

Model View Controller (MVC)

Nama : Bayu wahyu pambudi

Universita : Universita Negeri Surabaya

Kelompok : Kelompok 5

Link Repository: <https://gitlab.com/bayu.22140/pw1-bast7-bayu>

MVC (Model-View-Controller) adalah pola desain yang digunakan dalam pengembangan aplikasi web untuk memisahkan komponen logika aplikasi menjadi tiga bagian utama: Model, View, dan Controller. Model bertanggung jawab untuk mengelola data dan logika bisnis, berinteraksi dengan database, serta memproses aturan yang berkaitan dengan aplikasi. View berperan dalam menyajikan data dari Model kepada pengguna dalam format yang ramah pengguna, seperti HTML atau JSON, tanpa melibatkan logika bisnis. Controller bertindak sebagai penghubung antara Model dan View, di mana Controller menerima input dari pengguna, memprosesnya menggunakan Model, dan mengarahkan hasilnya ke View untuk ditampilkan.

Pola ini memiliki beberapa tujuan utama dalam pengembangan aplikasi. Salah satunya adalah untuk memudahkan pemeliharaan kode dengan memisahkan tanggung jawab setiap komponen, sehingga setiap bagian dapat diperbarui atau diperbaiki tanpa memengaruhi komponen lainnya. Selain itu, MVC mendukung skalabilitas, memungkinkan aplikasi dikembangkan secara modular dan memudahkan penambahan fitur baru. Pemisahan antara logika dan tampilan juga mempermudah desainer dan pengembang bekerja secara paralel tanpa saling mengganggu. Selain itu, MVC mendukung reusability, di mana komponen-komponen tertentu dapat digunakan kembali di bagian lain aplikasi atau proyek yang berbeda, mempercepat proses pengembangan secara keseluruhan. Pola MVC ini digunakan secara luas dalam kerangka kerja web modern seperti Laravel, Django, dan ASP.NET karena kemampuannya meningkatkan efisiensi dan kolaborasi dalam pengembangan aplikasi.

Model dan perannya dalam MVC

Model dalam arsitektur MVC (Model-View-Controller) adalah komponen yang menangani semua logika bisnis, pengelolaan data, dan interaksi dengan database atau sumber data lainnya. Model bertanggung jawab untuk mewakili struktur data dan aturan yang mengendalikan bagaimana data tersebut dapat diubah atau diproses dalam aplikasi.

Peran utama Model dalam MVC adalah:

- **Pengelolaan Data:** Model menyimpan, mengambil, dan memperbarui data dari sumber seperti database. Ini termasuk operasi CRUD (Create, Read, Update, Delete) yang menjadi inti dari pengelolaan data dalam aplikasi.

- **Logika Bisnis:** Model mengimplementasikan aturan bisnis dan logika yang berkaitan dengan data. Misalnya, jika aplikasi memiliki aturan tertentu terkait validasi, perhitungan, atau proses transformasi data, semua ini akan diatur oleh Model.
- **Pemisahan Tanggung Jawab:** Dengan menjaga logika dan data di dalam Model, komponen lain, seperti View dan Controller, tidak perlu terlibat dalam hal-hal tersebut. Hal ini menciptakan pemisahan yang jelas antara data, tampilan, dan pengelolaan input pengguna.
- **Menyediakan Data untuk View dan Controller:** Model bertindak sebagai sumber informasi yang digunakan oleh Controller dan View. Ketika pengguna mengirimkan permintaan melalui Controller, Controller akan berkomunikasi dengan Model untuk mengambil data yang diperlukan atau memproses data yang diberikan, dan kemudian memberikan hasilnya kepada View untuk ditampilkan.
- **Integrasi dengan Sumber Data Eksternal**:** Selain berinteraksi dengan database lokal, Model juga dapat berinteraksi dengan API eksternal atau layanan pihak ketiga lainnya untuk mengambil atau menyinkronkan data.

Secara singkat, Model adalah tulang punggung dari aplikasi dalam arsitektur MVC karena berfungsi sebagai pusat pengelolaan data dan logika bisnis yang memastikan data disajikan dengan benar dan diatur sesuai dengan aturan yang telah ditentukan.

