Logika Tampilan)

Laravel Blade (Best Practice): Blade dirancang khusus HANYA untuk Logika Tampilan (Display Logic). Jangan pernah menaruh logika bisnis yang berat di dalam file Blade Anda.

Prinsip Engineer: Controller untuk memproses, Blade untuk menampilkan.

Refactoring Logika Kondisional: PHP vs Blade

Standard PHP (Masa Lalu)

```php
<?php
if ($nilai > 75) {
    echo "Lulus";
} else {
    echo "Tidak Lulus";
}
?>
```

PHP standar membutuhkan tag buka-tutup <?php ?> dan kurung kurawal {} yang membuat HTML berantakan.

Laravel Blade (Refactored)

```blade
@if ($nilai > 75)
    <p>Lulus</p>
@else
    <p>Tidak Lulus</p>
@endif
```

Blade menggunakan simbol @ tanpa kurung kurawal. Kode menjadi jauh lebih bersih, mudah dibaca, dan konsisten di dalam layer view.

Menguasai Direktif Kondisional Khusus

```
@isset($user)
    <p>Nama: {{ $user->name }}</p>
@endisset
```

Mengecek eksistensi

Mengecek apakah variabel $user telah didefinisikan oleh Controller dan tidak bernilai null.

```
@empty($posts)
    <p>Tidak ada post tersedia.</p>
@endempty
```

Mengecek kekosongan

Mengecek apakah variabel $posts adalah string kosong, array kosong, atau null.

```
@unless($isVerified)
    <div>Akun Anda belum terverifikasi.</div>
@endunless
```

Kebalikan dari IF

Menjalankan blok kode HANYA jika kondisi bernilai false. Mengurangi penulisan operator negasi !.

Evolusi Iterasi: Dari Manual ke Semantik

```php
<?php

for ($i = 1; $i <= 5; $i++) {
    echo "Perulangan ke-$i <br>";
}

$j = 1;
while ($j <= 5) {
    echo "Perulangan ke-$j <br>";
    $j++;
}
$k = 1;
do {
    echo "Perulangan ke-$k <br>";
} while ($k <= 5);
```

Langkah 1: Fondasi PHP

for, while, do...while

Membutuhkan inisialisasi counter manual ($i=1) dan increment ($i++). Rentan terhadap infinite loops.

Langkah 2: PHP Foreach

foreach

Menghilangkan counter manual, secara otomatis memetakan key dan value, ideal untuk memproses array.

Langkah 3: Blade Directives

@foreach, @for, @while

Membawa semua kekuatan perulangan PHP ke dalam View dengan sintaks bebas tag PHP, ditambah fitur eksklusif Laravel.

Direktif @forelse: Looping Elegan dengan Fallback

Engineer's Note

Menggabungkan perulangan dan pengecekan kondisi ke dalam satu struktur, menghindari penulisan if (!empty($data)) secara manual.

```
@forelse ($posts as $post)
    <li>{{ $post->title }}</li>
@empty
    <p>Tidak ada postingan.</p>
@endforelse
```

