

LAPORAN PEMROGRAMAN WEB FRAMEWORK

“Konsep Router Pada CodeIgniter”



DISUSUN OLEH :

EDO KRISTIAN YUSAK

E31192060

GOLONGAN C

MANAJEMEN INFORMATIKA

TEKNOLOGI INFORMASI

POLITEKNIK NEGERI JEMBER

2021

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi telah banyak membantu pekerjaan setiap orang, lembaga atau perusahaan dalam menyelesaikan suatu masalah dan menjalankan tugasnya. Ketatnya persaingan teknologi menyebabkan peningkatan kebutuhan akan penggunaan teknologi informasi. Setiap orang, lembaga atau perusahaan yang ada saat ini selalu mengikuti perkembangan teknologi kebutuhan informasi untuk dapat membantu kegiatan operasionalnya dengan menggunakan sistem yang lebih efektif dan efisien dalam menjalankan bisnisnya. Salah satu kegiatan bisnis yang sedang populer saat ini adalah pembuatan dan penggunaan website untuk berbagai keperluan bagi setiap orang, lembaga atau perusahaan dalam mempromosikan penjualan ataupun kebutuhan bisnis dan manajerial.

Dalam perkembangan web Framework CodeIgniter banyak menawarkan kemudahan-kemudahan dalam membangun aplikasi website, karena Framework CodeIgniter sudah tersedia struktur aplikasi yang baik, coding yang standar, fungsi–fungsi dan library yang telah umum digunakan dalam pengembangan sistem. Dengan menggunakan Framework CodeIgniter pembangunan aplikasi dapat langsung fokus kepada business process yang dihadapi tanpa harus berfikir banyak masalah struktur aplikasi, standard coding dll.

1.2 Rumusan Masalah

- 1.2.1 Bagaimana konsep dasar Router pada CI?
- 1.2.2 Bagaimana membuat beberapa router pada CI?

1.3 Tujuan

- 1.3.1 Mahasiswa mampu memahami konsep dasar Router pada CI.
- 1.3.2 Mahasiswa mampu membuat beberapa router pada CI.

BAB II DASAR TEORI

2.1 Router di Codeigniter

Router pada framework codeigniter, memiliki tugas untuk menentukan controller serta method/fungsi yang akan dijalankan ketika pengguna aplikasi mengakses alamat/url tertentu. Ketika kita mengakses <http://localhost/ci> maka akan muncul tampilan welcome to ci, hal tersebut karena default controller mengarah pada file routers.php, anda bisa mengakses file routers.php didalam direktori application/config/routers.php.

Perhatikan pada line : 52 – 54

```
1 $route['default_controller'] = 'welcome';
2 $route['404_override'] = '';
3 $route['translate_uri_dashes'] = FALSE;
```

Keterangan :

- **\$route['default_controller'] = 'welcome'** ini merupakan pengaturan default controller yang otomatis akan dipanggil ketika halaman base_url web diakses, base url disini adalah alamat utama dari web, disitu kita menulis welcome artinya akan mengakses controller welcome, controller welcome adalah controller default yang merupakan bawaan codeigniter, untuk file controller berada di application\controllers, nah pada controller welcome, yang akan dijalankan awal adalah function index, pada function index tersebut menjalankan view welcome_message, dimana file view ini berada pada

direktori application\views, anda bisa mengganti nilai pada nilai default_controller, untuk mengarahkan ke controller tertentu saat base_url diakses

- **\$route['404_override'] = ''** merupakan pengaturan default controller yang akan diakses apabila halaman default controller tidak ditemukan, ataupun sebuah controller lainnya tidak ditemukan.
- **\$route['translate_uri_dashes'] = FALSE**, ini adalah pengaturan yang memperbolehkan anda menggunakan tanda dash (-) pada bagian url, anda bisa menggantinya dengan nilai TRUE, sebagai controller semisal anda memiliki controller dengan nama produk_makanan maka kita dapat mengakses pada urlnya menjadi produk-makanan

Tampilan dari link : <http://localhost/ci/index.php/welcome/index> akan sama dengan tampilan dari link : <http://localhost/ci>. Hal tersebut dikarenakan saat kita mengakses : <http://localhost/ci/index.php/welcome/index>, kita sedang mengakses function index didalam controller welcome.

Jadi untuk mengakses function / method didalam controller, kita perlu menuliskan :

- Base_url : localhost/belajarcodigniter
- Lalu tambahkan index.php
- Lalu tuliskan nama controller
- Lalu tuliskan nama function

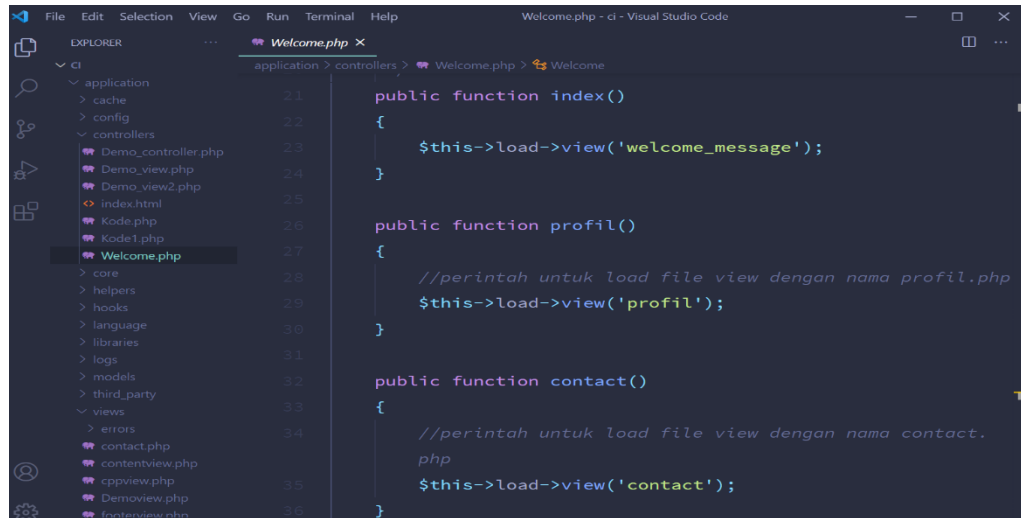
BAB III

HASIL DAN PEMBAHASAN

3.1 Kegiatan Praktikum

3.1.1 Membuat Beberapa Router

Membuat 2 function didalam controller Welcome dibawah function index.



```
public function index()
{
    $this->load->view('welcome_message');
}

public function profil()
{
    //perintah untuk load file view dengan nama profil.php
    $this->load->view('profil');
}

public function contact()
{
    //perintah untuk load file view dengan nama contact.
    php
    $this->load->view('contact');
```

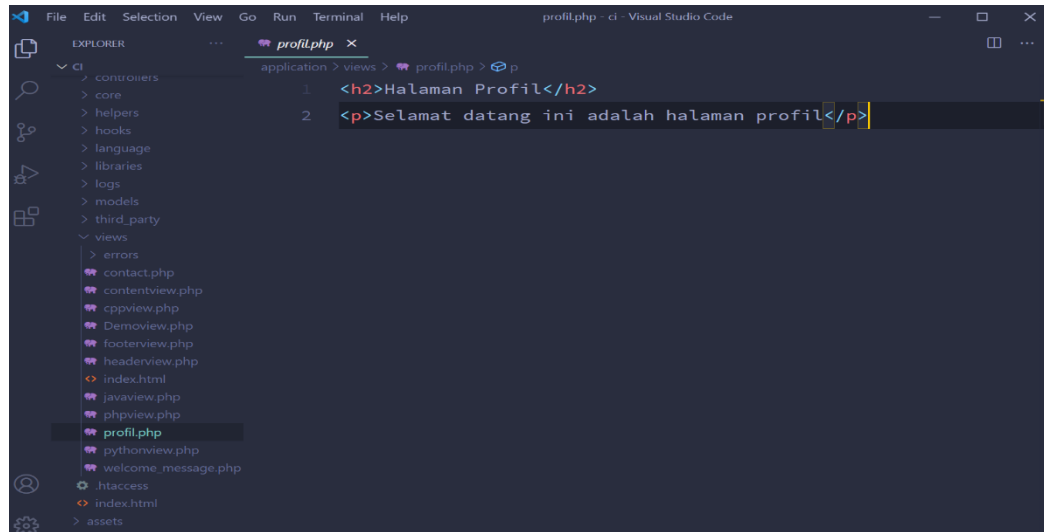
Berikutnya membuat 2 file view, dengan nama profil.php, dan contact.php didalam direktori application/views.