View dan perannya dalam MVC

View dalam arsitektur MVC (Model-View-Controller) adalah komponen yang bertanggung jawab untuk menyajikan data kepada pengguna dalam format yang mudah dipahami. View berfungsi sebagai antarmuka visual aplikasi, menampilkan hasil dari data yang diproses oleh Model melalui Controller. Meskipun View menampilkan data, ia tidak menangani logika bisnis atau pengelolaan data—itu merupakan tanggung jawab Model.

Peran utama View dalam MVC adalah:

- **Menampilkan Data:** View bertanggung jawab untuk menampilkan data yang dikirim oleh Model melalui Controller. Data ini biasanya diubah menjadi format yang mudah diakses oleh pengguna, seperti HTML, CSS, dan JavaScript dalam konteks aplikasi web.
- **Memisahkan Presentasi dari Logika:** Dalam MVC, View hanya fokus pada presentasi data, tidak terlibat dalam logika bisnis atau pengelolaan data. Pemisahan ini memudahkan pemeliharaan aplikasi karena desain tampilan dapat diubah tanpa mempengaruhi logika aplikasi, dan sebaliknya.
- **Berinteraksi dengan Pengguna:** View berfungsi sebagai titik interaksi antara pengguna dan aplikasi. Misalnya, pengguna memasukkan data melalui form atau mengklik tombol, dan View menampilkan hasil dari interaksi tersebut. Data yang dimasukkan akan dikirim ke Controller untuk diproses, dan hasilnya kemudian dikirim kembali ke View untuk ditampilkan.
- **Dinamis dan Responsif:** View dapat dibuat dinamis dengan menampilkan data secara real-time, memperbarui konten berdasarkan masukan pengguna, atau memanipulasi tampilan sesuai dengan kebutuhan tanpa harus memuat ulang seluruh halaman. Hal ini sering dilakukan dengan bantuan JavaScript atau teknologi front-end lainnya.

- **Pemisahan Desain dan Pengembangan:** Dengan membiarkan View menangani aspek presentasi, desainer dapat fokus pada antarmuka dan pengalaman pengguna, sementara pengembang berfokus pada logika bisnis dan pengelolaan data di dalam Model dan Controller. Ini memungkinkan kolaborasi yang lebih efisien antara tim desain dan pengembangan.

Secara keseluruhan, View dalam MVC adalah komponen yang memastikan data dari Model ditampilkan dengan benar dan menarik di sisi pengguna, sambil memisahkan aspek visual dari logika bisnis, sehingga mempermudah pengelolaan dan pemeliharaan aplikasi.

Controller dan perannya dalam MVC

Controller dalam arsitektur MVC (Model-View-Controller) adalah komponen yang bertindak sebagai penghubung antara Model dan View. Controller berfungsi untuk menerima input dari pengguna, memprosesnya dengan bantuan Model, dan kemudian mengarahkan hasilnya ke View untuk ditampilkan. Controller mengatur alur logika aplikasi dan bertanggung jawab atas interaksi pengguna dan pemrosesan data di balik layar.

Peran utama Controller dalam MVC adalah:

- **Mengelola Input Pengguna:** Controller menerima permintaan atau input dari pengguna, baik melalui antarmuka pengguna (seperti form, tombol, atau URL) atau melalui API. Controller kemudian memutuskan tindakan apa yang harus diambil berdasarkan input tersebut.
- **Mengatur Alur Aplikasi:** Controller bertindak sebagai pengendali logika alur aplikasi. Ini berarti Controller menentukan bagaimana permintaan pengguna ditangani, apakah perlu mengambil data dari Model, memproses data, atau mengirim hasil ke View untuk ditampilkan.
- **Berkomunikasi dengan Model:** Setelah menerima input pengguna, Controller akan berinteraksi dengan Model untuk mengambil atau memodifikasi data. Model kemudian melakukan operasi logika bisnis yang diperlukan dan memberikan data kembali ke Controller.
- **Mengirim Data ke View:** Setelah Controller mendapatkan hasil dari Model, ia akan memproses data tersebut (jika diperlukan) dan mengirimkannya ke View agar dapat ditampilkan kepada pengguna dalam format yang sesuai. Controller memutuskan data apa yang harus ditampilkan dan dalam konteks apa.
- **Memisahkan Tugas:** Dengan Controller memisahkan logika interaksi pengguna dari Model dan View, setiap komponen dalam MVC dapat fokus pada tugasnya masing-masing. Hal ini membuat pengembangan aplikasi lebih modular, mudah dipelihara, dan skalabel.
- **Pemrosesan Permintaan HTTP:** Dalam aplikasi web, Controller sering bertanggung jawab untuk memproses permintaan HTTP, baik GET maupun POST, serta menentukan bagaimana permintaan tersebut direspon, apakah dengan mengirimkan data JSON, halaman HTML, atau mengalihkan pengguna ke halaman lain.

