Pemrograman berbasis web 2

konsep mvc & view pada codeigniter 4

Graphical user interface, application

Description automatically generated

**DESKRIPSI MATAKULIAH**

**Pada mata kuliah ini mahasiswa belajar mendesain dan membuat web statis maupun dinamis dengan menerapkan Framework yang ada, seperti Bootstrap dan Codeigniter**

**CAPAIAN PEMBELAJARAN**

**Mahasiswa dapat menjelaskan konsep MVC, serta mampu menjelaskan View dalam pembuatan web menggunakan framework CI-4**

**DOSEN PENGAMPU**

**Wachid Darmawan, M.Kom**

**SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER**

**(STMIK) WIDYA PRATAMA**

KONSEP MVC & View PADA CODEIGNITER 4

**CAPAIAN PEMBELAJARAN**

**Mahasiswa dapat menjelaskan konsep MVC, serta mampu menjelaskan View dalam pembuatan web menggunakan framework CI-4.**

# Materi

**MATERI PEMBELAJARAN**

* **Struktur Direktori Codeigniter 4**
* **Mengenal Konsep MVC Codeigniter 4**
* **Router di Codeigniter 4**
* **Memahami View**
* **Latihan View di CodeIgniter**

**MATERI**

Sudah berhasil install CI-4 kan? Selanjutnya, kita akan mulai dengan mengenal struktur direktorinya (folder yang ada) terlebih dahulu, dilanjutkan membahas konsep MVC yang ada di Codeigniter 4.

# **Text Description automatically generated with medium confidenceStruktur Direktori Codeigniter 4**

Silahkan buka teks Editor VS Code, lalu pilih **File 🡪 Open Folder**. Cari folder 📁 **CI4** (atau folder installasi CodeIgnitir 4 yang ada di dalam folder **htdocs**) kemudian **Select Folder**. Berikut beberapa direktori dan file yang ada di folder CI4:

* 📁 **app:** folder ini akan berisi kode dari aplikasi yang kita kembangkan;
* 📁 **public:** folder ini berisi file yang bisa diakses oleh publik, seperti file index.php, robots.txt, favicon.ico, ads.txt, dll;
* 📁 **tests:** folder ini berisi kode untuk melakukan testing dengan PHPunit;
* 📁 **vendor:** folder ini berisi library yang dibutuhkan oleh aplikasi, isinya juga termasuk kode core dari system CI.
* 📁 **writable:** folder ini berisi file yang ditulis oleh aplikasi, nantinya, kita bisa pakai untuk menyimpan file yang di-upload, logs, session, dll.

Jika digambarkan struktur direktori/folder yang ada kurang lebih akan nampak seperti berikut:

Diagram

Description automatically generated

Selanjutnya kita bahas **file-file** yang ada di direktori CI4:

* 📄 **.env**: adalah file yang berisi variabel environment yang dibutuhkan oleh aplikasi.
* 📄 **.gitignore:** adalah file yang berisi daftar nama file dan folder yang akan diabaikan oleh Git.
* ⚙️ **build:** adalah script untuk mengubah versi codeigniter yang digunakan. Ada versi release (stabil) dan development (labil).
* 📜 **composer.json:** adalah file JSON yang berisi informasi tentang proyek dan daftar library yang dibutuhkannya. File ini digunakan oleh Composer sebagai acuan.
* 📜 **composer.lock:** adalah file yang berisi informasi versi dari libraray yang digunakan aplikasi.
* ©️ **license.txt:** adalah file yang berisi penjelasan tentang lisensi Codeigniter;
* 📜 **phpunit.xml.dist:** adalah file XML yang berisi konfigurasi untuk PHPunit.
* 📖 **README.md:** adalah file keterangan tentang codebase CI. Ini biasanya akan dibutuhkan pada repo github atau gitlab.
* ⚙️ **spark:** adalah program atau script yang berfungsi untuk menjalankan server, generate kode, dll.

