LAPORAN KONSEP MVC
PEMROGRAMAN BERBASIS PLATFORM

Dosen Pengampu :
Ir. Kartono Pinaryanto S.T., M.Cs.

DIBUAT OLEH :
Alexander Almas Santosa
NIM 225314175

KELAS : E

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA
2024

A. Pendahuluan

Model – View – Controller, atau kerap disingkat MVC adalah pola arsitektur pada perancangan lunak berorientasi objek. Tujuannya adalah memisahkan antara tampilan, data, dan proses.

Perbandingan antara programing prosedural dan MVC

Dalam procedural programming, semua komponen baik tampilan, data, dan proses ditulis dalam satu file seperti di bawah ini. Bisa bekerja, namun akan lebih sulit untuk di maintain, terlebih jika skala projek mulai membesar.

```php
<?php
$conn = mysqli_connect("localhost", "root", "") or die("Koneksi Gagal");
mysqli_select_db($conn, "043040023") or die("database salah");
?>
<!DOCTYPE html>
<html>
<head>
    <title>Daftar Mahasiswa</title>
</head>
<body>
<h2>Daftar Mahasiswa</h2>

<?php $result = mysqli_query($conn, "SELECT * FROM mahasiswa"); ?>

<?php while( $row = mysqli_fetch_assoc($result) ) { ?>
<ul>
    <li><img src="../img/<?php echo $row["gambar"]; ?>"></li>
    <li><?php echo $row["nama"]; ?></li>
    <li><?php echo $row["email"]; ?></li>
    <li><?php echo $row["jurusan"]; ?></li>
    <li><?php echo $row["universitas"]; ?></li>
</ul>
<?php } ?>
```