Secara keseluruhan, Controller adalah jembatan yang menghubungkan pengguna dengan aplikasi. Controller menerima input, berinteraksi dengan Model untuk memproses logika

bisnis, dan kemudian mengarahkan hasil tersebut ke View untuk ditampilkan, memastikan aplikasi berjalan sesuai dengan alur yang diharapkan.

Framework yang menggunakan konsep MVC

Salah satu framework pengembangan web yang menggunakan konsep MVC (Model-View-Controller) adalah Laravel, sebuah framework PHP yang sangat populer. Laravel didesain untuk memudahkan pengembangan aplikasi web dengan menyediakan struktur yang jelas berdasarkan pola MVC.

➤ Penjelasan Laravel dalam Konsep MVC:

- **Model di Laravel:**
Laravel menggunakan Eloquent ORM sebagai sistem Model-nya. Eloquent memungkinkan developer untuk berinteraksi dengan database secara lebih intuitif melalui model, tanpa perlu menulis query SQL yang kompleks. Model di Laravel mengelola semua interaksi dengan database, seperti operasi CRUD (Create, Read, Update, Delete), dan juga memproses logika bisnis. Setiap tabel dalam database memiliki model yang berhubungan, dan setiap instance dari model merepresentasikan satu baris dalam tabel.
- **View di Laravel:**
Laravel menggunakan sistem template bernama ****Blade**** untuk menangani View. Blade memungkinkan developer untuk membuat tampilan dinamis menggunakan syntax sederhana yang terintegrasi dengan PHP. View dalam Laravel bertanggung jawab untuk menampilkan data yang dikirim dari Controller. Dengan Blade, developer dapat menggunakan pengkondisian, loop, dan bahkan mewarisi template lain, menjadikan View lebih modular dan mudah dikelola.
- **Controller di Laravel:**
Controller dalam Laravel mengelola logika aplikasi dengan berperan sebagai penghubung antara Model dan View. Ketika pengguna mengirim permintaan, Controller akan mengambil input tersebut, memprosesnya dengan bantuan Model, lalu menentukan data apa yang harus dikirim ke View. Laravel juga mendukung ****Route**** yang membantu mengarahkan permintaan HTTP ke controller yang sesuai.

➤ Fitur Tambahan Laravel yang Mendukung MVC:

- **Routing yang Fleksibel:**
Laravel menyediakan routing yang mudah digunakan, mengarahkan permintaan pengguna (GET, POST) ke controller yang tepat. Developer dapat menentukan rute mana yang mengarahkan ke controller tertentu.
- **Middleware:**

Laravel mendukung penggunaan middleware untuk mengatur logika yang harus dijalankan sebelum atau sesudah permintaan mencapai controller. Ini berguna untuk mengelola autentikasi, logging, atau penanganan input.

- Validasi dan Otentikasi:
Laravel menyediakan fitur validasi data dan otentikasi bawaan yang mempermudah implementasi fitur keamanan dalam aplikasi web.
- Migration dan Seeder:
- Laravel memudahkan pengelolaan database melalui migration dan seeder yang memungkinkan developer untuk memodifikasi skema database dan mengisi tabel dengan data awal.