Kemudian, coba kita fokus ke folder 📁 **app**, di sana ada beberapa folder dan file yang perlu kita ketahui:

* 📁 **Config:** adalah folder yang berisi konfigurasi untuk aplikasi;
* 📁 **Controllers:** adalah folder yang berisi kode Controller dari aplikasi;
* 📁 **Database:** adalah folder yang berisi kode untuk migrasi database;
* 📁 **Filters:** adalah folder yang berisi kode untuk filter atau middleware;
* 📁 **Helpers:** adalah folder yang berisi kode untuk fungsi-fungsi helper;
* 📁 **Language:** adalah folder yang berisi kamus bahasa untuk aplikasi;
* 📁 **Libraries:** adalah folder yang berisi library tambahan untuk aplikasi;
* 📁 **Models:** adalah folder yang berisi kode untuk model;
* 📁 **ThridParty:** adalah folder yang berisi library dari pihak ketiga;
* 📁 **Views:** adalah folder yang berisi kode untuk view;
* 📜 **.htaccess:** adalah file yang berisi konfigurasi akses untuk web server;
* 📜 **Common.php:** adalah file yang berisi definisi fungsi untuk menindih fungsi core dari CI.
* 📜 **index.html:** adalah file placeholder untuk mencegah direct access.

Untuk penjelasan folder atau file lainya nanti akan berbarengan dengan perkuliahan.

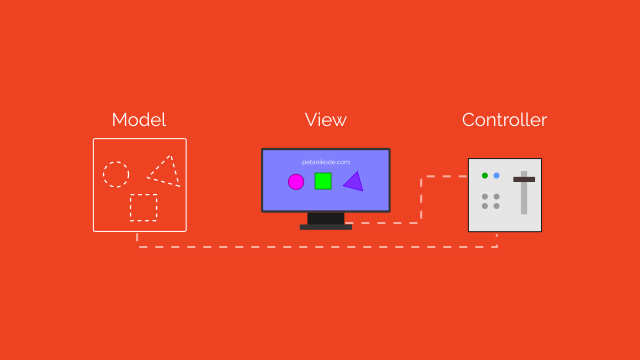
# **Mengenal Konsep MVC Codeigniter 4**

Codeigniter sejak awal menganut konsep MVC. Karena itu, CodeIgniter begitu cepat dikenal programer. karena, implementasi MVC di CI cukup mudah dipahami oleh programer.

# **Apa itu MVC?**

MVC adalah singkatan dari *Model–View–Controller*. MVC merupakan sebuah pola desain arsitektur yang umum digunakan dalam pengembangan aplikasi. Tujuanya agar kode program lebih konsisten dan tersetruktur. Sehingga kita akan mudah kolaborasi dengan tim.

Nah, MVC ini adalah pola penulisan kode yang umum dipakai.. dimana kode untuk *Model* ditempatkan kedalam folder yang sama, begitu juga dengan kode untuk *View* dan *Controller*. Coba perhatikan gambar berikut:



Gambar tersebut menyatakan bagaimana hubungan kode pada MVC.

# **Apa itu Model?**

Model adalah kode yang bertugas untuk membuat pemodelan data. Kadang juga dipakai untuk pemodelan bisnis. Model bisa mengakses data dari Database dan juga sumber lainnya.

Intinya: Kalau berkaitan tentang data, itu tugasnya model.

# **Apa itu View?**

View merupakan kode yang bertugas untuk membuat tampilan aplikasi. Yang perlu kita ingat: View fokusnya untuk membuat tampilan aplikasi, bukan yang lain. Kita tidak boleh query data dari view (meskipun itu bisa dilakukan). Kadang juga kita akan membuat sedikit logika di dalam view, seperti logika untuk menampilkan dan menghilangkan elemen tertentu. Jadi, apa yang kita lihat di aplikasi, itu adalah kode dari View.

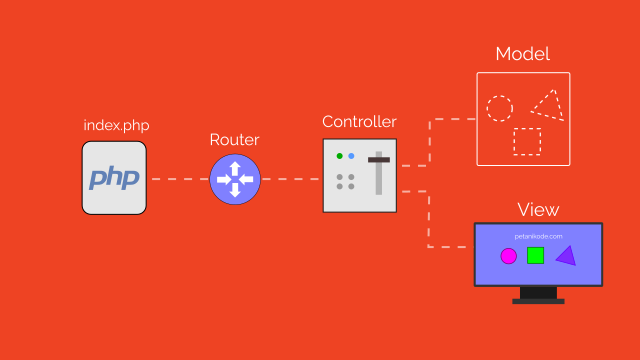
# **Apa itu Controller?**

Jadi, Controller inilah yang bertugas menyatukan Model dan View. Setiap ada request. Controller akan bertanggung jawab untuk membalas request tersebut. Jika itu request untuk menyimpan data ke database, maka Controller harus memanggil Model yang bertanggung jawab untuk data tersebut. Lalu meminta model untuk menyimpannya ke database. Jika itu request untuk view data, maka Controller akan mencari kode View yang bertanggung jawab untuk menampilkan data tersebut. Kadang juga Controller akan membutuhkan beberapa library dan helper untuk memproses request.