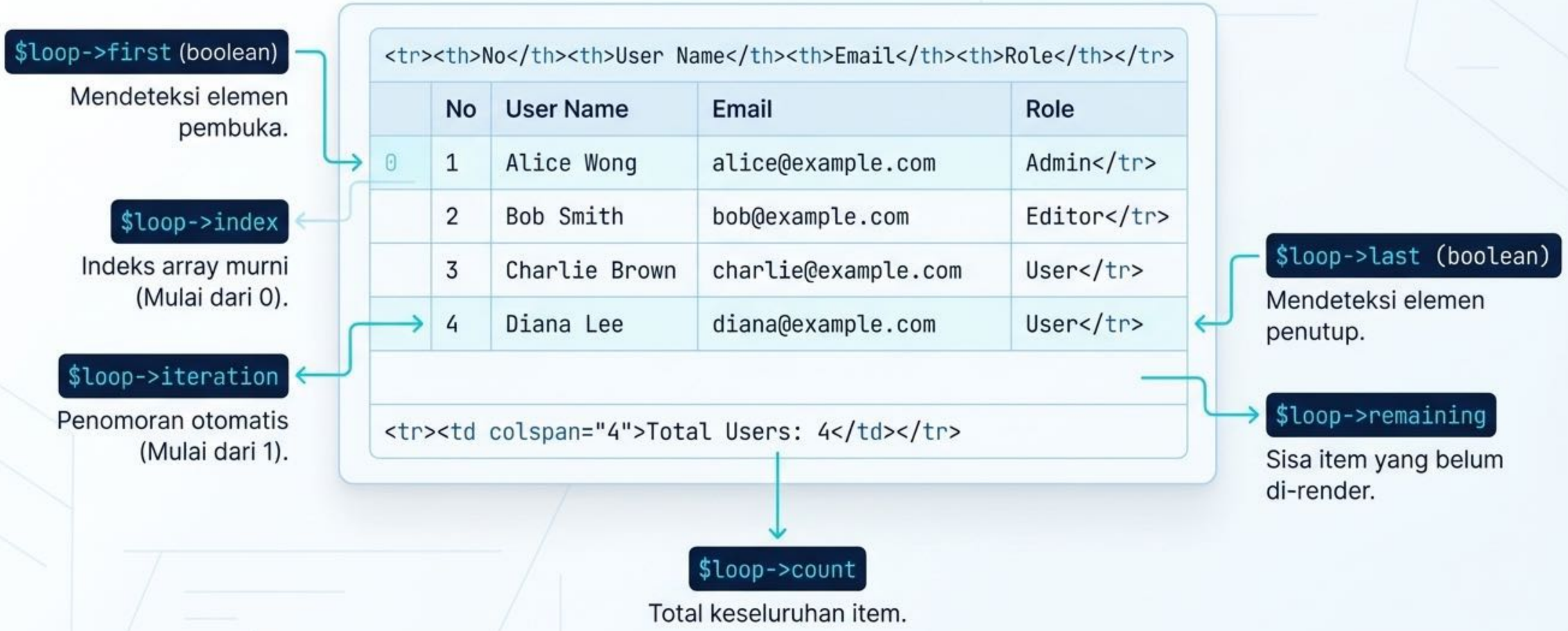
Kondisi Ideal: Array Ada Data

Jika $posts berisi data, Blade mengeksekusi blok iterasi sama persis seperti @foreach biasa.

Fallback: Array Kosong

Jika $posts kosong dari database, iterasi dilewati dan mengeksekusi pesan fallback ("Tidak ada postingan").

Anatomi Variabel Pembantu: Objek $loop



Implementasi $loop pada Tampilan Dunia Nyata

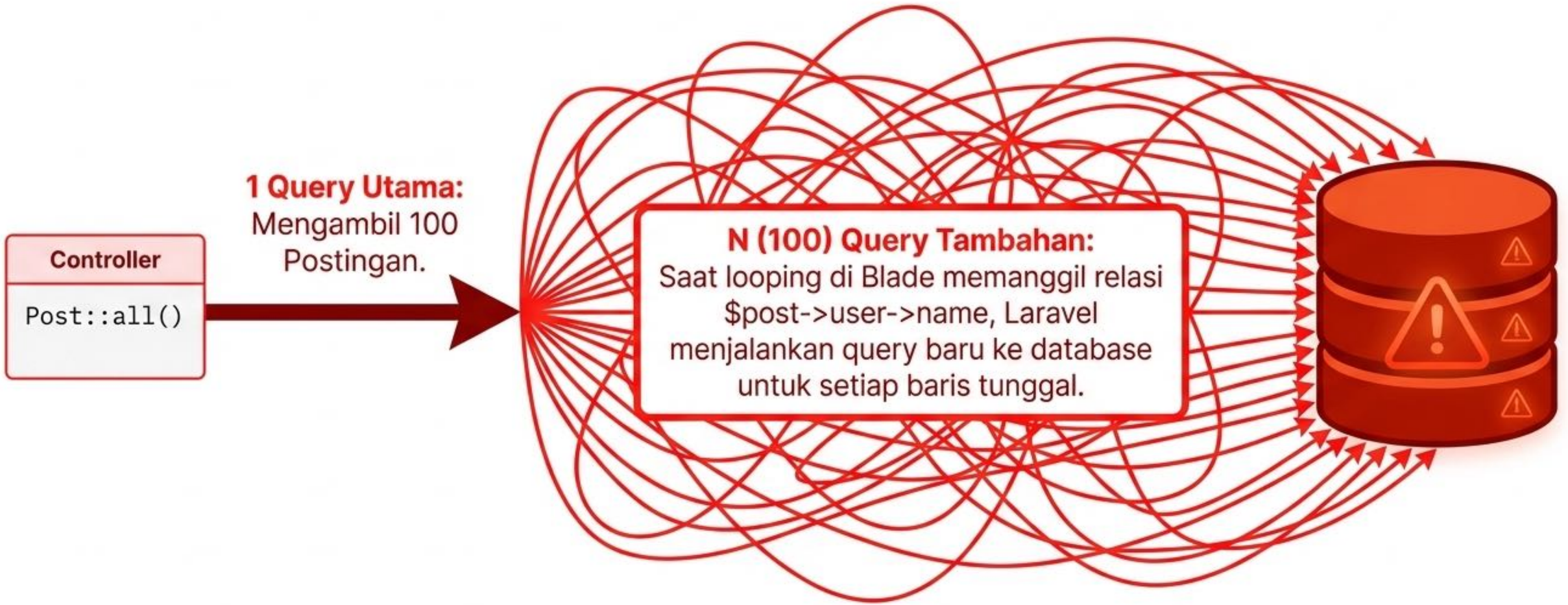
Laravel otomatis menyuntikkan variabel **$loop** setiap kali Anda menggunakan direktif **@foreach** atau **@forelse**.

```blade
@foreach($users as $user)
    <tr>
        <td>{{ $loop->iteration }}</td> <!-- Nomor urut -->
        <td>{{ $user->name }}</td>
        @if($loop->first)
            <td>Ini baris pertama</td>
        @endif
        @if($loop->last)
            <td>Ini baris terakhir</td>
        @endif
    </tr>
@endforeach
```