contact.php



```
<h2>Halaman Contact</h2>
<p>Selamat datang ini adalah halaman contact</p>
```

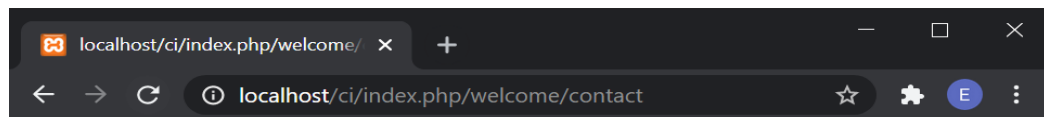
profil.php



```
1 <h2>Halaman Profil</h2>
2 <p>Selamat datang ini adalah halaman profil</p>
```

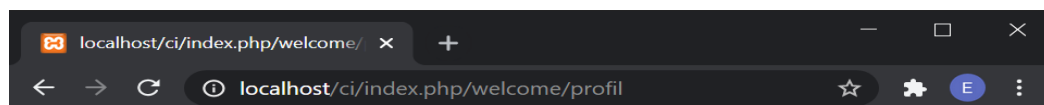
Output saat akses halaman tersebut dengan link :

localhost/ci/index.php/welcome/contact dan
localhost/ci/index.php/welcome/profil



Halaman Contact

Selamat datang ini adalah halaman contact

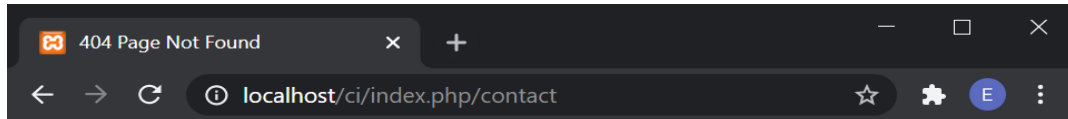


Halaman Profil

Selamat datang ini adalah halaman profil

Halaman akan tampil, karena memang kita langsung mengakses function dalam controllerwelcome dibagian URL. Tetapi apakah bisa kita mengaksesnya dengan alamat :

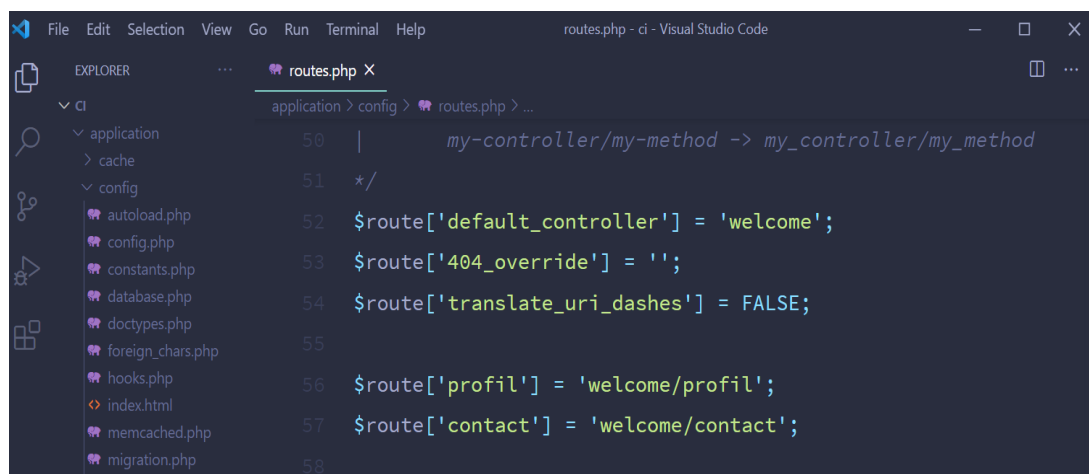
localhost/ci/index.php/contact dan localhost/ci/index.php/profil



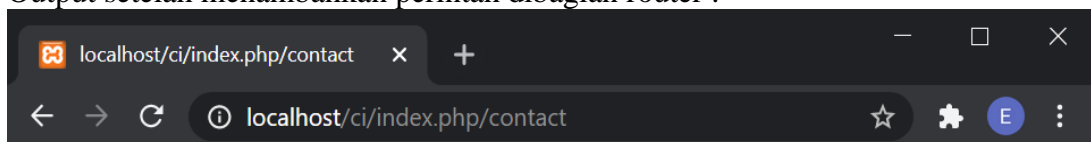
404 Page Not Found

The page you requested was not found.

maka akan muncul tampilan 404 not found. Hal ini dikarenakan dibagian router belum diset, sehingga perlu ditambahkan perintah dibagian di router.



Output setelah menambahkan perintah dibagian router :



Halaman Contact

Selamat datang ini adalah halaman contact

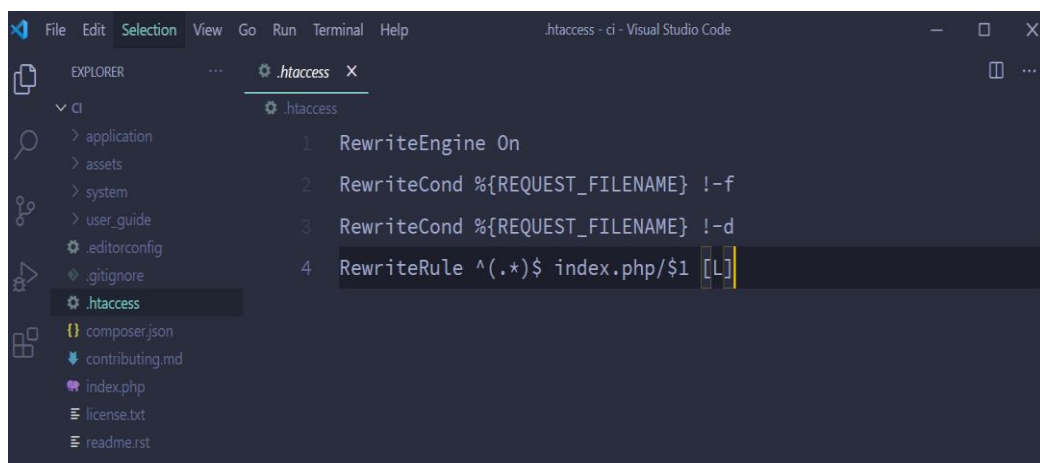
Kita tidak harus menambahkan route di file routers.php, setiap kali anda membuat route baru, karena Codeigniter otomatis mendeteksi route berdasarkan nama controller dan function/method yang dibuat.

3.2 Tugas / Soal

3.2.1 Untuk mengakses file pada CI normalnya link adalah :

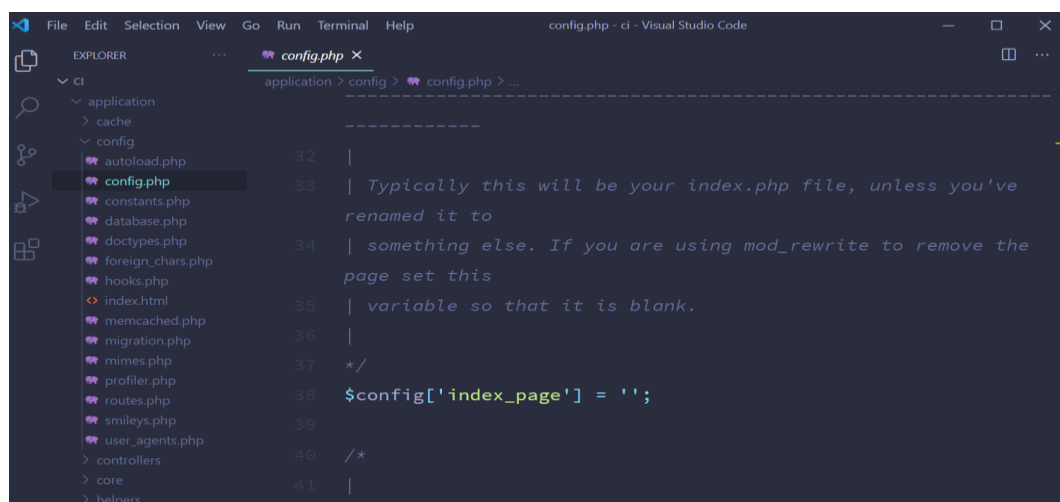
localhost/nama_folder/index.php/controller. Rubahlah agar aksesnya menjadi localhost/ci/controller.