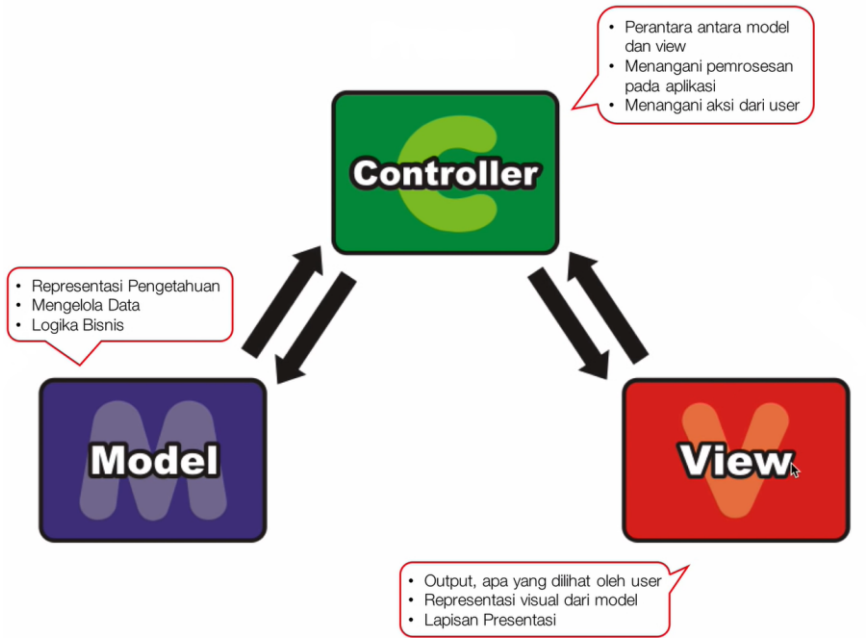
Koneksi DB

Presentasi

SQL Query

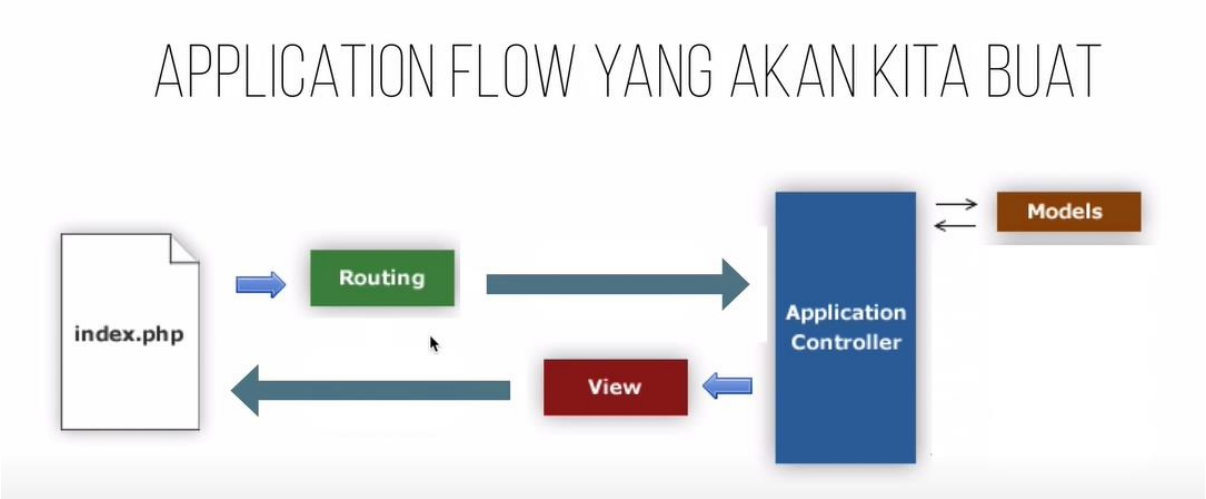
Presentasi

Dengan MVC, kita membagi semua komponen itu dalam tiga kategori seperti dibawah ini:



MVC biasanya lebih dipilih, karena kode yang dihasilkan lebih terstruktur dan terorganisasi, logic dan tampilan terpisah dengan jelas, perawatan kode lebih mudah, penggunaan konsep OOP, dan banyak framework yang support arsitektur MVC ini.

Setelah ini, kita akan mempraktekkan penerapan MVC menggunakan PHP, dan berikut adalah alur aplikasi yang akan kita buat.

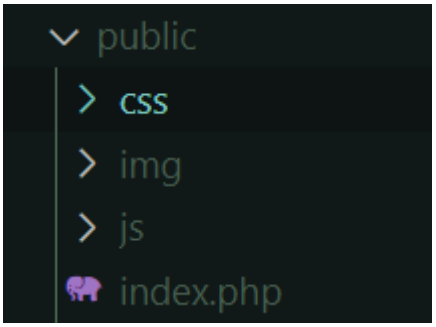


B. Persiapan

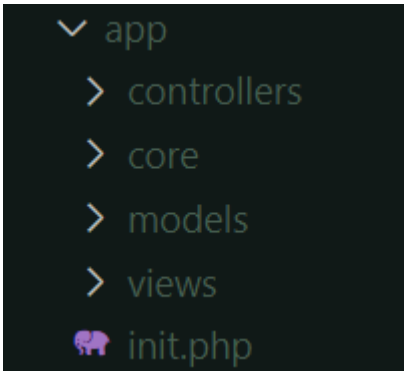
Pembuatan aplikasi dimulai dari fase persiapan, yang dimana kita akan menjalankan sebuah proses bernama bootstrapping. Di sini, kita membuat beberapa folder dan file yang terstruktur agar aplikasi dapat berjalan sesuai keinginan nantinya. Kita mulai dengan menyalakan Apache Server dari XAMPP, dan membuka folder htdocs yang sudah biasa kita gunakan. Dalam folder htdocs, kita buat folder baru yang digunakan sebagai tempat menyimpan aplikasi yang akan kita buat (contoh: phpmvc).

Dalam folder phpmvc ini, kita akan membuat dua folder utama, yaitu folder **public** dan folder **app**.

1. Folder **public**
   Berisi berbagai dokumen yang bisa diakses oleh user biasa. File css, gambar, dan javascript disimpan di sini. Folder public juga menjadi alamat utama dari keseluruhan aplikasi MVC sehingga file index.php disimpan di sini. Berikut adalah struktur dari folder public, sesuai :



2. Folder **app**
   Folder dan file-file utama dari aplikasi MVC diletakkan di sini, baik model, view dan juga controller. Di sini juga terdapat file init.php. Struktur file nya seperti di bawah ini:



   Sekarang menuju ke folder app/core, lalu buat file App.php dan Controller.php, dengan isi seperti di bawah ini:

```php
<?php

class Controller {

}

?>
```

```php
<?php

class App {

    public function __construct() {
        echo 'ok';
    }

}

?>
```

   Kita membuat constructor di class App.