Mencetak nomor urut tabel secara otomatis tanpa perlu membuat variabel counter seperti $i = 1 di luar loop.

Memungkinkan kita memberikan class CSS khusus (seperti border tebal) hanya pada elemen pertama atau terakhir.

Pembunuh Kinerja yang Sunyi: N+1 Problem



Total: 101 Query dijalankan untuk satu halaman web. Server akan kelebihan beban.

Solusi Arsitektur: Eager Loading dengan **with()**

Controller

```
Post::with('user')->get()
```

Query 1: Mengambil semua 100 Postingan.

Query 2: Mengambil semua User yang terkait dengan postingan tersebut dalam **SATU** query tersendiri (SELECT * FROM users WHERE id IN ...).

Total: Hanya 2 Query. Pengambilan data direlasikan di memori aplikasi, bukan membebani database.

Refactoring Controller untuk Skalabilitas

Anti-Pattern (N+1 Problem)

```php
// Hindari ini di controller:
$posts = Post::all();
// Jika setiap post memiliki relasi user
```

$posts = Post::all(); — Terlihat sederhana, namun sangat berbahaya jika Model Post memiliki relasi user yang akan diakses di dalam perulangan Blade.

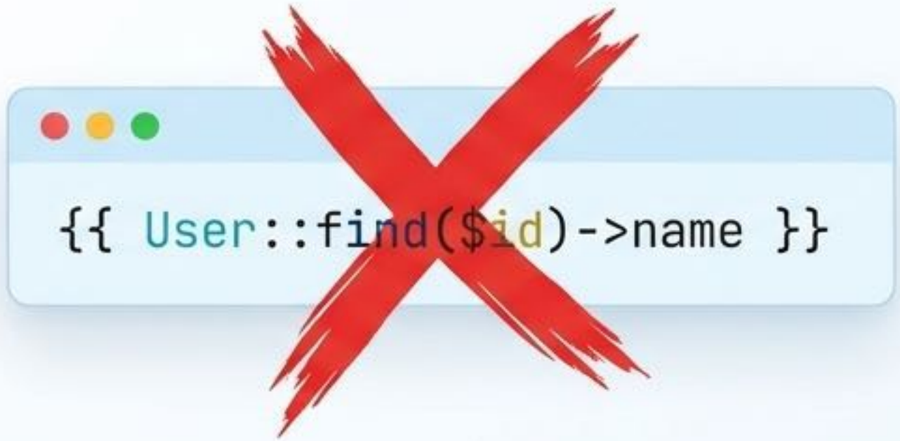
Best Practice (Eager Loading)

```php
$posts = Post::with('user')->get();
```

$posts = Post::with('user')->get(); — Metode with() menginstruksikan Laravel untuk menarik relasi user di awal (upfront) sebelum data dikirim ke View.

Kesalahan Umum Engineer Pemula

Logika Berat di View



✓ **Solusi:** Eksekusi query di **Controller**, lempar variabel yang sudah matang ke Blade. **Jangan jalankan query database langsung di Blade.**

PHP Murni di View

✓ **Solusi:** Gunakan Direktif Blade (**@if**, **@foreach**) untuk menjaga konsistensi sintaks. Hindari PHP murni di view.

Checklist Code Review Engineer Laravel

- ✅ **Pisahkan Komponen.** Apakah tampilan ini bisa dipecah? Gunakan `@include` atau komponen Blade untuk bagian UI yang dapat digunakan ulang (reusable).
- ✅ **Nol Query di Blade.** Apakah view ini mengeksekusi logika database? Pindahkan semuanya ke `Controller`.
- ✅ **Waspada N+1.** Saat melakukan iterasi data yang berelasi, apakah saya sudah menggunakan Eager Loading `with()` di `Controller`?
- ✅ **Manfaatkan** `$loop`. Apakah saya membuat variabel counter manual? Hapus dan ganti dengan metadata `$loop` bawaan.

PRINSIP AKHIR

“

Gunakan Blade secara eksklusif untuk Presentation Logic. Jika sebuah kode tidak berhubungan langsung dengan cara data ditampilkan kepada pengguna, maka kode tersebut tidak boleh berada di dalam View.

