

Book Cover

Daftar Isi

VIEW	3
Memecah File	4
Biasakan Memakai Sub View	5
Penamaan Subview	7
Layout vs Konten	8
Tidak Mencampur PHP dan JS	9
Format Tanggal	10
Format Angka	11
View Composer Terlalu Magic	12

VIEW

View sering dianaktirikan ketika sedang koding karena biasanya disini kita akan lebih banyak menulis *tag* HTML dibanding *real code*. Ketika membahas *clean code*, biasanya contoh yang disertakan selalu dalam bentuk kode PHP. Jarang sekali dibahas tentang View. Mungkin karena View bukan "PHP", tidak ada *logic*, dan jarang dijadikan tolak ukur kemampuan seorang programmer ketika *interview*.

Jika kamu baru belajar Laravel, besar kemungkinan posisimu saat ini adalah "*fullstack*" web developer. Secara sederhana, "*fullstack*" berarti kamu bisa membuat sebuah aplikasi web secara mandiri mulai dari desain basis data, melakukan query SQL, koding *logic* aplikasi di Laravel, hingga membuat tampilan dengan HTML, CSS, dan Javascript.

Jika dibawa ke lingkup Laravel, berarti kamu harus mahir dalam membuat Model (interaksi dengan basisdata), View (tampilan), dan Controller (*logic* aplikasi). Setelah menguasai ketiga elemen itu, kamu bisa mendeklarasikan diri sebagai avatar telah menguasai MVC.

Nah, karena saat ini kamu masih harus berhubungan dengan View, maka posisinya harus kita setarakan dengan yang lain. Sebuah View yang clean sama pentingnya dengan Controller dan Model yang clean. Bahkan View harusnya bisa lebih diprioritaskan karena "membersihkan" View jauh lebih mudah dilakukan dibanding membersihkan Controller atau Model.

Ketika kerja tim, dimana ada senior dan junior programmer, biasanya sang senior (orang yang punya wewenang mereview kode) lebih mentoleransi View yang tidak rapi dibanding Controller atau Model. Kita harus mengubah pembiaraan ini. Semua aspek MVC sama pentingnya. View juga bagian dari kode, dia berhak mendapatkan perlakuan yang sama baiknya darimu.


Memecah File

Salah satu kemampuan yang harus dikuasai untuk menulis kode yang *friendly* adalah keberanian untuk memecah kode atau *file*.

Di awal proyek, semua masih terlihat rapi. Kodenya masih sedikit. Seiring berjalannya waktu, ada penambahan fitur disana-sini, tambal sulam bug di kanan dan di kiri. Kode yang awalnya masih terlihat dalam satu layar sekarang harus di-*scroll* berkali-kali untuk melihat keseluruhan isinya.

Programmer yang baik tahu kapan harus mulai memecah file ketika jumlah barisnya sudah mulai membengkak dan berpotensi susah dikelola dikemudian hari.

Biasakan Memakai Sub View

Kita ambil contoh halaman dashboard berikut ini. 

Pada umumnya, tampilan di atas akan diimplementasi menjadi file blade seperti ini:

```
@extends('layout')

@section('content')
    <h1>Statistik Laporan</h1>
    <section>
        Filter
        ...
    </section>
    <section>
        Grafik
        ...
    </section>
    <section>
        Tabel
        ...
    </section>
@endsection
```

Sekarang mari kita coba untuk memecahnya menjadi *sub view*. Bagaimana caranya?

Secara kasat mata, kita bisa melihat ada tiga komponen utama yang menyusun halaman dashboard di atas:

1. Filter
2. Grafik
3. Tabel

Kode yang baik adalah kode yang mencerminkan kebutuhan fungsional aplikasinya. Oleh karena itu, setelah mengetahui komponen penyusun dari sebuah halaman, maka langkah berikutnya adalah membuat **sub view** untuk masing-masing komponen tersebut. Setelah itu, kita cukup memanggil **@include** dari view utama.

```
@extends('layout')

@section('content')
    <h1>Statistik Laporan</h1>
    @include('filter')
    @include('grafik')
    @include('tabel')
@endsection
```

Penamaan Subview

- Diawali underscore.

Layout vs Konten

View utama untuk mengatur layout, subview untuk merender konten.

Tidak Mencampur PHP dan JS

Mencampur kode PHP dan Javascript akan mengurangi readability dan kemampuan editor/IDE untuk menganalisis kode.

Format Tanggal

Y-m-d untuk mesin, d-m-Y untuk manusia.

Format Angka

Rata kanan jika di dalam tabel. `number_format()`;

View Composer Terlalu Magic

Variable yang di-inject dari View Composer sulit untuk di-trace dari mana asalnya. Manfaatkan pemanggilan secara eksplisit menggunakan View Injection.