MODUL

PEMROGRAMAN WEB LANJUT

PENGENALAN FRAMEWORK CODEIGNITER UNTUK PENGEMBANGAN APLIKASI WEB

JURUSAN S1 INFORMATIKA
UNIVERSITAS AMIKOM YOGYAKARTA
TAHUN AJARAN 2018/2019

Daftar Isi

1. Perkenalan Codeigniter

1.1 Framework

Framework adalah sebuah struktur konseptual dasar yang digunakan untuk memecahkan sebuah permasalahan, bahkan isu-isu kompleks yang ada. Sebuah framework telah berisi sekumpulan arsitektur/konsep-konsep yang dapat mempermudah dalam pemecahan sebuah permasalahan. Perlu diingat, framework bukanlah peralatan/tools untuk memecahkan sebuah masalah, tetapi sebagai ALAT BANTU.

Keuntungan yang didapat dalam penggunaan framework adalah :

- **Menghemat Waktu Pengembangan –** Dengan struktur dan library yang telah disediakan oleh framework developer hanya fokus ke proses bisnis yang akan dikerjakan.
- **Reuse of Code –** Dengan menggunakan framework maka perkerjaan akan memiliki struktur yang baku, sehingga dapat digunakan kembali di proyek-proyek lainnya.
- **Bantuan Komunitas -** Ada komunitas-komunitas yang siap membantu jika ada permasalahan.
- **Kumpulan Best Practice –** sebuah framework merupakan kumpulan best practice atau action-action yang sudah teruji oleh expert.

1.2 Codeigniter

CodeIgniter adalah sebuah web application framework yang bersifat open source digunakan untuk membangun aplikasi php dinamis. Tujuan utama pengembangan Codeigniter adalah untuk membantu developer untuk mengerjakan aplikasi lebih cepat daripada menulis semua kode dari awal. Codeigniter menyediakan berbagai macam library yang dapat mempermudah dalam pengembangan aplikasi.

Kelebihan yang dimiliki Codeigniter diantaranya :

- Ringan dengan ukuran sekitar 2MB
- Menggunakan pattern MVC
- URL friendly

- Cepat
- Mudah dipelajari
- Mudah dimodifikasi dan beradaptasi
- Dokumentasi lengkap dan jelas

1.3 Struktur MVC

MVC adalah konsep dasar yang harus diketahui sebelum mengenal CodeIgniter. MVC adalah singkatan dari Model View Controller, yaitu sebuah pattern/teknik pemrogramanan yang memisahkan bisnis logic (alur pikir), data logic (penyimpanan data) dan presentation logic (antarmuka aplikasi) atau secara sederhana adalah memisahkan antara desain, data dan proses.

Adapun komponen-komponen MVC antara lain:

1. **Model**

   Model berhubungan dengan data dan interaksi ke database atau webservice. Model juga merepresentasikan struktur data dari aplikasi yang bisa berupa basis data maupun data lain, misalnya dalam bentuk file teks, file XML maupun webservice. Biasanya di dalam model akan berisi class dan fungsi untuk mengambil, melakukan update dan menghapus data website.

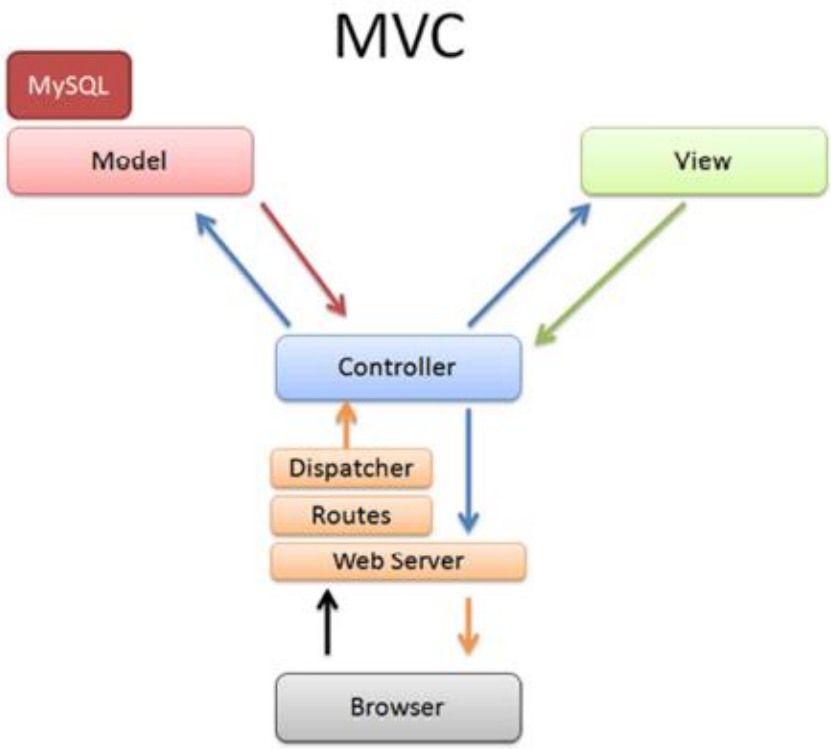
2. **View**

   View berhubungan dengan segala sesuatu yang akan ditampilkan ke end-user. View dapat dikatakan sebagai halaman website yang dibuat dengan menggunakan HTML dan bantuan CSS atau JavaScript. Hindari adanya logika atau pemrosesan data di view. View hanya dikhususkan untuk menampilkan data-data hasil dari model dan controller

3. **Controller**

   Controller bertindak sebagai penghubung data dan view. Tugas controller adalah menyediakan berbagai variabel yang akan ditampilkan di view, memanggil model untuk melakukan akses ke basis data, menyediakan

penanganan kesalahan/error, mengerjakan proses logika dari aplikasi serta melakukan validasi atau cek terhadap input.