Intinya: Controller bertanggung jawab untuk membalas *request* dari client. Tapi sebelum itu, ada **router** di depan controller yang bertanggung jawab untuk menentukan Controller mana yang akan dieksekusi.

# **Router di Codeigniter 4**

Coba buka file **app/Config/Routes.php**. File ini adalah file router yang bertugas untuk menentukan, Controller mana yang akan bertanggung jawab pada request tertentu.



Sebagai contoh: Perhatikan kode ini.

$routes->get('/', 'Home::index');

Saat ada request ke halaman root (**/**) atau alamat domain dari aplikasi, maka Controller yang akan bertugas adalah **Home** dan method yang akan dipakai adalah **index**.

Intinya: Router akan memberikan tugas pada Controller tertentu untuk membalas sebuah request.

# **Memahami View**

Veiw merupakan sebuah halaman web atau bagian header, footer atau sidebar yang menyusun sebuah halaman. Sebuah view tidak bisa dipanggil secara langsung, harus dipanggil melalui sebuah controller. Untuk membuat sebuah view letakkan file yang akan dijadikan view letakkan kedalam **app/Views**. Contoh sebuah fie view dengan nama **BlogView.php**.

<html>

<head>

<title>My Blog</title>

</head>

<body>

<h1>Welcome to my Blog!</h1>

</body>

</html>

Untuk menampilkan sebuah view dapat menggunakan perintah **echo** dengan memanggil fungsi **view()**. Contoh dapat dilihat digambar dibawah ini:

echo view(‘name’);

Untuk menampilkan banyak view dapat menggunakan **echo view()** secara berulang dibaris berikutnya seperti dibawah ini:

public function index(){

$data = [

'page\_title' => 'Your title'

];

echo view('header');

echo view('menu');

echo view('content', $data);

echo view('footer');

}

Jika kita membuat sebuah folder didalam folder Views ketika kita akan memanggilnya gunakan perintah sebagai berikut:

echo view(‘nama\_direktori/name’);

Untuk menambahkan data dinamis yang akan dikirimkan kedalam sebuah view, data dapat ditampung kedalam sebuah array. Contoh mengirimkan data dinamis ke view:

$data = [

'title' => 'My title',

'heading' => 'My Heading',

'message' => 'My Message'

];

echo view('blogview', $data);

Mengirim sebuah nested array kedalam sebuah view

public function index(){

$data = [

'todo\_list' => ['Clean House', 'Call Mom', 'Run Errands'],

'title' => "My Real Title",

'heading' => "My Real Heading"

];

echo view('blogview', $data);

}

<html>

<head>

<title><?= $title ?></title>

</head>

<body>

<h1><?= $heading ?></h1>

<h3>My Todo List</h3>

<ul>

<?php foreach ($todo\_list as $item):?>

<li><?= $item ?></li>

<?php endforeach;?>

</ul>

</body>

</html>

CodeIgniter mendukung sistem tata letak yang sederhana, namun sangat fleksibel, yang membuatnya mudah untuk menggunakan satu atau beberapa tata letak halaman dasar di seluruh aplikasi Anda. Tata letak mendukung bagian konten yang dapat disisipkan dari tampilan apa pun yang sedang dirender. Anda dapat membuat tata letak yang berbeda untuk mendukung halaman arsip blog satu kolom, dua kolom, dan banyak lagi. Tata letak tidak pernah dirender secara langsung. Sebaliknya, Anda merender tampilan, yang menetapkan tata letak yang ingin diperluas.

Untuk membuat sebuah layout kita akan menggunakan function **renderSection()**. Fungsi ini akan digunakan menjadi placeholder untuk sebuah halaman yang kita buat. Contoh sebagai berikut:

<!doctype html>

<html>

<head>

<title>My Layout</title>

</head>

<body>

<?= $this->renderSection('content') ?>

</body>

</html>

Untuk memasukkan sebuah template yang sudah disiapkan sebelumnya kedalam sebuah halaman gunakan kode **$this->extend()**. Contoh sebagai berikut:

<?= $this->extend('default') ?>

<?= $this->section('content') ?>

<h1>Hello World!</h1>

<?= $this->endSection() ?>

# **Latihan View di CodeIgniter**

* 1. Lakukan instalasi CodeIgniter4 kedalam htdocs (boleh menggunakan cara manual atau menggunakan composer) kemudian berinama projek **pertemuan3**.
  2. Jalankan local server development CodeIgniter4 dengan cara menjalankan **php spark serve**. (Pastikan posisi direktori ada didalam folder projek).