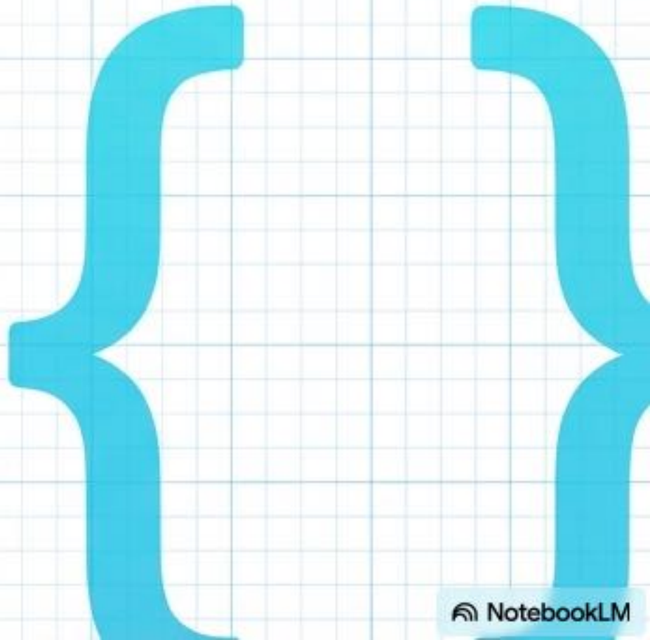
Menulis kode yang bersih bukan hanya tentang sintaks yang rapi, melainkan tentang membangun arsitektur yang dapat bertahan seiring berkembangnya skala aplikasi.


Laravel 11 Blade Control Structures and Looping
Blade Directives
Conditional Structures
@if, @elseif, @else
@unless (negative if)
@isset (check defined and not null)
@empty (check if string/array/null)
@switch, @case
Looping Mechanisms
@foreach (standard iteration)
@forelse (loop with empty fallback)
@for (traditional loop)
@while (conditional loop)
$loop Variable Metadata
index (starts at 0)
iteration (starts at 1)
remaining (items left)
count (total items)
first (boolean)
last (boolean)
Database Optimization
N+1 Problem Prevention
Eager Loading (with() method)
Minimize queries in views
Best Practices
Blade for display logic only
Business logic in Controllers/Services
Clean syntax over pure PHP tags
Use @forelse for empty data handling
Reusable components (@include)
Comparison: Blade vs Standard PHP
Syntax: @ symbol vs curly braces
Readability: Cleaner view files
Helpers: Automatic $loop variable

Panduan Visual Arsitektural

Implementasi Arsitektur MVC di Laravel 11

Blueprint & Studi Kasus: Modul Mahasiswa



Controller & Model

Tempat eksekusi logika bisnis dan interaksi database.

View (Blade)

Fokus mutlak pada tampilan. Dilarang keras menjalankan query database di lapisan ini.

Separation of Concerns: Menjaga kode tetap rapi, aman, dan dapat diskalakan.

Alur Permintaan (The Request Flow)

Siklus hidup data dari interaksi pengguna hingga tampilan akhir.

Tahap 1: Route Setup

routes/web.php

```
Route::get('/mahasiswa', [MahasiswaController::class, 'index']);
```

URL yang diakses oleh pengguna.

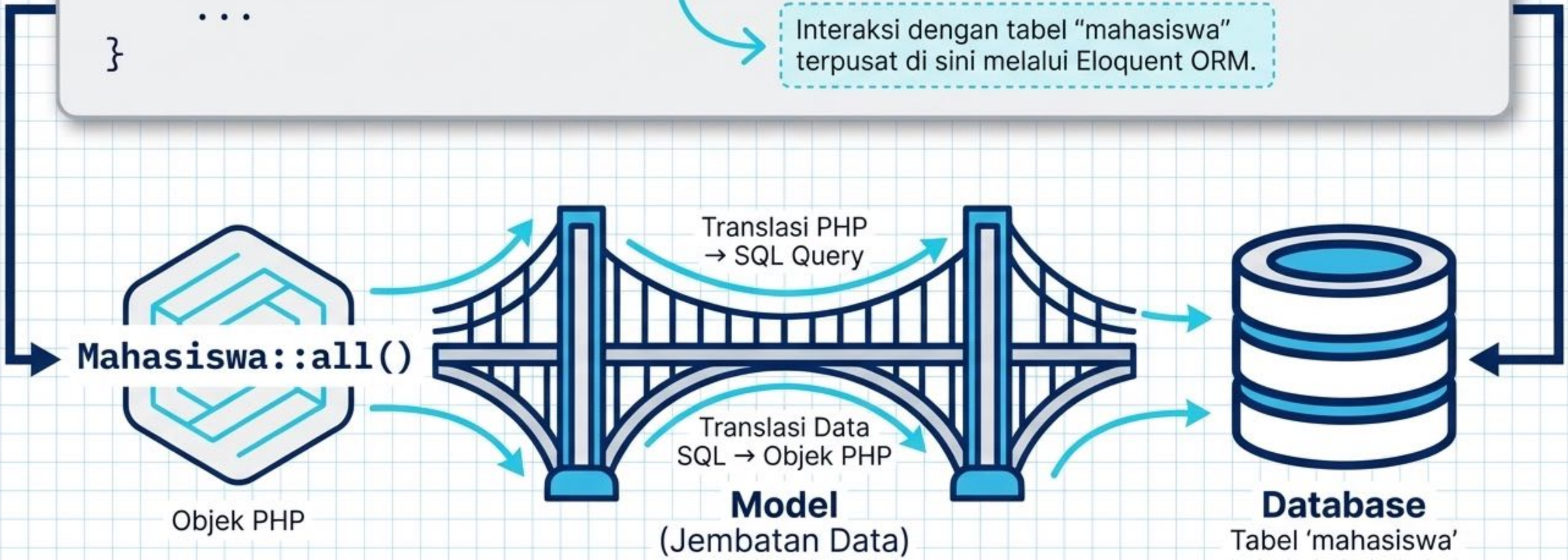
Mengarahkan permintaan ke method index di dalam MahasiswaController.