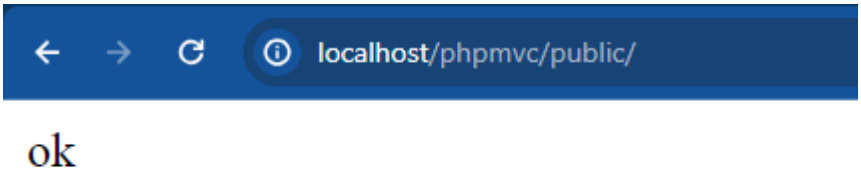
   Juga pastikan untuk membuat file init.php di folder app, yang berisi kode require once untuk file App.php dan Controller.php.

```
phpmvc > app > init.php
<?php
require_once 'core/App.php';
require_once 'core/Controller.php';
?>
```

Melalui requirement init.php, kita bisa menginstansiasikan class App di file index.php seperti di bawah ini:

```php
<?php
require_once '../app/init.php';

$app = new App;
?>
```

Untuk mengetes hasil bootstrapping, kita bisa membuka link localhost/phpmvc/public, dan seharusnya jika berhasil maka 'OK' akan dicetak, seperti di bawah ini. Ini berarti bahwa class App berhasil diinstansiasikan. Dan kita siap untuk masuk tahap berikutnya.



C. Routing

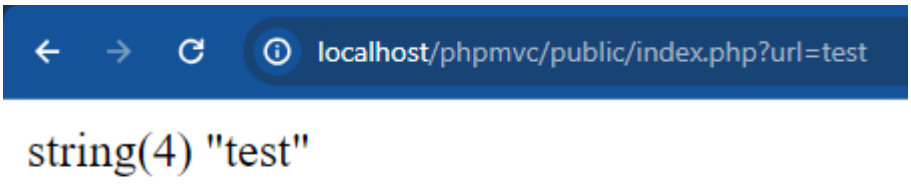
Di tahap ini, kita akan mengelola URL yang dikirimkan. URL akan dikelola dan ditampilkan dengan rapi, atau biasa disebut pretty URL. Harapannya melalui URL, kita bisa memanggil Controller default (misal /home), dan controller lain bisa terpanggil sesuai apa yang dimasukkan ke URL jika ada (misal /about). Selain itu kita juga mengelola agar folder app tidak bisa diakses secara publik.

Pertama tama, kita buat string parser seperti di App.php, dalam class App:

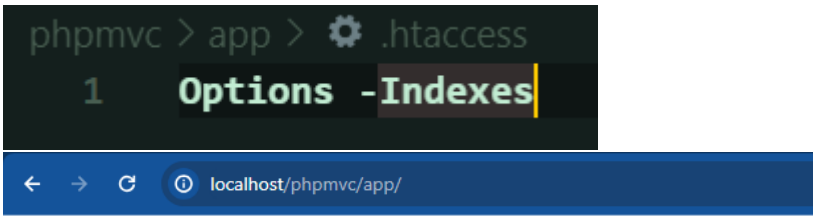
phpmvc > app > core > App.php > ...

```php
<?php

class App {

    public function __construct() {
        $url = $this->parseURL();
    }

    public function parseURL() {
        if(isset($_GET['url'])) {
            $url = $_GET['url'];
            return $url;
        }
    }

}
```

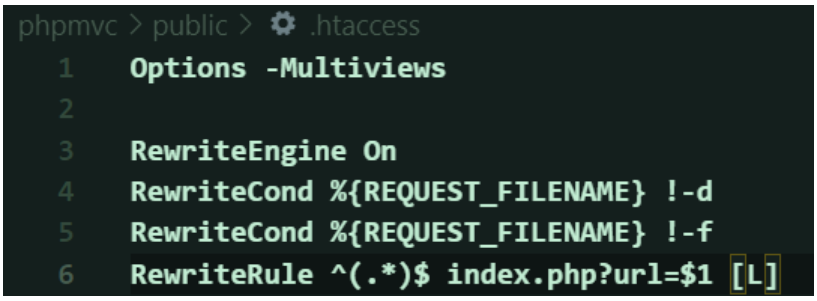
Method parseURL() digunakan untuk mengambil dan memisah-misah URL jadi beberapa bagian. Tapi untuk sementara, kita gunakan hanya untuk mengambil string url terlebih dahulu.