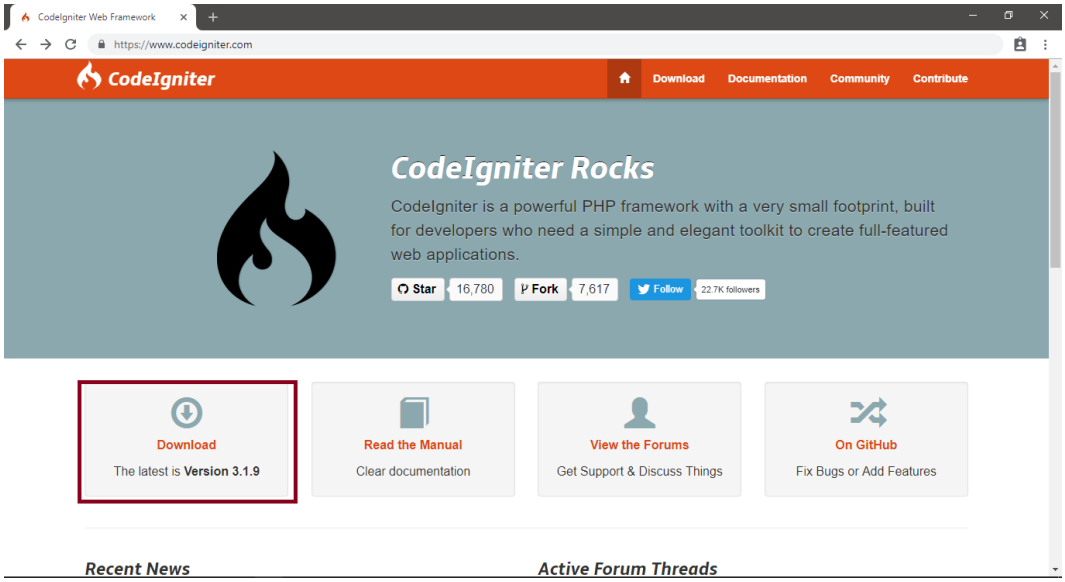
Alur kerja CodeIgniter akan tampak seperti gambar diatas. Browser berinteraksi melalui controller. Controller-lah yang akan menerima dan membalas semua request dari browser. Untuk data maka controller akan meminta ke Model dan untuk UI/template akan meminta ke View. Jadi “Otak” dari aplikasi ada di controller, “Muka” aplikasi ada di view dan “Data” ada di model.

2. Instalasi dan Konfigurasi Codeigniter

2.1 Instalasi

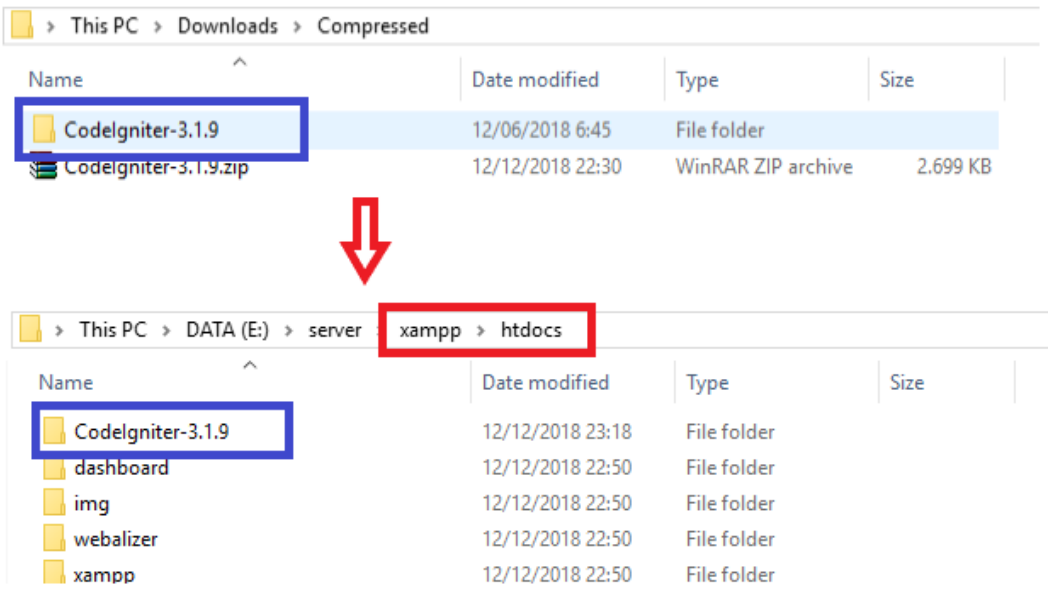
Instalasi CodeIgniter sangat mudah. Meskipun namanya instalasi tetapi karena CodeIgniter adalah aplikasi berbasis website maka sebenarnya yang perlu dilakukan adalah meng-copy folder aplikasi CodeIgniter ke dalam folder htdocs atau DocumentRoot dari web server yang telah diinstal.

1. Sebelum melakukan instalasi yang perlu dilakukan pertama kali adalah mendapatkan kode sumber dari CodeIgniter itu sendiri yang dapat didownload di https://www.codeigniter.com/

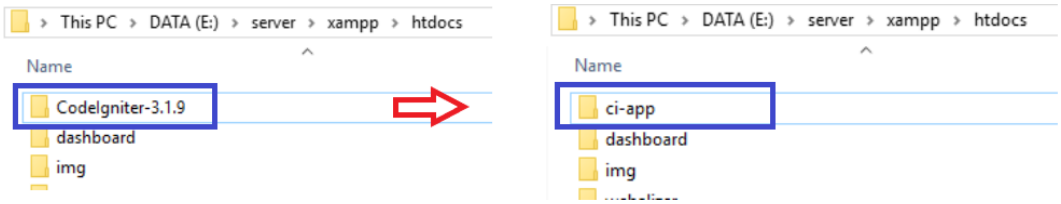


Download Codeigniter dengan cara klik link download pada area yang di tandai dengan **kotak warna merah** seperti ditunjukkan pada gambar diatas.

2. Selanjutnya ekstrak file Codeigniter dan letakkan folder hasil ekstrak tadi di **DocumentRoot web server**, yaitu folder **htdocs.**



3. Ubah nama folder Codeigniter yang **sudah di letakkan pada htdocs** menjadi **ci-app**, seperti ditunjukkan pada gambar dibawah ini.



Adapun struktur utama dari CodeIgniter terbagi menjadi dua bagian, yaitu **application** dan **sistem**/core CodeIgniter. Application adalah tempat programmer meletakkan kode program yang akan dibuat (ditandai dengan warna merah sedangkan sistem/core CodeIgniter yang berwarna ungu) . Folder sistem berisi library-library dan helper bawaan CodeIgniter.