Secara default, file index.php akan dimasukkan ke dalam URL kita akan tetapi kita dapat dengan mudah menghapus file ini menggunakan file .htaccess dengan beberapa aturan sederhana. Berikut adalah contoh file semacam itu, menggunakan metode "negatif" di mana semuanya dialihkan kecuali item yang ditentukan :



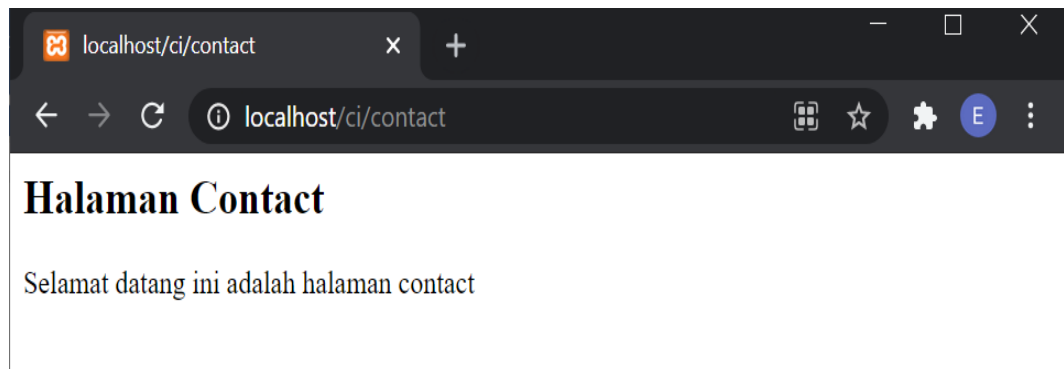
```
File Edit Selection View Go Run Terminal Help .htaccess - ci - Visual Studio Code
EXPLORER
ci
  application
  assets
  system
  user_guide
  editorconfig
  gitignore
  htaccess
  composer.json
  contributing.md
  index.php
  license.txt
  readme.rst
.htaccess
1 RewriteEngine On
2 RewriteCond %{REQUEST_FILENAME} !-f
3 RewriteCond %{REQUEST_FILENAME} !-d
4 RewriteRule ^(.*)$ index.php/$1 [L]
```

Ketika kita telah mengganti namanya menjadi sesuatu yang lain atau menggunakan mod_rewrite untuk menghapus halaman, Selanjutnya kita atur \$config['index_page'] = 'index.php'; menjadi kosong \$config['index_page'] = '';



```
File Edit Selection View Go Run Terminal Help config.php - ci - Visual Studio Code
EXPLORER
ci
  application
  cache
  config
    autoload.php
    config.php
    constants.php
    database.php
    doctypes.php
    foreign_chars.php
    hooks.php
    index.html
    memcached.php
    migration.php
    mimes.php
    profiler.php
    routes.php
    smileys.php
    user_agents.php
  controllers
  core
  helpers
config.php
32 |
33 | Typically this will be your index.php file, unless you've
34 | renamed it to
35 | something else. If you are using mod_rewrite to remove the
36 | page set this
37 | variable so that it is blank.
38 $config['index_page'] = '';
39
40 /*
41 |
```


Setelah kita menambahkan file .htaccess maka saat kita menuliskan pada URL untuk menjalankan programnya cukup dengan `http://localhost/ci/contact`



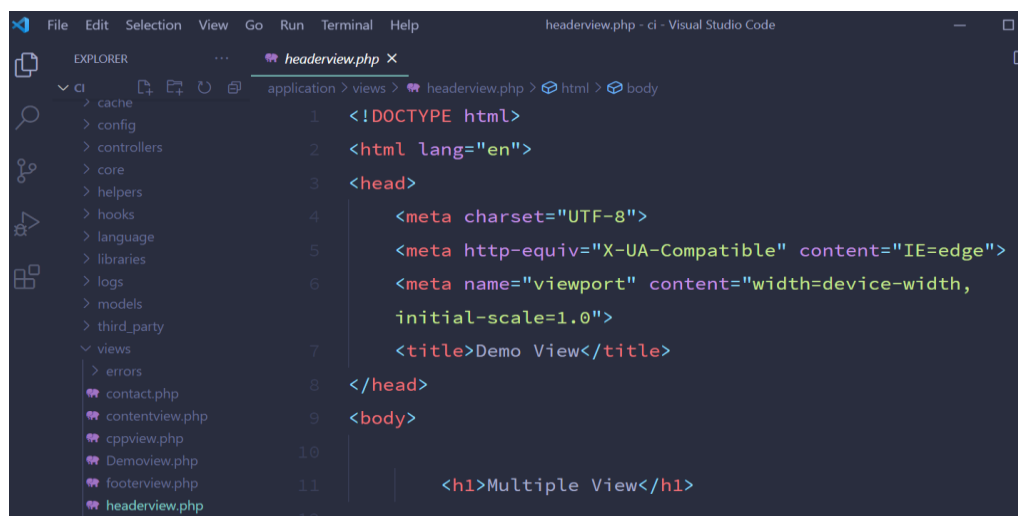
3.2.2 Cobalah kode dibawah ini :

Kode 1 :