Tahap 2: Definisi Model

app/Models/Mahasiswa.php

```
class Mahasiswa extends Model {
    ...
}
```

Interaksi dengan tabel “mahasiswa” terpusat di sini melalui Eloquent ORM.



Tahap 3: Controller (Otak dari Logika)

app/Http/Controllers/MahasiswaController.php

```
public function index() {
    $mahasiswa = Mahasiswa::with('relasi')->get();
    return view('mahasiswa.index', compact('mahasiswa'));
}
```

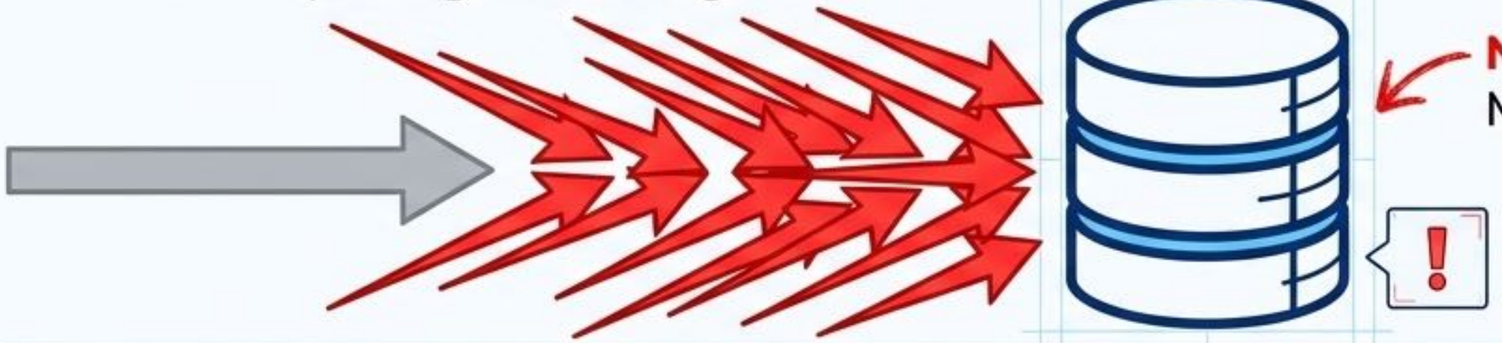
Logika pengambilan data dieksekusi 100% di sini.

Data dikemas dan dikirim ke View dengan bersih.

Praktik Terbaik: Jangan pernah meletakkan logika ini di dalam file Blade.