Adapun susunan folder CodeIgniter secara default di antaranya adalah:

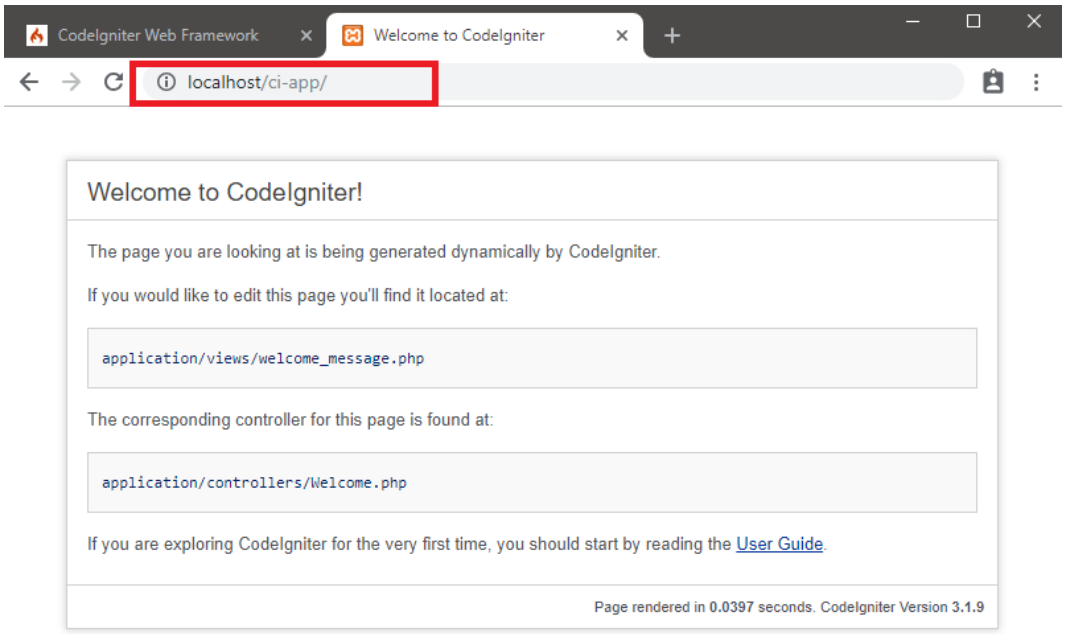
- Folder **application** : disinilah aplikasi yang akan di bangun diletakkan.
  - Folder **config** - tempat menyimpan semua file konfigurasi yang ada di dalam aplikasi, mulai dari database, router dan autoload aplikasi.
  - Folder **controllers** - tempat menyimpan semua file controller.
  - Folder **models** - tempat menyimpan semua model.
  - Folder **views** - tempat menyimpan semua file view aplikasi.

- Folder **system** : menyimpan semua file baik itu file aplikasi yang dibuat maupun core frameworknya.
  - Folder **codeigniter** - tempat menyimpan semua file internal CI.

- Folder **database** - tempat menyimpan semua driver database drivers dan class yang akan digunakan.
- Folder **helpers** - tempat menyimpan semua helper core CI.
- Folder **libaries** - tempat menyimpan semua library core CI

- Folder **user_guide** : berisi userguide/manual penggunaan CI.

4. Setelah meletakkan CodeIgniter ke dalam folder htdocs selanjutnya buka browser dan arahkan url ke aplikasi web ci-app http://localhost/ci-app/, apabila didapatkan tampilan seperti gambar di bawah ini maka Codeigniter sukses di install.



Beberapa hal yang harus diperhatikan untuk menjalankan CodeIgniter secara default adalah :

- Pastikan Apache dan PHP telah terinstal dan berjalan di komputer.
- Pastikan peletakan source code CodeIgniter di folder/directory web apache (biasanya htdocs) dan memiliki permission setidak-tidaknya read only

2.2 Konfigurasi

Walaupun CodeIgniter dapat berjalan dengan konfigurasi default, tetapi untuk sebuah aplikasi yang nyata harus tetap melakukan konfigurasi, setidaknya pada bagian **base_url** dan **router**.

File konfigurasi terletak dalam folder **application/config**. Adapun file-file yang terdapat dalam direktori tersebut dan sering digunakan antara lain:

- **Config.php.** Pada file konfigurasi config.php berisi konfigurasi secara umum mengenai CodeIgniter, seperti peletakan baseurl, suffix, frontcontroller, serta metode yang digunakan URI dan lain-lain. Adapun konfigurasi-konfigurasi yang perlu diperhatikan adalah :
  - ✓ **$config[‘base_url’]** - Konfigurasi ini berisi alamat url sebuah aplikasi. Jika menggunakan helper url maka konfigurasi ini harus di-set dengan benar. Contoh: aplikasi akan diakses dengan menggunakan domain www.domain.com/ci-app maka pada konfigurasi ini harus diisikan:
    ```
    $config['base_url']="http://www.domain.com/ci-app/";
    ```
  - ✓ **$config[‘index_php’]** - Konfigurasi ini berisi file yang menjadi frontcontroller. Konfigurasi ini berhubungan dengan base_url. Jika menggunakan .htaccess untuk mempercantik url maka isi variabel ini harus dikosongkan.
- **Autoload.php.** Konfigurasi ini bertujuan untuk menentukan sumber daya apa yang akan diload secara otomatis. Cara penggunaannya sederhana, misalnya jika ingin meload library database, pagination dan lain-lain secara otomatis maka tinggal mengubah $autoload[‘libraries’] menjadi :
  ```
  $autoload['libraries']=array('database','session')
  ```
- **Routes.php.** Konfigurasi di file ini bertujuan untuk menentukan kemana routing oleh library route akan dilakukan. Hal paling sederhana yang harus dilakukan adalah mengubah default controller (controller yang

akan dibuka ketika tidak ada uri yang diberikan oleh browser). Adapun yang perlu diubah adalah

```
$route['default_controller']="welcome";
```

2.3 Kesepakatan Koding (*Coding Standart*) CodeIgniter

Sebelum melakukan koding menggunakan codeigniter maka ada baiknya mengetahui apa saja kesepakatan-kesepakatan yang ada di codeigniter. Kesepakatan-kesepakatan tersebut akan membuat kode yang ditulis lebih mudah dipahami oleh developer lainnya . Adapun kesepakatan tersebut di antaranya :

- **PHP Closing Tak**

  Ketika akan menulis library, helper, controller ataupun model maka sebaiknya tidak menggunakan tanda penutup pada dokumen php ?>. Hal tersebut dilakukan untuk mencegah adanya spasi atau karakter yang tidak diinginkan pada code program sehingga membuat aplikasi error. Juga disarankan untuk memberikan informasi tentang akhir dokumen dan berisi path dokumen tersebut.