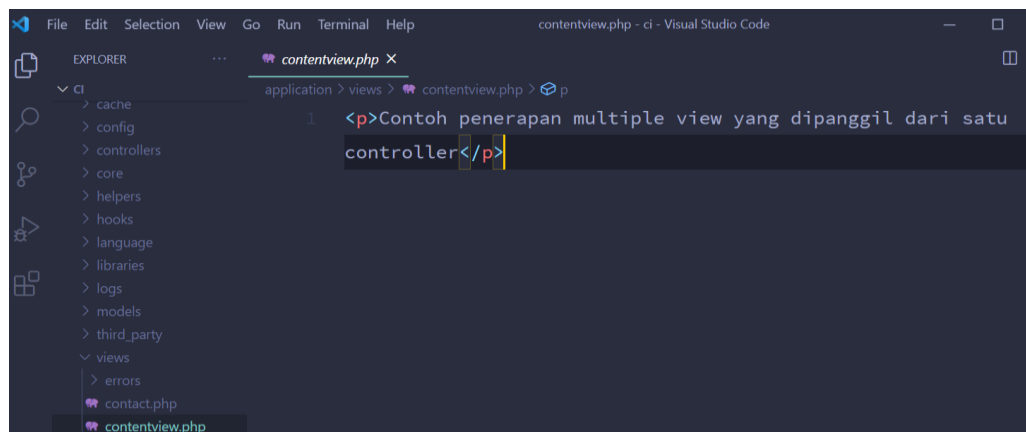
View

Load multiple view

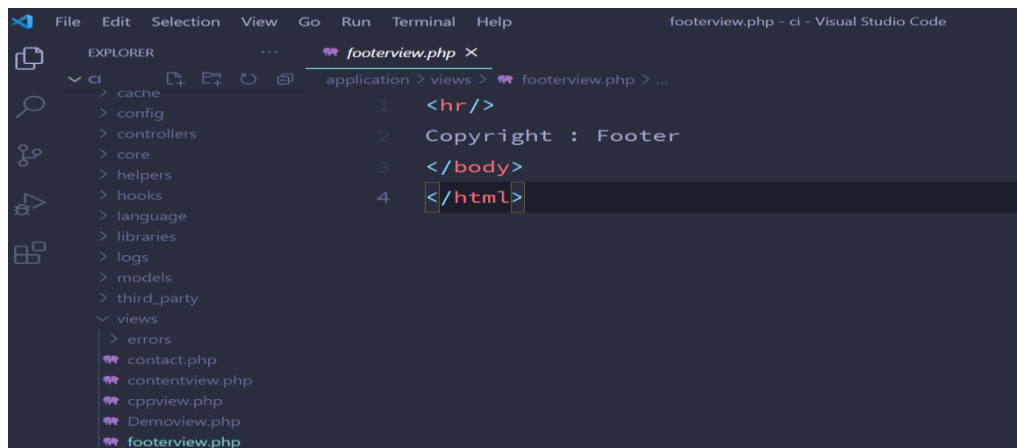
headerview.php



contentview.php

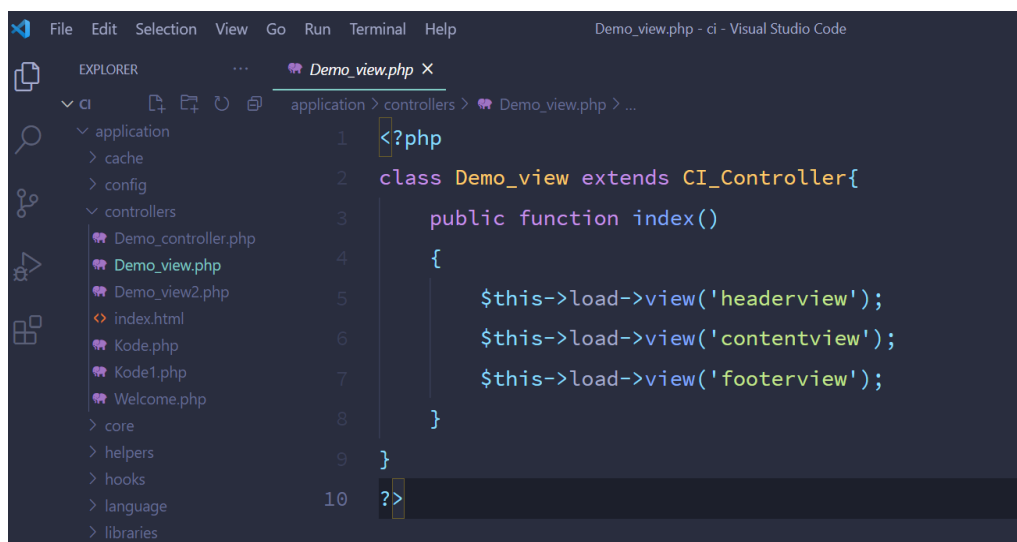


footerview.php



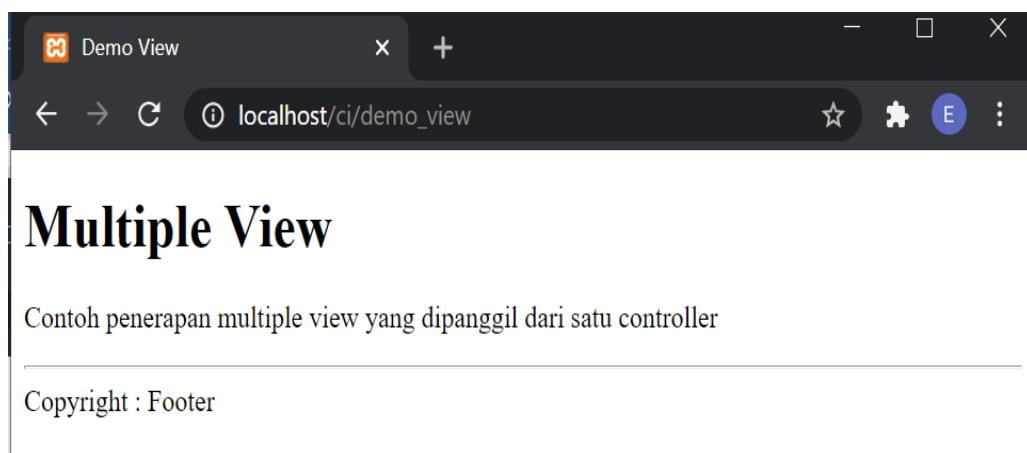
```
1 <hr/>
2 Copyright : Footer
3 </body>
4 </html>
```

Membuat file Demo_view.php pada controller :



```
1 <?php
2 class Demo_view extends CI_Controller{
3     public function index()
4     {
5         $this->load->view('headerview');
6         $this->load->view('contentview');
7         $this->load->view('footerview');
8     }
9 }
10 ?>
```

Output :

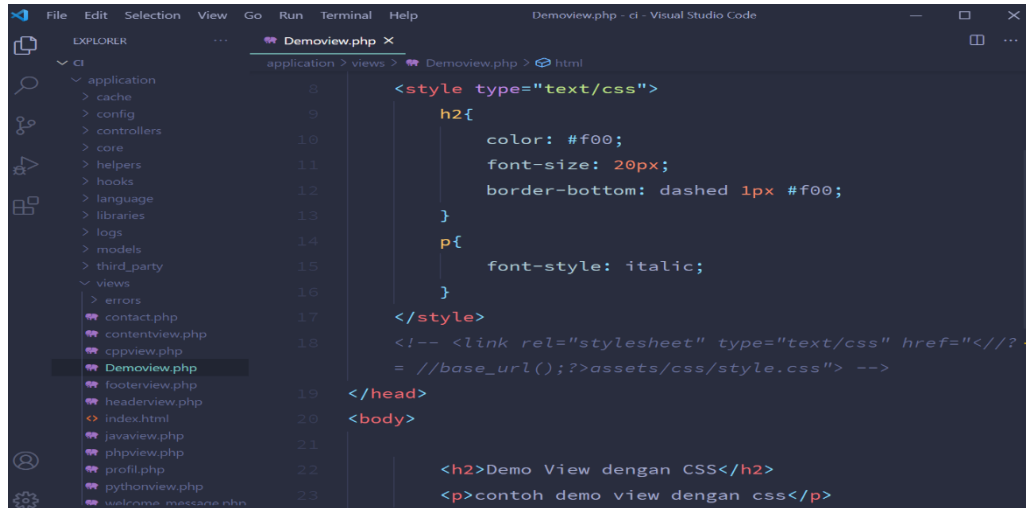


Sama halnya dengan tugas nomer 1 , setelah menambahkan file .htaccess maka pada saat akan menjalankan program kita hanya mengetikkan pada URL <http://localhost/ci/controller> tidak perlu menggunakan index.php lagi karena kita sudah hilangkan pada tugas sebelumnya dan berlaku juga pada tugas berikutnya selama berada pada folder codeigniter yang sama.

Kode 2 :

Menyisipkan CSS ke View 1

Demoview.php



```
File Edit Selection View Go Run Terminal Help
Demoview.php - ci - Visual Studio Code

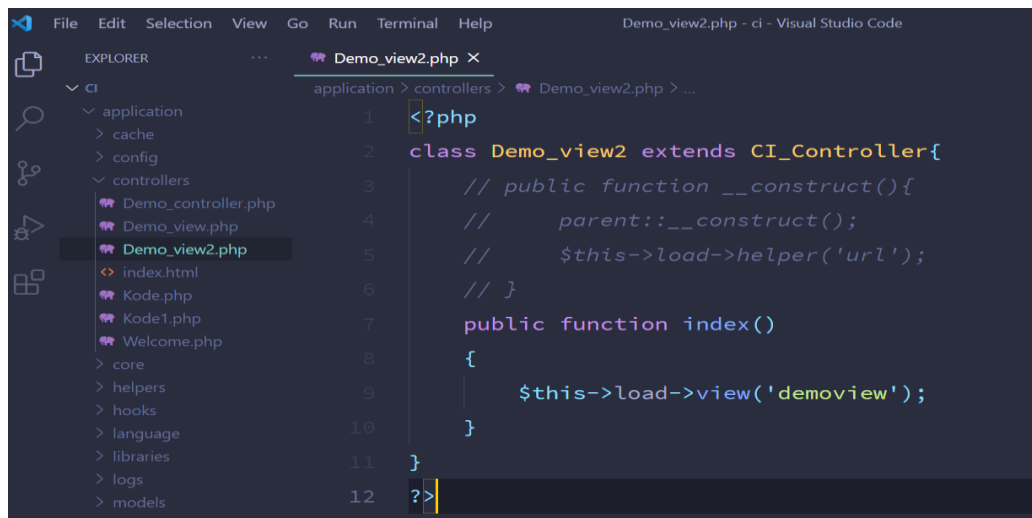
EXPLORER
ci
  application
  cache
  config
  controllers
  core
  helpers
  hooks
  language
  libraries
  logs
  models
  third_party
  views
    errors
    contact.php
    contentview.php
    cppview.php
    Demoview.php
    footerview.php
    headerview.php
    index.html
    javaview.php
    phpview.php
    profil.php
    pythonview.php
    webview2 - madrasah.nidn

application > views > Demoview.php > html
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

<style type="text/css">
    h2{
        color: #f00;
        font-size: 20px;
        border-bottom: dashed 1px #f00;
    }
    p{
        font-style: italic;
    }
</style>
<!-- <link rel="stylesheet" type="text/css" href="//base_url();?>assets/css/style.css" -->
</head>
<body>

    <h2>Demo View dengan CSS</h2>
    <p>contoh demo view dengan css</p>
```