Text

Description automatically generated

* 1. Buka folder menggunakan VS Code atau teks editor yang lain.
  2. Copy paste file **env** (buat backup), kemudian file **env - Copy** di rename menjadi **.env** selanjutnya edit bagian **production** menjadi **development** dan juga bagian **baseURL**. Seperti dibawah ini:

Text

Description automatically generated

* 1. Buka file **Routes.php** yang ada di folder **app/Config**. Kemudian ubah pada bagian **default routes** menjadi seperti dibawah ini:

Text

Description automatically generated

* 1. Download file bahan praktikum (**starter-template.zip**) file ada di classroom.
  2. Buat sebuah file dengan nama **template.php** pada folder **app/Views**. Copy code yang ada di file **starter-template.txt** (dari hasil download) kedalam file **template.php**
  3. Pada bagian **<h1>Bootstrap… s/d …document. </p>**

A screenshot of a computer

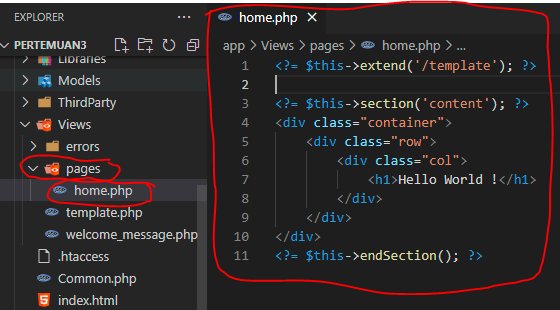
Description automatically generated with medium confidence

* 1. Kemudian ganti menjadi kode seperti dibawah ini:

A screenshot of a computer

Description automatically generated with medium confidence

* 1. Buat sebuah folder didalam folder **Views** dengan nama **pages**. Kemudian buat sebuah file baru dengan nama **home.php** kemudian isikan kode dibawah ini:



* 1. Buat sebuah file di **app/Controller** dengan nama **Pages.php** kemudian isikan dengan source code dibawah ini:

Text

Description automatically generated

* 1. Buka localserver development pada browser kemudian arahkan kedalam controller pages. **http://localhost:8080/pages/** kemudian akan tampil halaman seperti bawah ini:

Graphical user interface, application, website

Description automatically generated

**RANGKUMAN**

* Codeigniter 4, menggunakan konsep MVC. Request pada Codeigniter pertamakali akan diterima oleh file **public/index.php** kemudian diteruskan ke router dan diproses oleh Controller.
* View layouts dapat kita gunakan untuk mempermudah dalam pembuatan sebuah view.

**TES FORMATIF**

1. Rubah tampilan Home menjadi seperti berikut:

Graphical user interface, text, website

Description automatically generated

1. Untukk halaman **about** tangkap variabel yang ada pada variabel data lihat panduan pada **poin k** (**Pages.php**). Sehingga tampil dihalaman **about**, tampilanya seperti ini:

Graphical user interface, website

Description automatically generated

1. Untuk halaman **contact** gunakanlah **array** **didalam array** pada saat mengambil data dari page (***referensi array:*** [***https://www.w3schools.com/php/php\_arrays.asp***](https://www.w3schools.com/php/php_arrays.asp)), tampilanya seperti ini:

Graphical user interface, application, website

Description automatically generated

1. Yang dikumpulkan 1 folder **pertemuan3** yg ada di **htdocs** (**C:\xampp\htdocs\pertemuan3**) dan **screenshot** hasil dari Tes Formatif yang sudah dilakukan, semua file dan folder tersebut di kompres dalam bentuk ZIP, contoh: **21.240.0789-Wachid.zip**

**DAFTAR PUSTAKA**

Dokumentasi CI 4: <https://codeigniter4.github.io/userguide/general/common_functions.html>

Dokumentasi Array: <https://www.w3schools.com/php/php_arrays.asp>

Muhardian, Ahmad. Memahami MVC Pada CodeIgniter 4. Lombok: Petani Kode, 2021.