  Contoh salah:

  ```
  <?php echo "Here's my code!"; ?>
  ```

  Contoh Benar:

  ```
  <?php echo "Here's my code!";

  /* End of file myfile.php */

  /* Location: ./system/modules/mymodule/myfile.php */
  ```

- **Penamaan Class dan Method**

  Penamaan Class harus dimulai dengan huruf besar. Jika class menggunakan beberapa kata maka kata-kata tersebut dipisahkan menggunakan underscore dan bukan camelcase.

Contoh salah:

```
class superclass
class SuperClass
```

Contoh Benar:

```
class Super_class
```

Aturan di atas juga berlaku untuk **method**, contohnya

Contoh kurang tepat:

`function fileproperties()` //Tidak deskriptif dan memiliki underscore.

`function fileProperties()` //Tidak deskriptif dan underscore CamelCase.

`function getfileproperties()` //Kurang underscore.

`function getFileProperties()` //Menggunakan CamelCase.

`get_the_file_properties_from_the_file()` //Terlalu panjang.

Contoh Tepat:

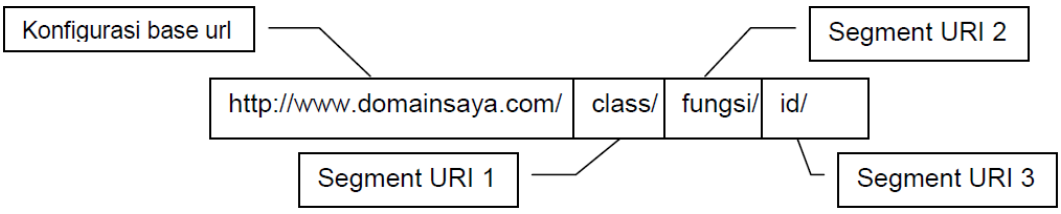
`function get_file_properties()` //Deskriptif, pakai underscore, dan huruf kecil.

3. Hello World Codeigniter

3.1 Penggunaan Controller

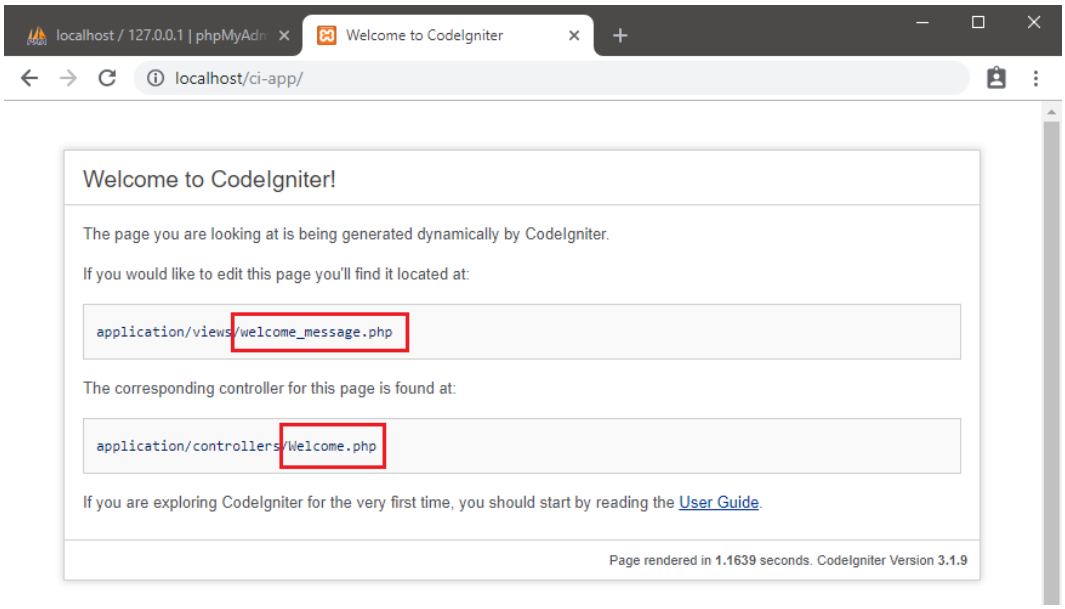
Sebuah Controller dapat dikatakan sebagai jantung dari suatu aplikasi, karena controller menentukan bagaimana permintaan HTTP yang harus ditangani. Sebuah kelas Controller adalah sebuah file yang terletak di dalam folder **application/controllers** dan memiliki nama file yang sama dengan nama kelasnya dan dikaitkan dengan URL.

Segmen-segmen pada URL pada codeigniter mencerminkan Controller yang dipanggil. Contoh: http://www.domainsaya.com/class/fungsi/id maka domain tersebut dapat dipecah menjadi bagian-bagian di antaranya:



- **Konfigurasi Base Url**, Bagian ini merupakan url yang di masukkan pada konfigurasi base_url yang merupakan url paling dasar untuk mengakses web atau aplikasi yang dibuat
- **Segmen URI pertama** yaitu class. Class tersebut merupakan nama kelas controller yang akan di panggil. Apabila segment ini kosong maka akan digantikan dengan default controller yang telah disetting di konfigurasi router.php
- **Segmen URI kedua** yaitu fungsi dari class controller yang telah di panggil tadi. Apabila segmen kedua ini kosong maka fungsi yang dipanggil adalah fungsi index dari kelas controller tersebut
- **Segmen URI ketiga** biasanya berisi parameter dari fungsi. Jika fungsi dari controller yang dipanggil mempunyai parameter maka parameternya harus dimasukkan sebagai segmen URI sesuai urutan.