Demo_view2.php



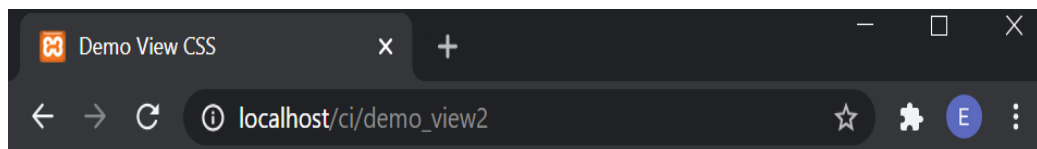
```
File Edit Selection View Go Run Terminal Help
Demo_view2.php - ci - Visual Studio Code

EXPLORER
ci
  application
  cache
  config
  controllers
  core
  helpers
  hooks
  language
  libraries
  logs
  models
  third_party

application > controllers > Demo_view2.php > ...
1
2
3
4
5
6
7
8
9
10
11
12

<?php
class Demo_view2 extends CI_Controller{
    // public function __construct(){
    //     parent::__construct();
    //     $this->load->helper('url');
    // }
    public function index()
    {
        $this->load->view('demoview');
    }
}
?>
```

Output :



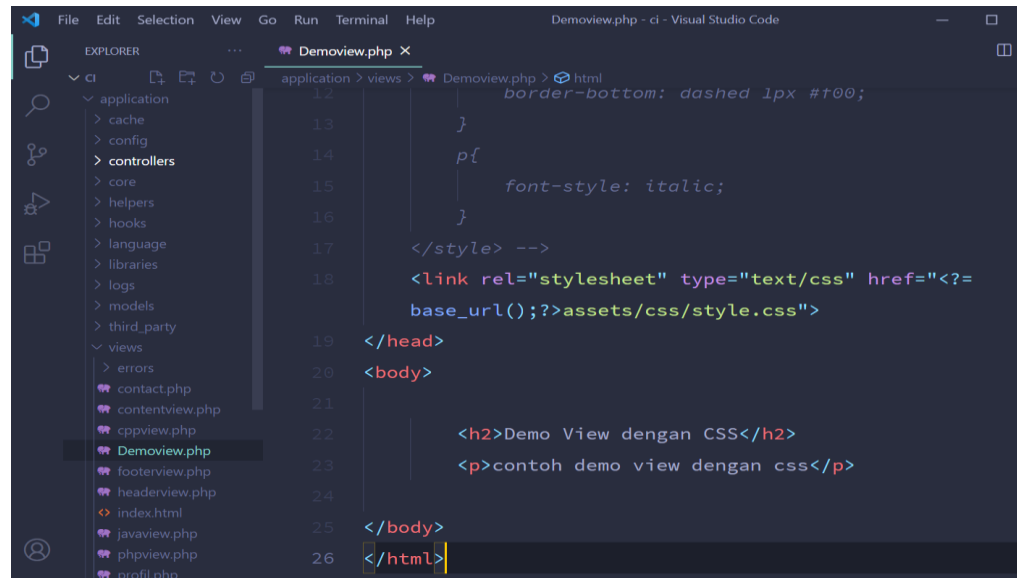
Demo View dengan CSS

contoh demo view dengan css

Kode 3 :

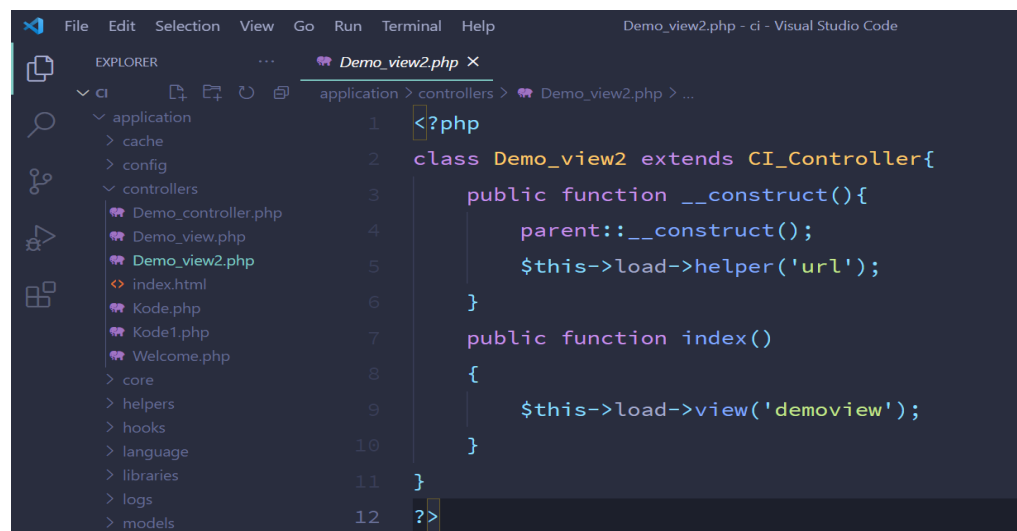
Menyisipkan CSS ke View 2

demoview.php



```
12 border-bottom: dashed 1px #f00;
13 }
14 p{
15     font-style: italic;
16 }
17 </style> -->
18 <link rel="stylesheet" type="text/css" href="{?=
19     base_url();}?>assets/css/style.css">
20 </head>
21 <body>
22     <h2>Demo View dengan CSS</h2>
23     <p>contoh demo view dengan css</p>
24 </body>
25 </html>
```

Menambahkan function `__construct` di `Demo_view2.php` untuk menangkap perintah pada `Demoview.php` memanggil `assets/css/style.css`



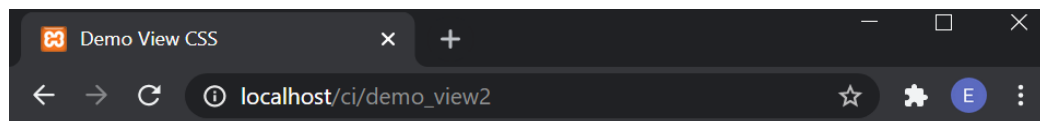
```
1 <?php
2 class Demo_view2 extends CI_Controller{
3     public function __construct(){
4         parent::__construct();
5         $this->load->helper('url');
6     }
7     public function index()
8     {
9         $this->load->view('demoview');
10    }
11 }
12 ?>
```

Sebelumnya saya telah membuat style.css yang diletakkan pada folder assets/css dengan menambahkan beberapa style dengan perintah sebagai berikut :



```
1 h2{
2     color: #f00;
3     font-size: 20px;
4     border-bottom: dashed 1px #f00;
5 }
6
7 p{
8     font-style: italic;
9 }
```

Untuk output sama dengan style yang sebelumnya dikarenakan saya memasukkan perintah yang sama pada program , namun yang membedakan hanya pada kode program ini memanggil style.css pada folder assets/css

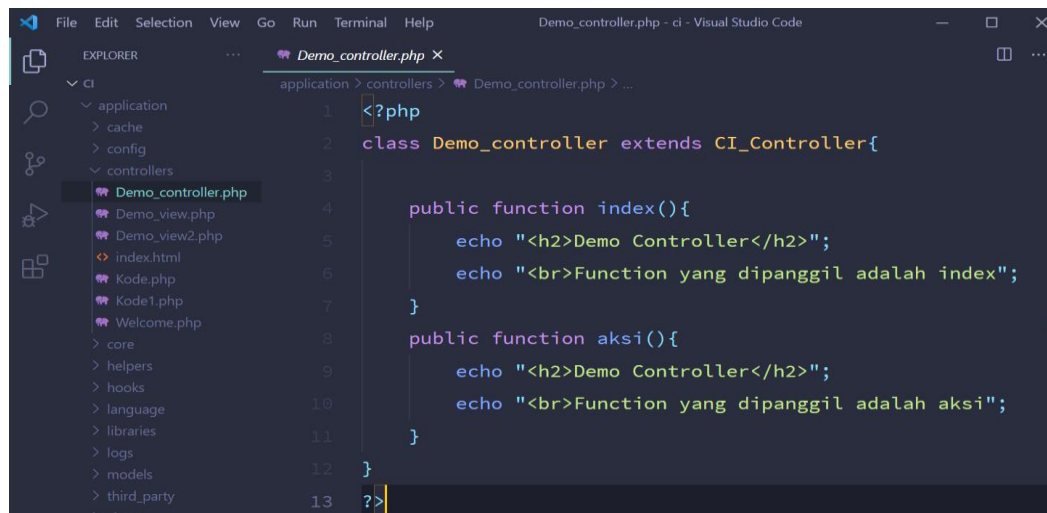


Demo View dengan CSS

contoh demo view dengan css

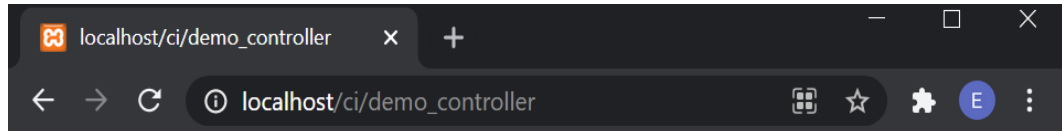
Controller

Demo_controller.php



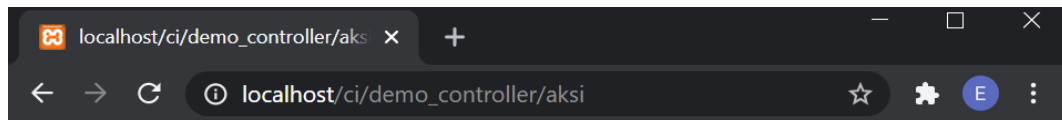
```
1 <?php
2 class Demo_controller extends CI_Controller{
3
4     public function index(){
5         echo "<h2>Demo Controller</h2>";
6         echo "<br>Function yang dipanggil adalah index";
7     }
8     public function aksi(){
9         echo "<h2>Demo Controller</h2>";
10        echo "<br>Function yang dipanggil adalah aksi";
11    }
12 }
13 ?>
```