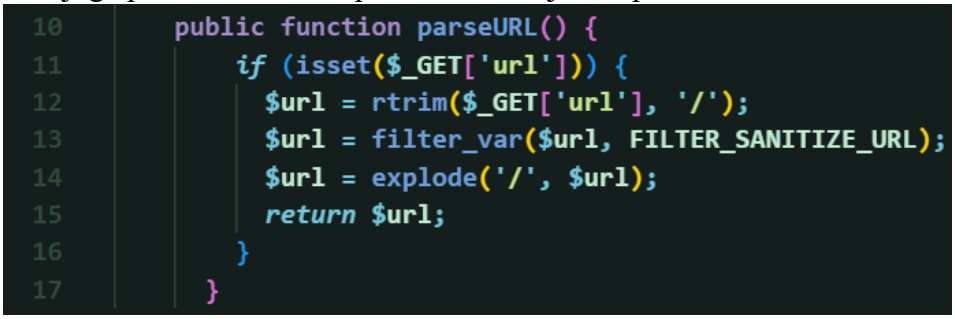
Untuk menulis ulang URL nya, kita perlu membuat file .htaccess, yang digunakan untuk mengubah konfigurasi Apache Server. Disini, kita bisa menggunakan config RewriteEngine. Sebelumnya, kita akan menghapus akses folder app ke publik, dengan cara menggunakan config Options -Indexes. Buat file .htaccess pada folder app, lalu tulis Options -Indexes, seperti di bawah ini.



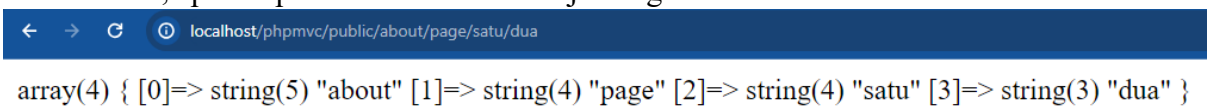
Lalu buat .htaccess lagi pada folder public, untuk rewrite URL nya nanti.



Dan juga perbaiki function parseURL menjadi seperti ini:



Lalu kita tes, apakah parseURL sudah bekerja dengan baik:



About, page, satu, dan dua sudah terbagi ke dalam array masing-masing. Artinya tiap kata di url sudah terbagi sesuai harapan.

D. Controller

Di routing, kita sudah membagi URL menjadi beberapa bagian. Bagian-bagian ini akan kita manfaatkan sebagai referensi controller, method, dan parameter untuk menjalankan page tertentu, sesuai URL nya. Jika tidak ada input apa-apa pada URL, maka controller default lah yang dipanggil.

Dalam kasus aplikasi ini, URL pada section pertama merupakan referensi controller, lalu dilanjutkan ke method, dan sisanya adalah parameter. Contoh linknya ../home/about/20, maka controllernya home, methodnya about, dan parameternya 20.

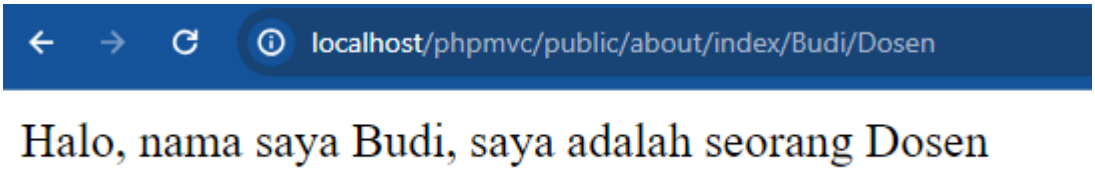
Menggunakan hasil parsing URL dari method di routing, kita dapat membuat link baru yang merujuk ke file yang terkait dengan isi URL, sesuai controller, method, dan variabelnya.