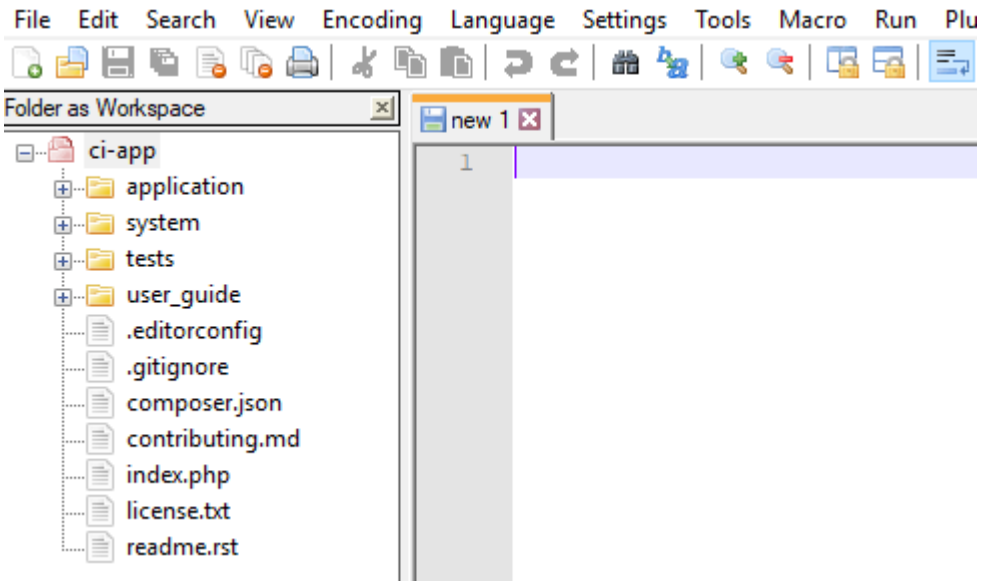
Pada saat memanggil aplikasi ci-app dengan URL http://localhost/ci-app/ yang dilakukan oleh codeigniter adalah menjalankan **controller default** dan **fungsi default** yang sudah di setting pada file **application/config/routers.php**.
Sebagai contoh pada gambar dibawah ini codeigniter menjalankan controller **Welcome** sebagai controller default yang berada di folder **application/controllers/Welcome.php** dan memanggil fungsi **index** yang berada didalam controller **Welcome**. Fungsi index me-load **view** yang berada di folder **application/views/welcome_message.php** sebagai tampilan end user.



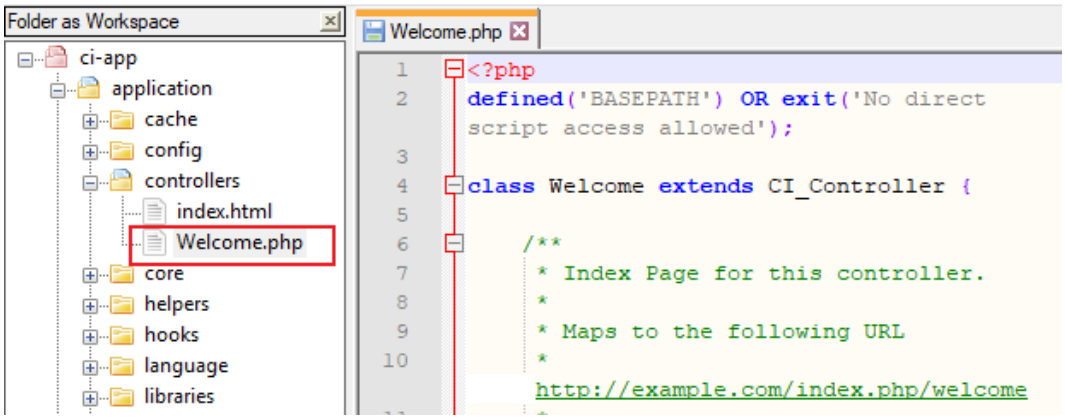
3.2 Menampilkan Hello World

Lakukan sedikit modifikasi pada controller Welcome untuk membuat program hello world. Adapun langkah-langkah yang harus dilakukan sebagai berikut :

1. Buka folder ci-app yang ada di folder htdocs menggunakan text editor



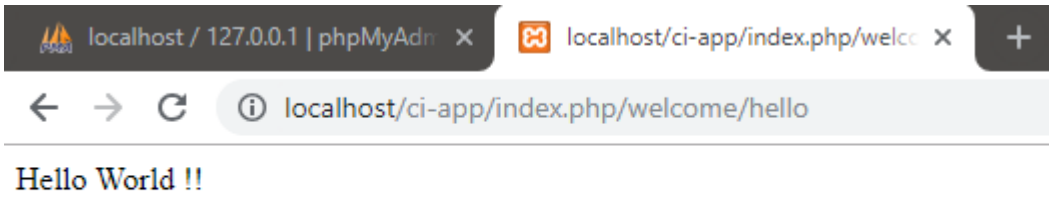
2. Buka file controller Welcome.php yang ada di folder **application/controllers/Welcome.php**



3. Buat fungsi/method baru dengan nama **hello()** di bawah method **index()** dan tampilkan pesan hello world.

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Welcome extends CI_Controller {

    /**
     * Index Page for this controller.
     *
     */
    public function index()
    {
        $this->load->view('welcome_message');
    }

    public function hello()
    {
        echo 'Hello World !!';
    }
}
```

4. Tampilkan menggunakan browser dengan mengarahkan URL ke http://localhost/ci-app/index.php/welcome/hello



Program Hello World yang berhasil ditampilkan belum sepenuhnya mengikuti aturan codeigniter karena hanya memanfaatkan controller saja untuk menampilkan data ke end-user, seharusnya tugas menampilkan data di tangani oleh view yang akan dibuat pada tahap selanjutnya.

3.3 Menggunakan Helper dan Library

CodeIgniter menyediakan dua jenis sarana yang dapat digunakan untuk membantu proses pengembangan aplikasi, antara lain:

- Library

  Library dapat dikatakan sebagai kumpulan tools yang dapat digunakan untuk membantu sebuah proses. CodeIgniter telah menyediakan banyak library yang dapat digunakan secara langsung. Library pada dasarnya adalah sebuah kelas yang diletakkan di dalam folder system/libraries atau application/libraries. Library yang terletak di dalam folder system merupakan library bawaan dari CodeIgniter yang secara default di beri awalan CI_. Untuk library buatan sendiri harus diletakkan di dalam folder application/libraries.
- Helper

  Helper adalah kumpulan fungsi yang diletakkan di dalam folder system/helpers atau applications/helpers. Biasanya helper sering digunakan dalam view untuk membantu proses-proses yang berulang, seperti generate html, url, security, dan lain-lain.

Agar dapat menggunakan library, helper dan pluggin, maka ketiganya harus di load terlebih dahulu. Ada dua cara yang dapat dilakukan untuk men-load sebuah library dan helper antara lain:

1. Menambahkan Pada Konfigurasi Autoload

   Menambahkan sebuah library di autoload berarti seluruh aplikasi akan dapat menggunakan library tersebut secara langsung. Sebaiknya library yang di load dengan cara ini adalah jenis library yang dipakai di seluruh aplikasi seperti login, template, dan lain-lain. Letak file autoload berada di folder **application/config/autoload.php.** Contoh jika akan menggunakan library database dan session pada autoload