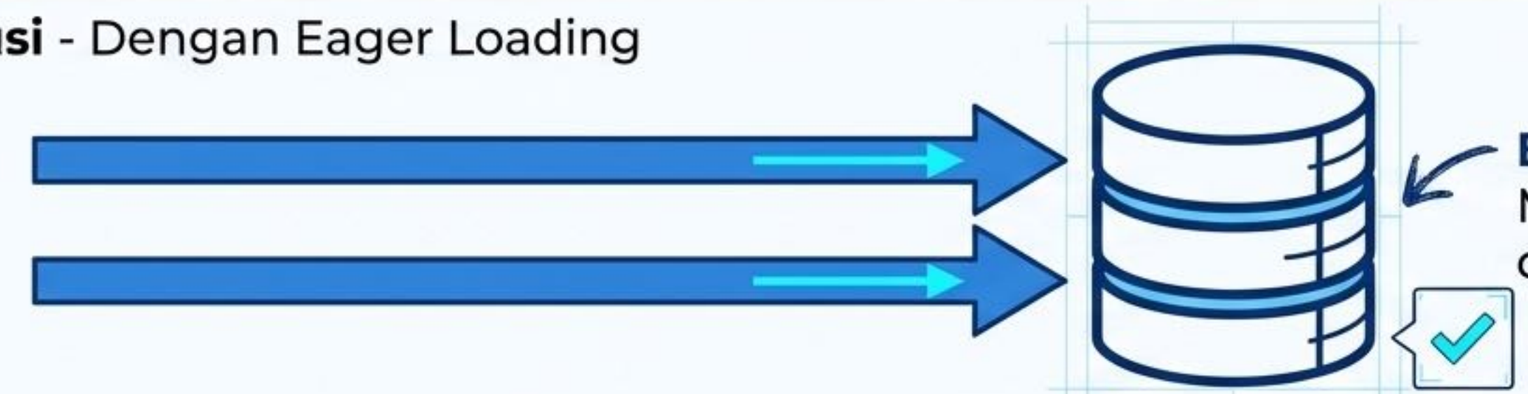
Optimasi Performa: Eager Loading (Deep Dive)

Masalah N+1 - Tanpa Eager Loading



N+1 Queries:
Menguras Performa

Solusi - Dengan Eager Loading



Eager Loading with():
Mengemas permintaan dalam 2 query efisien.

Kesimpulan: Penggunaan **Mahasiswa::with(...)** wajib diterapkan jika ada relasi tabel untuk memastikan skalabilitas.

Transisi Data Terisolasi

Setelah data dikirim melalui fungsi **view()**, **Controller** melepaskan kendali. View sekarang hanya bertugas merender data tersebut tanpa memikirkan dari mana atau bagaimana data itu didapat.

Tahap 4: View (Arsitektur Blade vs PHP Tradisional)

PHP Tradisional

```php
<?php foreach($mahasiswa as $mhs): ?>
    <p><?php echo $mhs->nama; ?></p>
<?php endforeach; ?>
```

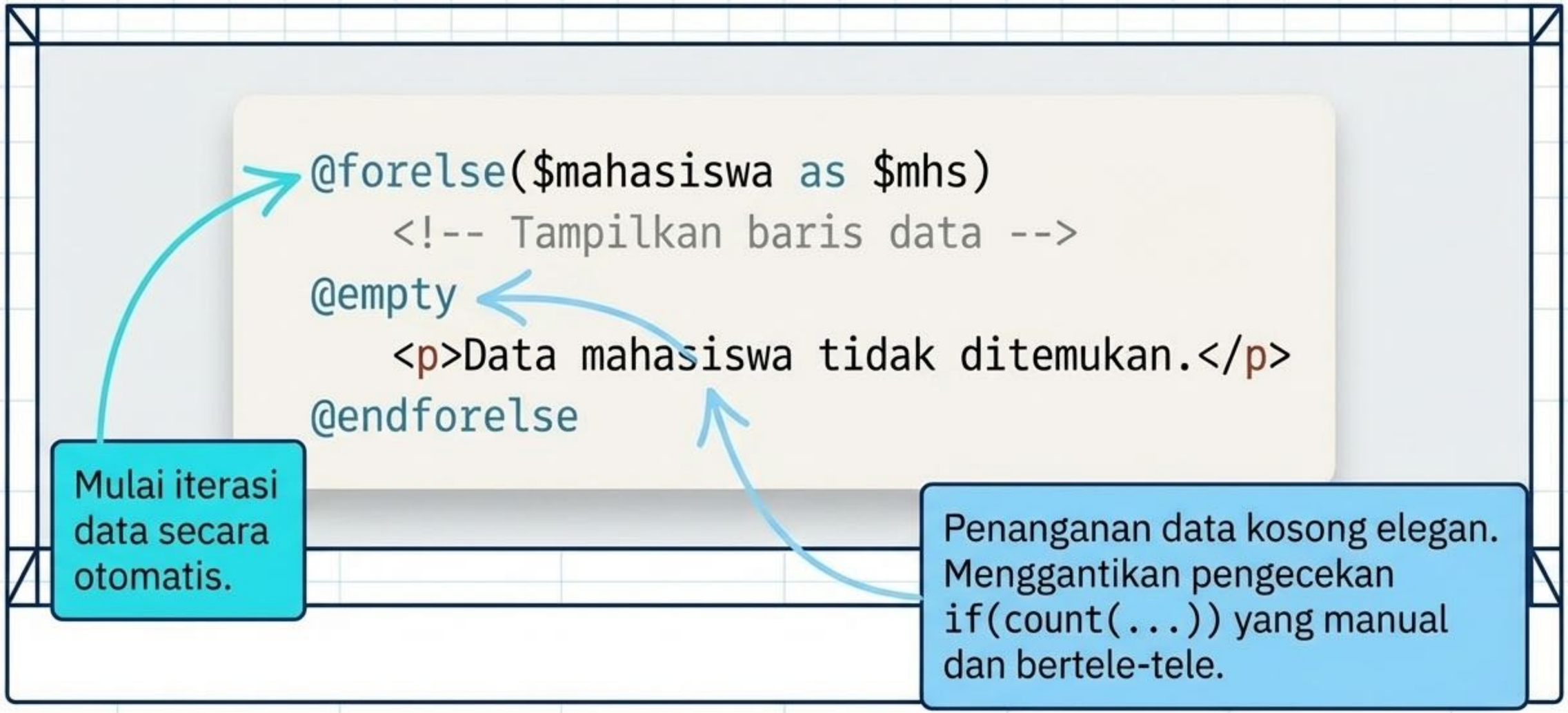
resources/views/mahasiswa/index.blade.php