➤ Kelebihan Laravel:

- Kemudahan Penggunaan:
- Laravel memiliki dokumentasi yang lengkap dan mudah dipahami oleh pengembang dari berbagai tingkat keahlian.
- Modular dan Skalabel:
Dengan pemisahan tugas berdasarkan MVC, aplikasi yang dibangun dengan Laravel dapat lebih mudah dikembangkan dan dipelihara.
- Dukungan Komunitas yang Kuat:
Laravel memiliki komunitas besar yang menyediakan berbagai paket dan library yang siap digunakan.

Dengan konsep MVC, Laravel memungkinkan pengembang untuk membuat aplikasi web yang terstruktur dengan baik, memisahkan logika bisnis, antarmuka pengguna, dan pengelolaan data, sehingga pengembangan dan pemeliharaan aplikasi menjadi lebih efisien dan terorganisir.

Parsing data

1. Menambahkan Route dengan URL `/parse-data`

```
Route::get('/parse-data/{nama_lengkap}/{email}/{jenis_kelamin}',  
[App\Http\Controllers\ParsingDataController::class, 'parseData']);
```

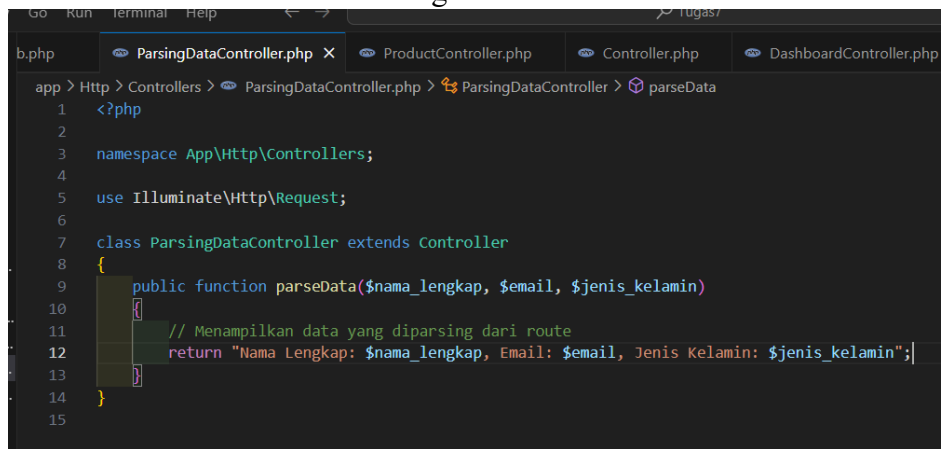
Udah ada di php artisan route:list

```
POST|HEAD parse-data/{nama_lengkap}/{email}/{jenis_kelamin} .....
```

2. Membuat Controller `ParsingDataController`

```
PS C:\Users\ASUS\Documents\MSIB\e_commerce\pw1-bast7-bayu\Tugas7> php artisan make:controller ParsingDataController
```

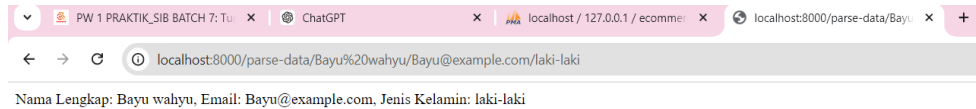
- Tambahkan Metode untuk Parsing Data



```
b.php ParsingDataController.php ProductController.php Controller.php DashboardController.php  
app > Http > Controllers > ParsingDataController.php > ParsingDataController > parseData  
1 <?php  
2  
3 namespace App\Http\Controllers;  
4  
5 use Illuminate\Http\Request;  
6  
7 class ParsingDataController extends Controller  
8 {  
9     public function parseData($nama_lengkap, $email, $jenis_kelamin)  
10     {  
11         // Menampilkan data yang diparsing dari route  
12         return "Nama Lengkap: $nama_lengkap, Email: $email, Jenis Kelamin: $jenis_kelamin";  
13     }  
14 }  
15
```

3. Jalankan dan Akses Route `/parse-data`

- Akses Route di Browser



localhost:8000/parse-data/Bayu%20wahyu/Bayu@example.com/laki-laki

Nama Lengkap: Bayu wahyu, Email: Bayu@example.com, Jenis Kelamin: laki-laki