```
$autoload['libraries'] = array('database', 'session');
```

2. Menggunakan Perintah Loader Library

   Menggunakan library loader untuk men-load library. Library loader adalah sebuah library CodeIgniter yang otomatis di load. Loader berfungsi sebagai pengatur dari sumberdaya-sumberdaya yang ada di dalam CodeIgniter seperti Model, View, Library, Helper, dan pluggin. Cara penggunaannya adalah:

```
$this->load->library('nama_library');
$this->load->helper('nama_helper');
$this->load->pluggin('nama_pluggin');
```

   Nama library, helper dan pluggin harus di isi dengan huruf kecil.

   Ketika sebuah library sudah di-load maka library tersebut menjadi property pada object Controller. Adapun cara penggunaannya adalah sebagai berikut:

```
$this->nama_library->fungsi();
```

3.4 Controller dan View

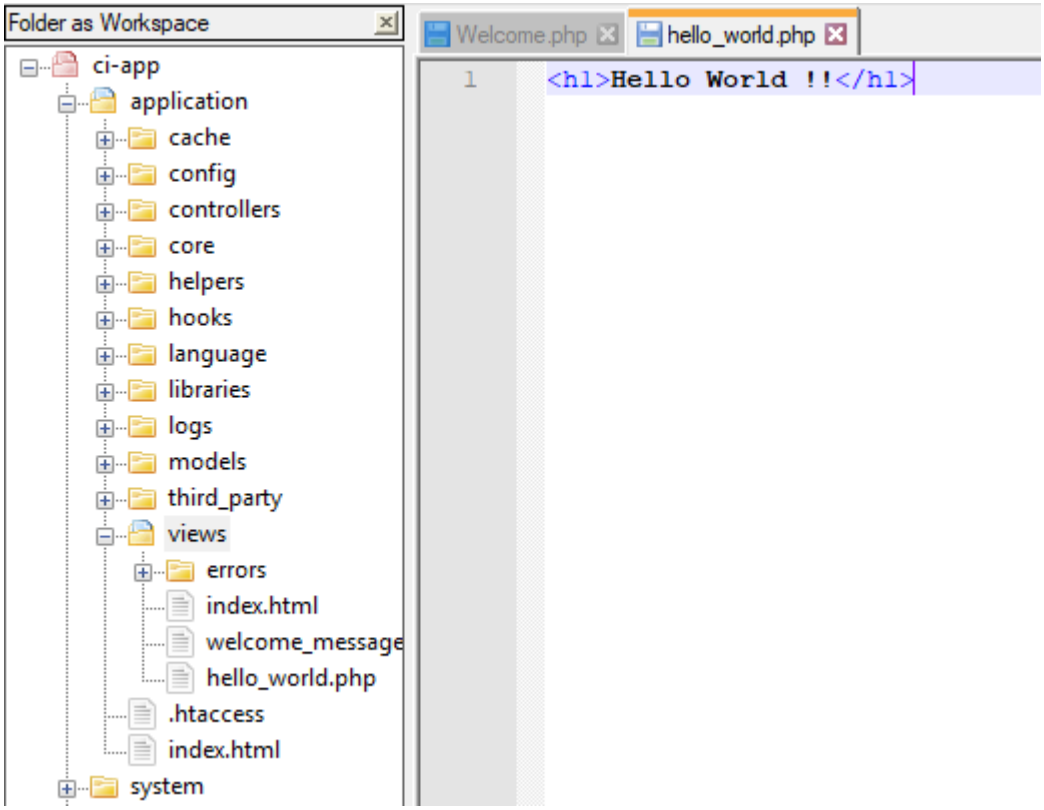
Pada contoh program Hello World yang sudah dibuat adalah cara untuk menampilkan tulisan "Hello World !!" secara langsung di controller. Namun sebenarnya hal tersebut bisa dilakukan di view.

1. Tambahkan kode program kedalam fungsi **hello()**

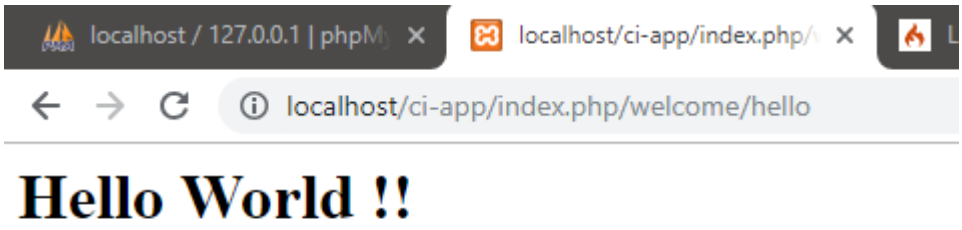
```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Welcome extends CI_Controller {

    /**
     * Index Page for this controller.
     */
    public function index()
    {
        $this->load->view('welcome_message');
    }

    public function hello()
    {
        $this->load->view('hello_world');
    }
}
```

2. Selanjutnya buatlah file **hello_world.php** di folder **application>views** (**application/views/hello_world.php**) yang berisi tulisan:

```
<h1>Hello World !!</h1>
```



3. Maka kode diatas akan memberikan hasil yang sama dengan contoh program pertama (tanpa menggunakan view), yang berbeda hanya tulisannya saja.



Sebuah View sebenarnya hanyalah sebuah halaman web atau bagian dari halaman web, seperti sebuah header, footer, sidebar, dan lain-lain. Bahkan, View bisa menjadi fleksibel karena view dapat dimasukkan ke dalam view yang lain jika dibutuhkan. Untuk memanggil file view dapat digunakan fungsi seperti berikut ini