Arsitektur Blade

```
@foreach($mahasiswa as $mhs)
    <p>{{ $mhs->nama }}</p>
@endforeach
```

Direktif Blade (@ dan {{ }}) menjaga struktur HTML tetap elegan, bersih, dan jauh lebih mudah dibaca daripada tag PHP standar.

Blade Deep-Dive: Penanganan Data Kosong



Blade Deep-Dive: Metadata Iterasi

No.	Nama Mahasiswa
1	Budi Santoso
2	Siti Aminah
3	Ahmad Rizki

```
<td>{{ $loop->iteration }}</td>
<td>{{ $mhs->nama }}</td>
```

Metadata Iterasi Otomatis: Menghasilkan nomor urut (dimulai dari 1) tanpa perlu mendeklarasikan variabel $counter manual di luar loop.

Matriks Perbandingan: Anti-Pattern vs Praktik Terbaik

✕ Anti-Pattern	✓ Praktik Terbaik Laravel 11
Menjalankan **Mahasiswa::get()** di dalam **index.blade.php**.	Query selesai dieksekusi seluruhnya di **MahasiswaController**.
Menggunakan **if + foreach** untuk mengecek ketersediaan data.	Menggunakan direktif **@forelse** dan **@empty**.
Membiarkan relasi diload di dalam looping (Menyebabkan **N+1**).	Menggunakan Eager Loading **with()** sejak di Controller.

Ringkasan Eksekutif Arsitektur

- **Separation of Concerns Terjaga:** Logika bisnis dan UI terisolasi rapi.
- **Kode View Bersih:** Pemanfaatan sintaks Blade untuk keterbacaan tingkat tinggi.
- **Penanganan Edge-Case Elegan:** Pemanfaatan metadata loop dan pengkondisian kosong bawaan.
- **Performa Optimal:** Perlindungan dari N+1 Query Problem melalui Eager Loading.

Blueprint ini adalah standar implementasi untuk ekosistem Laravel 11 yang dapat diskalakan.

Struktur Kontrol di PHP

Struktur kontrol digunakan untuk mengatur alur eksekusi program.

Jenis-jenis:

- if, else, elseif
- switch
- match (fitur PHP 8)

```php
if ($nilai > 75) {
    echo "Lulus";
} else {
    echo "Tidak Lulus";
}
```

Struktur Kontrol di Laravel (Blade)

Laravel menggunakan **Blade Template Engine**.

Blade Directive Lainnya:

- **@if, @elseif, @else, @unless**
- **@isset** (Mengecek apakah variabel terdefinisi)
- **@empty** (Mengecek apakah variabel kosong.)
- **@switch ... @case**

```
@if ($nilai > 75)
    <p>Lulus</p>
@else
    <p>Tidak Lulus</p>
@endif
```

```
@isset($user)
  <p>Nama: {{ $user->name }}</p>
@endisset

@empty($posts)
  <p>Tidak ada post tersedia.</p>
@endempty
```

```
@unless($isVerified)
  <div>Akun Anda belum terverifikasi.</div>
@endunless
```

Perulangan di PHP

Untuk mengulangi blok kode berdasarkan kondisi.:

- **for**
- **while**
- **do...while**
- **foreach**

```php
foreach ($users as $user) {
    echo $user->name;
}
```

```php
<?php

for ($i = 1; $i <= 5; $i++) {
    echo "Perulangan ke-$i <br>";
}

$j = 1;
while ($j <= 5) {
    echo "Perulangan ke-$j <br>";
    $j++;
}

$k = 1;
do {
    echo "Perulangan ke-$k <br>";
    $k++;
} while ($k <= 5);

?>
```

Perulangan di Laravel (Blade)

Larvel Loop Direcvtive :

- **@foreach**
- **@for**
- **@while**
- **@forelse**

```
@foreach ($users as $user)
    <li>{{ $user->name }}</li>
@endforeach

@forelse ($posts as $post)
    <li>{{ $post->title }}</li>
@empty
    <p>Tidak ada postingan.</p>
@endforelse
```