Pada method Demo_controller terdapat 2 function yakni index dan aksi yang dapat kita run dengan mengetikkan URL `http://localhost/ci/demo_controller/index` ataupun juga bisa dengan `http://localhost/ci/demo_controller` untuk menampilkan function index dan `http://localhost/ci/demo_controller/aksi` untuk menampilkan function aksi.



Demo Controller

Function yang dipanggil adalah index

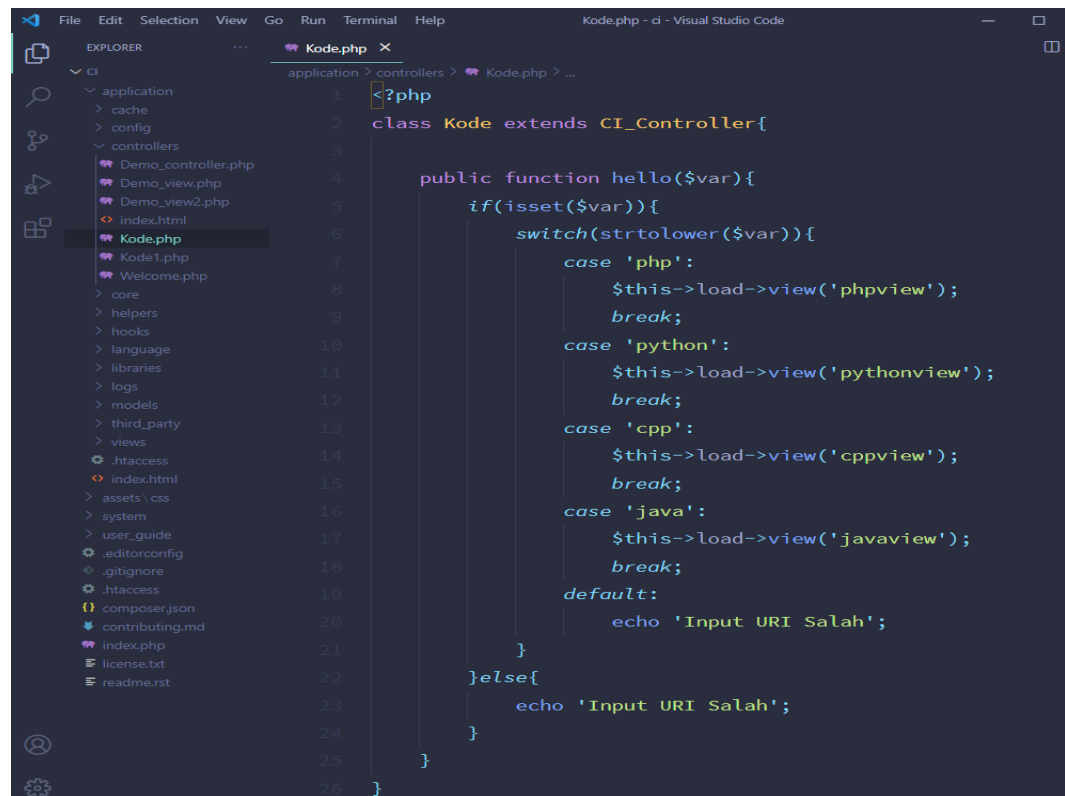


Demo Controller

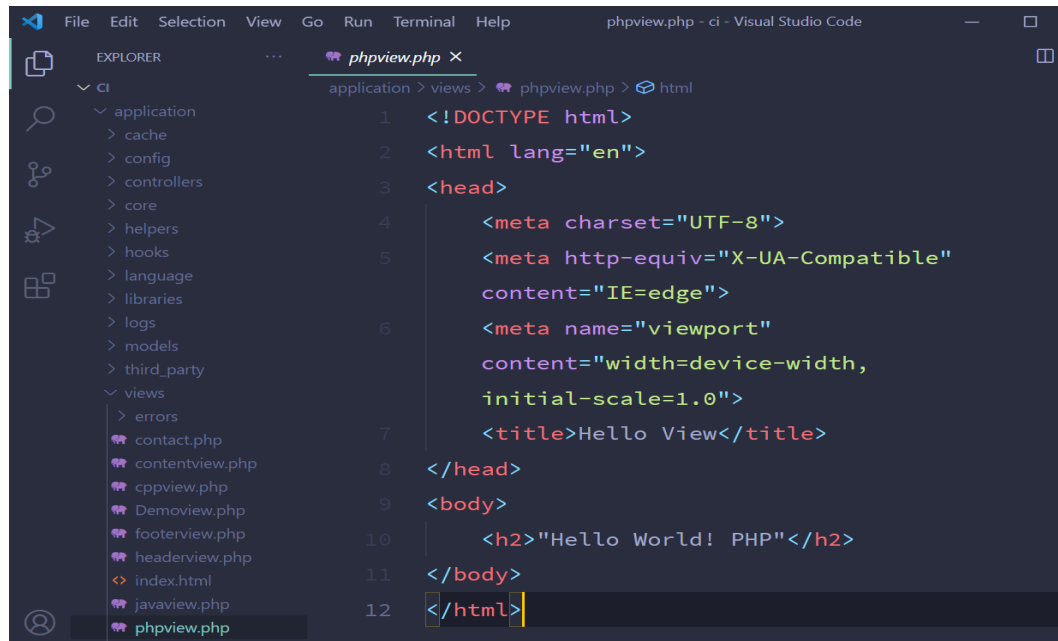
Function yang dipanggil adalah aksi

Melewatkan segment URI kedalam metode

Kode.php

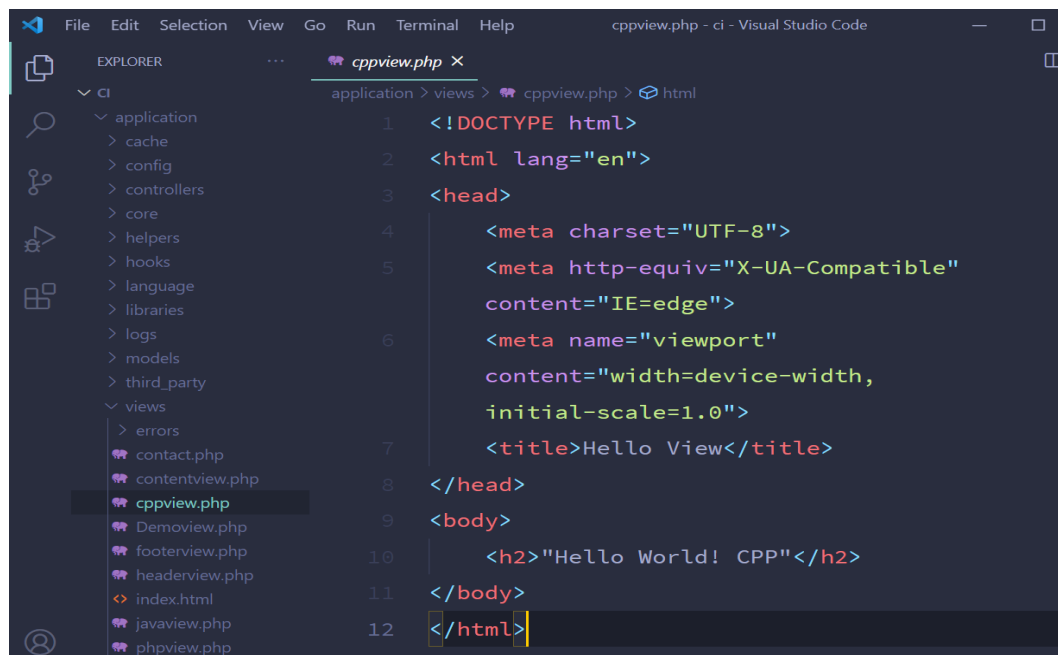


phpview.php



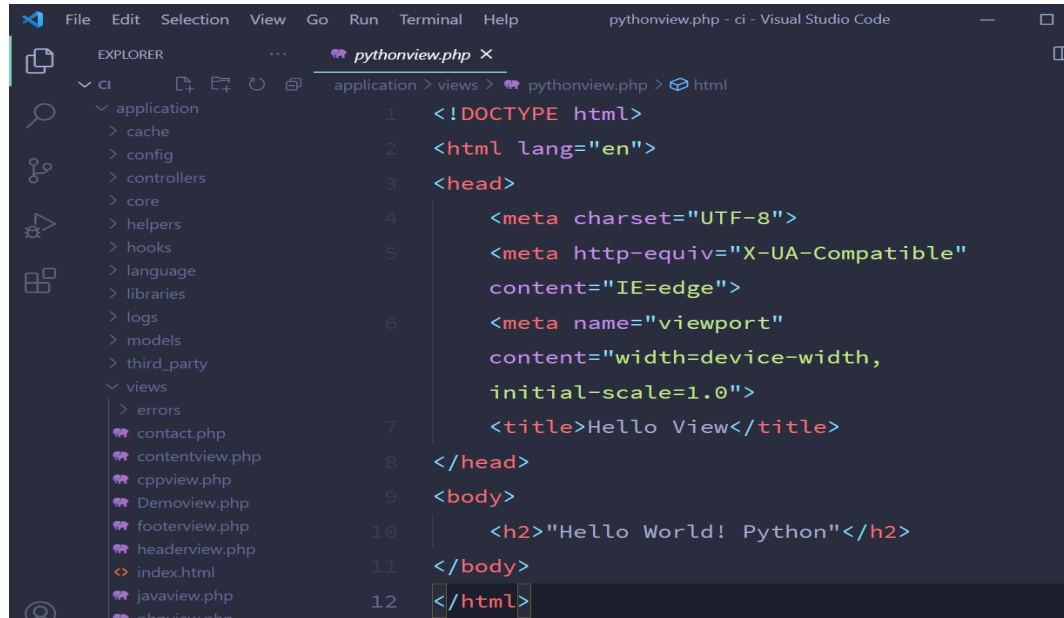
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible"
6         content="IE=edge">
7     <meta name="viewport"
8         content="width=device-width,
9         initial-scale=1.0">
10    <title>Hello View</title>
11 </head>
12 <body>
13     <h2>"Hello World! PHP"</h2>
14 </body>
15 </html>
```

cppview.php



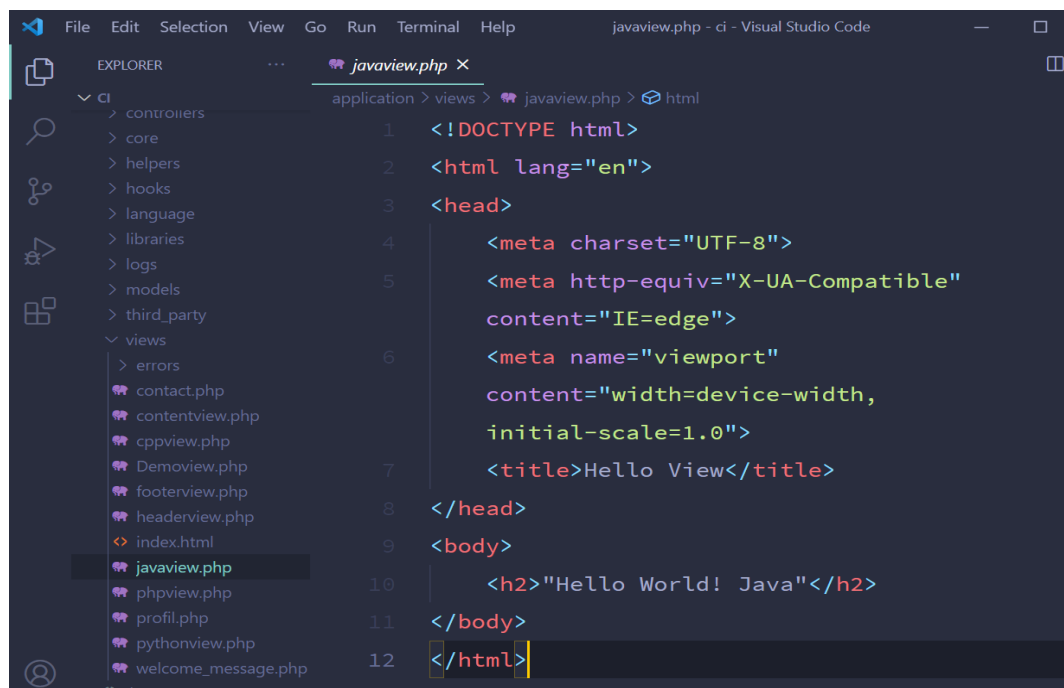