```
$this->load->view('nama_view');
```

Nama_view adalah nama file halaman view. Dan file tersebut harus diletakkan di dalam folder **application/views**.

Fungsi view sendiri memiliki beberapa parameter:

1. **Nama file view** - Nama file yang hendak di-load yang terletak di dalam folder application/view
2. **Data Parameter** - Parameter ini digunakan untuk melewatkan data dari controller ke dalam view.

Contoh:

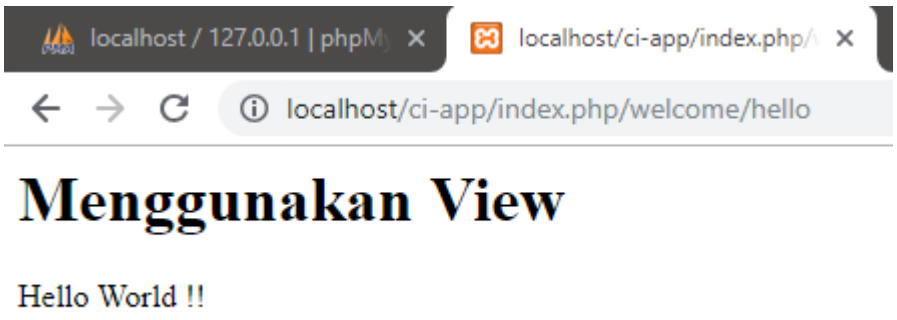
1. Pada file controller dengan nama Welcome.php tambahkan code program seperti berikut ini di dalam fungsi **hello**

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Welcome extends CI_Controller {

    /**
     * Index Page for this controller.
     */
    public function index()
    {
        $this->load->view('welcome_message');
    }

    public function hello()
    {
        $data['judul']="Menggunakan View";
        $data['isi']="Hello World !!";
        $this->load->view('hello_world', $data);
    }
}
```

2. Selanjutnya ubahlah isi file view bernama **hello_world.php** (**application/views/hello_world.php**) dengan kode program seperti berikut ini:

```
<h1><?php echo $judul; ?></h1>
<p><?php echo $isi; ?></p>
```

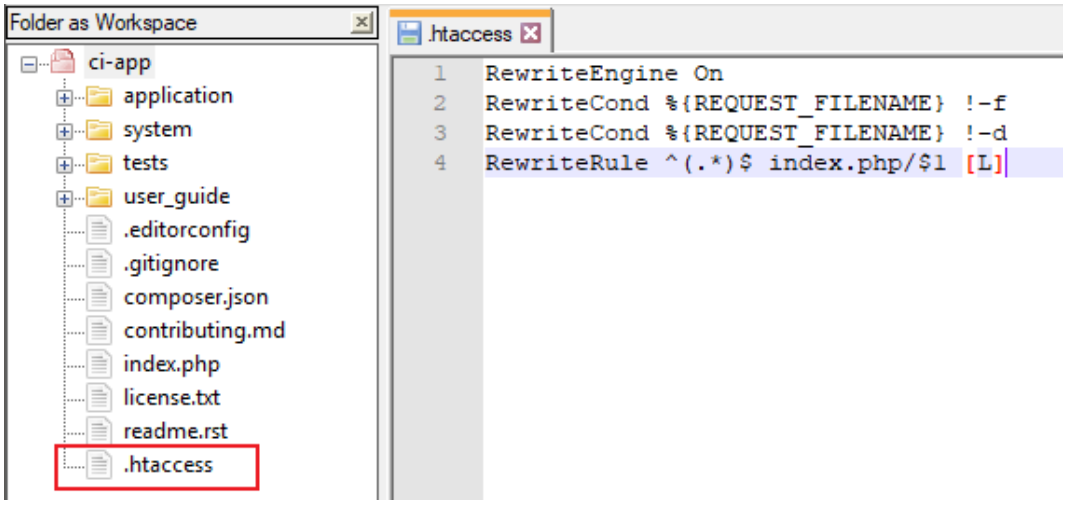
3. Jika ditampilkan menggunakan browser, kurang lebih tampilannya akan seperti berikut ini



3.5 Mempercantik URL

Jika diperhatikan URL dari program yang sudah dibuat masih terdapat nama file dan ekstensinya yaitu index.php, untuk menghilangkan nama file tersebut bisa menggunakan setingan codeigniter dan file .htaccess. adapun langkah yang dapat dilakukan adalah :

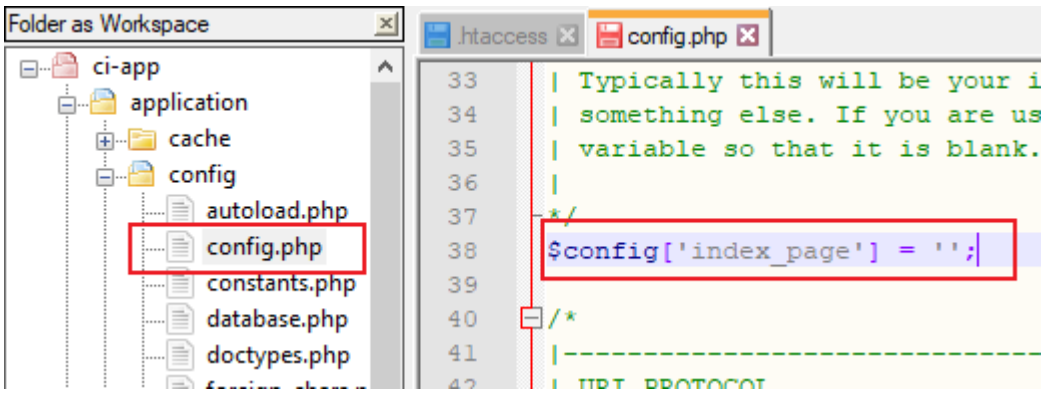
1. Membuat file **.htaccess** di folder root aplikasi ci-app