phpmvc > app > core > App.php > App > __construct

```php
<?php

class App {
    protected $controller = 'Home';
    protected $method = 'index';
    protected $params = [];

    public function __construct() {
        $url = $this->parseURL();

        if (isset($url[0])) {
            // Controller
            if( file_exists('../app/controllers/'.$url[0].'.php')) {
              $this->controller = $url[0];
              unset($url[0]);
            }
        }
        require_once '../app/controllers/'.$this->controller.'.php';
        $this->controller = new $this->controller;

        // Method
        if (isset($url[1])) {
            if (method_exists($this->controller, $url[1])) {
            $this->method = $url[1];
            unset($url[1]);
            }
        }
```

```php
        // Parameters
        if (!empty($url)) {
            $this->params = array_values($url);
        }

        // Jalankan controller & method, serta kirimkan params jika ada
        call_user_func_array([$this->controller, $this->method], $this->params);
    }

    public function parseURL() {
        if (isset($_GET['url'])) {
          $url = rtrim($_GET['url'], '/');
          $url = filter_var($url, FILTER_SANITIZE_URL);
          $url = explode('/', $url);
          return $url;
        }
      }
}

?>
```

Constructor sekarang dilengkapi agar bisa membuat link ke dokumen sesuai url yang ditulis dalam format Pretty URL. Contoh penggunaan pretty URL:



Default

Berikut adalah About.php yang digunakan diatas:

phpmvc > app > controllers > About.php > About > page

```php
<?php

class About {
    public function index($nama = 'Almas', $pekerjaan = 'Mahasiswa') {
        echo 'Halo, nama saya '.$nama.', saya adalah seorang '.$pekerjaan;
    }
    public function page() {
        echo 'About/page';
    }
}
```

E. View

Setelah melalui routing dan controller di atas, kita sudah dapat mengelola URL sesuai dengan jalur MVC nya. Sekarang kita akan mengubah struktur view, agar lebih terstruktur dan lebih mudah diakses serta dirawat. Tiap halaman memiliki folder view sendiri, yang berisi index.php sebagai default file nya. Controller digunakan untuk mengarahkan halaman ke folder view ini. Selain itu, baik header dan footer bisa dipisah ke folder templates, agar bisa dimanfaatkan sebagai template untuk file lainnya. Karena kemungkinan, akan ada banyak view dalam satu aplikasi.

Title Document juga bisa disesuaikan dengan bantuan controller. Sehingga meskipun menggunakan template, Titlenya masih bisa di ubah-ubah.

Contoh penggunaan header-footer template:

phpmvc > app > views > home > index.php

```
1
2       <h1>Selamat datang di website.</h1>
3
```

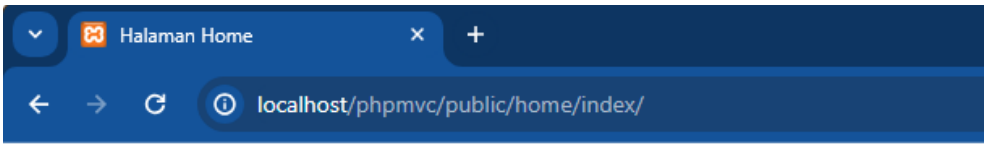
phpmvc > app > views > templates > header.php

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Halaman <?= $data['judul']; ?></title>
7  </head>
8  <body>
```

phpmvc > app > views > templates > footer.php

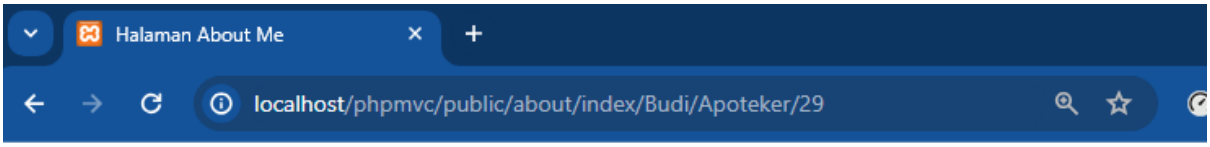
```
1  </body>
2  </html>
```

Contoh hasil view



Selamat datang di website.

Contoh About Me



About Me

Halo, nama saya Budi, umur saya 29 tahun, saya adalah seorang Apoteker.