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible"
6         content="IE=edge">
7     <meta name="viewport"
8         content="width=device-width,
9         initial-scale=1.0">
10    <title>Hello View</title>
11 </head>
12 <body>
13     <h2>"Hello World! CPP"</h2>
14 </body>
15 </html>
```

pythonview.php



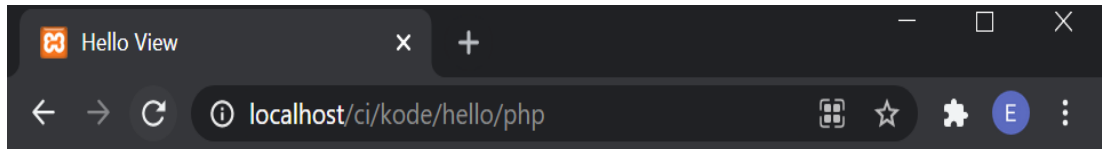
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible"
6         content="IE=edge">
7     <meta name="viewport"
8         content="width=device-width,
9         initial-scale=1.0">
10    <title>Hello View</title>
11 </head>
12 <body>
13     <h2>"Hello World! Python"</h2>
14 </body>
15 </html>
```

javaview.php



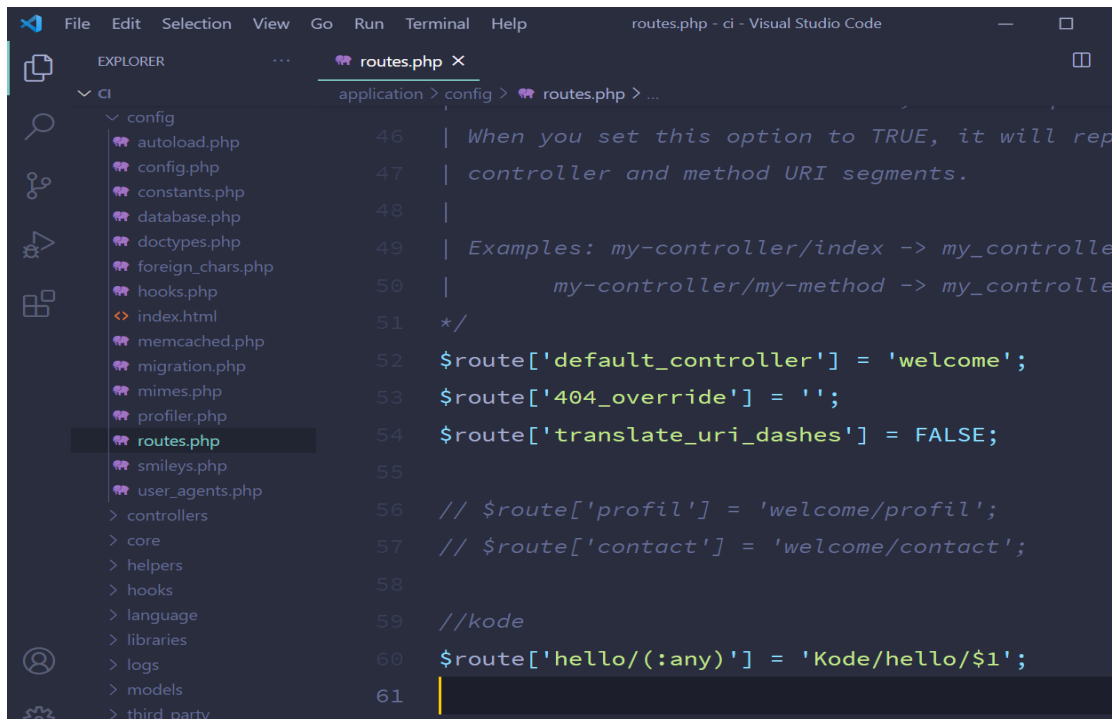
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible"
6         content="IE=edge">
7     <meta name="viewport"
8         content="width=device-width,
9         initial-scale=1.0">
10    <title>Hello View</title>
11 </head>
12 <body>
13     <h2>"Hello World! Java"</h2>
14 </body>
15 </html>
```