Loop Helpers di Blade

Saat menggunakan loop, tersedia variabel **$loop**:

- **$loop->index (mulai dari 0)**
- **$loop->iteration (mulai dari 1)**
- **$loop->first**
- **$loop->last**
- **$loop->count**
- **$loop->remaining** (Sisa iterasi)

```blade
@foreach ($items as $item)
    <p>{{ $loop->index }} - {{ $item }}</p>
@endforeach
```

```blade
@foreach($users as $user)
  <tr>
    <td>{{ $loop->iteration }}</td> <!-- Nomor urut -->
    <td>{{ $user->name }}</td>
    @if($loop->first)
      <td>Ini baris pertama</td>
    @endif
    @if($loop->last)
      <td>Ini baris terakhir</td>
    @endif
  </tr>
@endforeach
```

Loop Helpers di Blade

Properti yang tersedia:

- $loop->index: Indeks dimulai dari 0.
- $loop->iteration: Iterasi dimulai dari 1.
- $loop->remaining: Sisa iterasi.
- $loop->count: Total item.
- $loop->first: true jika iterasi pertama.
- $loop->last: true jika iterasi terakhir.

Studi Kasus Singkat

- **Tampilkan daftar produk dengan pengecekan stok**

```
@foreach ($products as $product)
    <p>{{ $product->name }}</p>
    @if ($product->stock > 0)
        <span>Stok tersedia</span>
    @else
        <span>Stok habis</span>
    @endif
@endforeach
```

CONTOH

Controller: Mengambil data dari database

```php
// app/Http/Controllers/UserController.php

namespace App\Http\Controllers;

use App\Models\User;

class UserController extends Controller
{
    public function index()
    {
        $users = User::all(); // Ambil semua data user dari database
        return view('users.index', compact('users'));
    }
}
```

CONTOH

Route: Mengarahkan URL ke Controller

```php
// routes/web.php

use App\Http\Controllers\UserController;

Route::get('/users', [UserController::class, 'index']);
```

CONTOH

Blade View: Menampilkan data dengan loop

```blade
<!-- resources/views/users/index.blade.php -->

<!DOCTYPE html>
<html>
<head>
    <title>Daftar Pengguna</title>
</head>
<body>
    <h1>Daftar Pengguna</h1>

    <ul>
        @foreach ($users as $user)
            <li>{{ $loop->iteration }}. {{ $user->name }} - {{ $user->email }}</li>
        @endforeach
    </ul>
</body>
</html>
```

CONTOH

Handling Data Kosong dengan @forelse

```
1 <ul>
2     @forelse ($users as $user)
3         <li>{{ $user->name }}</li>
4     @empty
5         <li>Tidak ada pengguna terdaftar.</li>
6     @endforelse
7 </ul>
```

CONTOH

Implementasi

```php
// app/Http/Controllers/UserController.php
public function index() {
  $users = User::all();
  $role = 'admin';
  return view('users.index', compact('users', 'role'));
}
```

```blade
//resources/views/users/index.blade.php

@extends('layouts.app')

@section('content')
  @if($role === 'admin')
    <h1>Daftar Admin</h1>
  @else
    <h1>Daftar Pengguna</h1>
  @endif

  <table class="table">
    @forelse($users as $user)
      <tr>
        <td>{{ $loop->iteration }}</td>
        <td>{{ $user->name }}</td>
        <td>
          @if($user->is_active)
            <span class="badge bg-success">Aktif</span>
          @else
            <span class="badge bg-danger">Non-aktif</span>
          @endif
        </td>
      </tr>
    @empty
      <tr>
        <td colspan="3">Tidak ada pengguna.</td>
      </tr>
    @endforelse
  </table>
@endsection
```

Tips & Best Practices

- Gunakan Blade untuk logika tampilan, bukan logika bisnis.
- Gunakan **@forelse** untuk menghindari kondisi kosong.
- Manfaatkan **$loop** untuk kontrol tambahan dalam iterasi.
- Simpan logika kompleks di controller atau **ViewModel**.
- **Minimalkan Logika di View**: Gunakan controller atau service class untuk logika kompleks.
- **Gunakan Direktif Blade**: Hindari PHP murni di view untuk konsistensi.
- **Optimasi Query**: Gunakan eager loading (**with()**) untuk menghindari **N+1** problem saat looping data.
- **Pisahkan Komponen**: Gunakan **@include** atau komponen Blade untuk bagian yang reusable.

Kesalahan Umum

- **N+1 Problem:**

  php

  ```php
  // Hindari ini di controller:
  $posts = Post::all(); // Jika setiap post memiliki relasi user
  ```

  Gunakan:

  php

  ```php
  $posts = Post::with('user')->get();
  ```

- **Logic Berat di View**: Jangan jalankan query database langsung di Blade.