Segmen di URL, mengikuti pendekatan Model-View-Controller, biasanya mewakili: `http://localhost/ci/kode/hello/php` . Segmen pertama mewakili kelas pengontrol yang harus dipanggil. Segmen kedua mewakili fungsi kelas, atau metode, yang harus dipanggil. Segmen ketiga, dan setiap segmen tambahan, mewakili ID dan variabel apa pun yang akan diteruskan ke pengontrol.

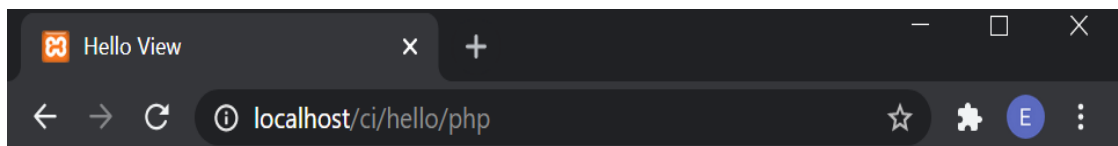


"Hello World! PHP"

Library URI dan URL Helper berisi fungsi yang memudahkan untuk bekerja dengan data URI kita. Selain itu, URL kita dapat dipetakan ulang menggunakan fitur perutean URI agar lebih fleksibel. Sehingga saya menambahkan kode program pada route sebagai berikut :



Pada routes.php saya menambahkan `$route['hello/(:any)'] = 'Kode/hello/$1';` yang arti dari program tersebut adalah ketika kita mengetikkan pada URL `http://localhost/ci/hello/`(apapun yg user ketikkan) maka akan diarahkan ke controller Kode dan methodnya hello.



"Hello World! PHP"

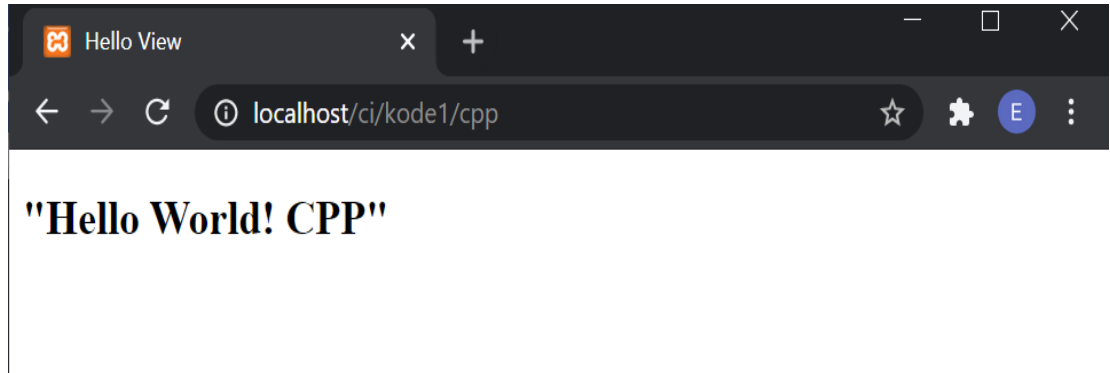
Memetakan nama metode yang akan dipanggil

Kode1

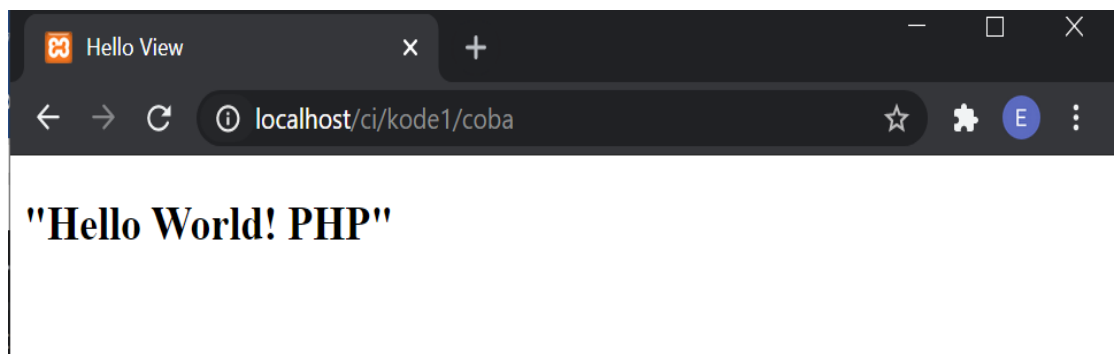
A screenshot of a code editor interface, likely Visual Studio Code, showing a file named 'Kode1.php' in the 'application > controllers' directory. The file contains PHP code for a class 'Kode1' that extends 'CI_Controller'. The class has several public methods: 'index()', 'hello_python()', 'hello_cpp()', 'hello_java()', and '_remap()'. The '_remap()' method uses a switch statement to route requests based on the URL extension (e.g., '.php', '.python', '.cpp', '.java') to the corresponding method. If no extension is found, it defaults to the 'index()' method. The Explorer sidebar on the left shows the project structure, including directories like 'application', 'cache', 'config', 'controllers', 'core', 'helpers', 'hooks', 'language', 'libraries', 'logs', 'models', 'third_party', 'views', and files like 'index.html', 'index.php', 'license.txt', and 'readme.rst'. The main editor area shows the code for 'Kode1.php' with line numbers from 1 to 36. The code is as follows:

```
1 <?php
2 class Kode1 extends CI_Controller {
3
4     public function index(){
5         $this->load->view('phpview');
6     }
7     public function hello_python(){
8         $this->load->view('pythonview');
9     }
10    public function hello_cpp(){
11        $this->load->view('cppview');
12    }
13    public function hello_java(){
14        $this->load->view('javaview');
15    }
16    public function _remap($var){
17        if(isset($var)){
18            switch(strtolower($var)){
19                case 'python':
20                    $this->hello_python();
21                    break;
22                case 'cpp':
23                    $this->hello_cpp();
24                    break;
25                case 'java':
26                    $this->hello_java();
27                    break;
28                default:
29                    $this->index();
30            }
31        }else{
32            $this->index();
33        }
34    }
35 }
36 ?>
```

Penjelasan pada program diatas adalah apabila kita menuliskan apapun setelah method Kode1 , disini saya mencontohkan <http://localhost/ci/kode1/cpp> maka program akan menangkap function _remap dan kemudian dilanjutkan ke function index untuk mengarahkan view mana yang akan ditampilkan pada output.



Namun apabila variabel yang kita tuliskan tidak ada dalam function `_remap` maka program akan menampilkan halaman index.



BAB IV

PENUTUP

4.1 Kesimpulan

URL routing (route) adalah salah satu metode yang digunakan untuk memetakan URL ke dalam sumber daya tertentu dengan memberikan nama lain dari alamat sumber daya yang dimaksud. URL routing sering digunakan untuk beberapa hal seperti menjadikan URL sumber daya yang sulit dibaca manusia dengan membuat pemetaan baru ke URL alias dari route yang lebih mudah dibaca manusia, membuat URL sumber daya menjadi lebih pendek dengan memberikan penamaan routing yang lebih pendek, dan memantau agar URL sesuai dengan format yang diinginkan dengan memanfaatkan fungsi regular expression (regex).

DAFTAR PUSTAKA

- Basuki, AP. 2010. *Membangun Web Berbasis PHP dengan Framework Codeigniter* : Yogyakarta . Lokomedia.
- Sirenden, Bernadus Herdi dan Ester Laekha Dachi. 2012. *Buat Sendiri Aplikasi Petamu Menggunakan CodeIgniter dan Google Maps API*. Yogyakarta: Andi Offset.