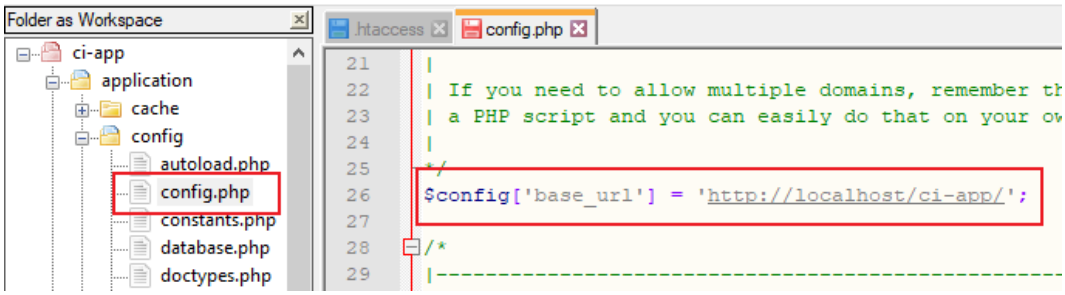
2. Adapun isi dari dari file .htaccess adalah

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L]
```

3. Mengubah konfigurasi **application/config/config.php**. Membuang "index.php" pada `$config['index_page']='index.php';` menjadi `$config['index_page'] = '';`



4. Menambahkan base URL aplikasi web ke dalam `$config['base_url'] = '';` dirubah menjadi `$config['base_url'] = 'http://localhost/ci-app/';`



Dengan menyelesaikan tahap ini sudah bisa menghilangkan index.php pada URL http://localhost/ci-app/index.php/welcome/hello dapat diakses dengan URL http://localhost/ci-app/welcome/hello tanpa index.php URL jadi lebih rapi.

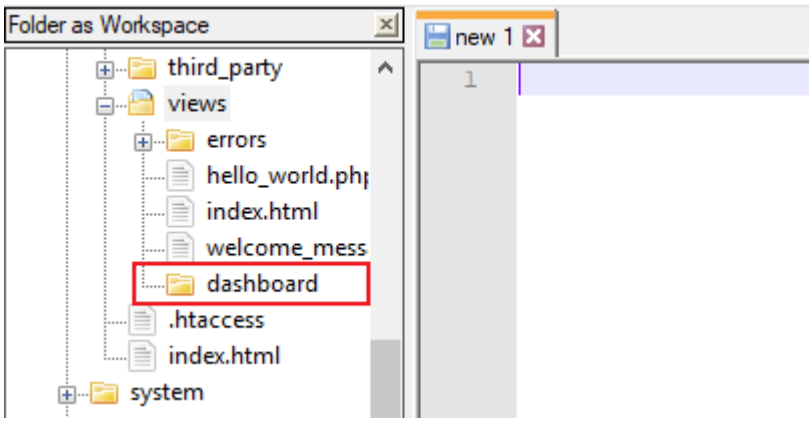
4. Database & CRUD Data

Untuk CRUD data yang akan dilakukan adalah mengubah program pengelolaan data yang sudah dibuat pada pertemuan sebelumnya kedalam framework Codeigniter.

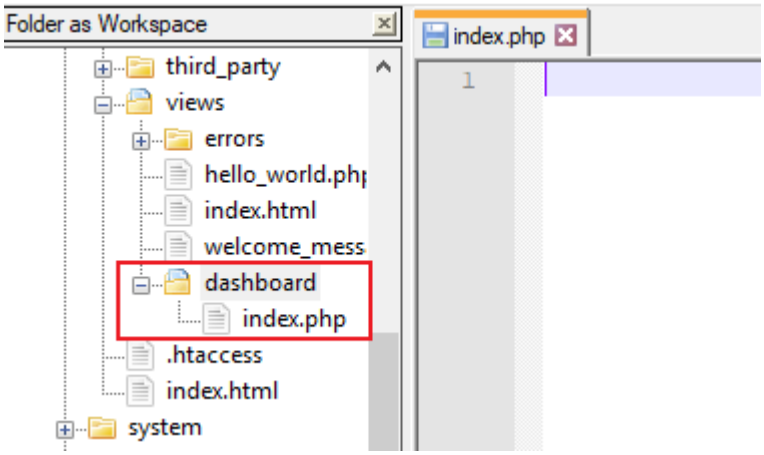
4.1 Menyiapkan Tampilan

Terlebih dahulu siapkan template untuk tampilan admin program ci-app, tampilan yang akan di gunakan yaitu tampilan modul admin pada pertemuan sebelumnya. Untuk membuat tampilan langkah-langkahnya adalah

1. Buat folder baru dengan nama **dashboard** di dalam folder application/views, (**application/views/dashboard**)



2. Buat file baru **index.php** di dalam folder dashboard (**application/views/dashboard/index.php**)

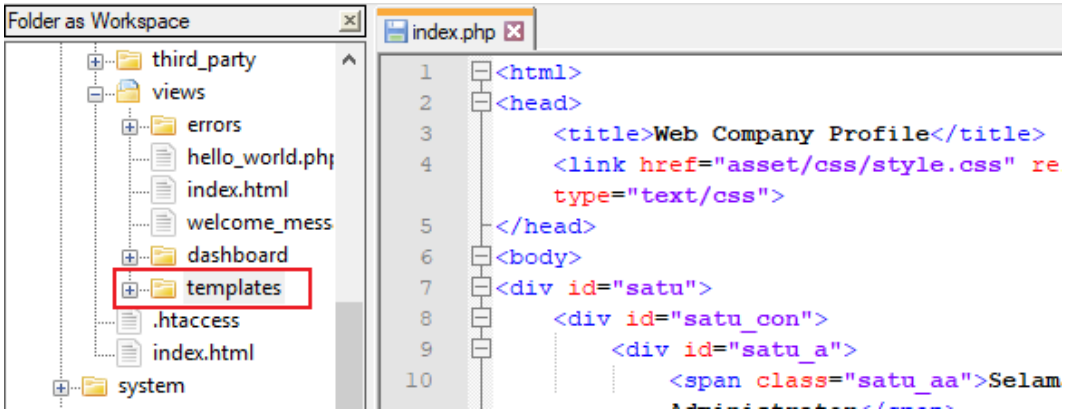


3. Salin isi file **index.html** modul admin pertemuan sebelumnya, yang berada di dalam folder **pertemuan8/admin/index.html** kedalam file **index.php** yang baru saja dibuat

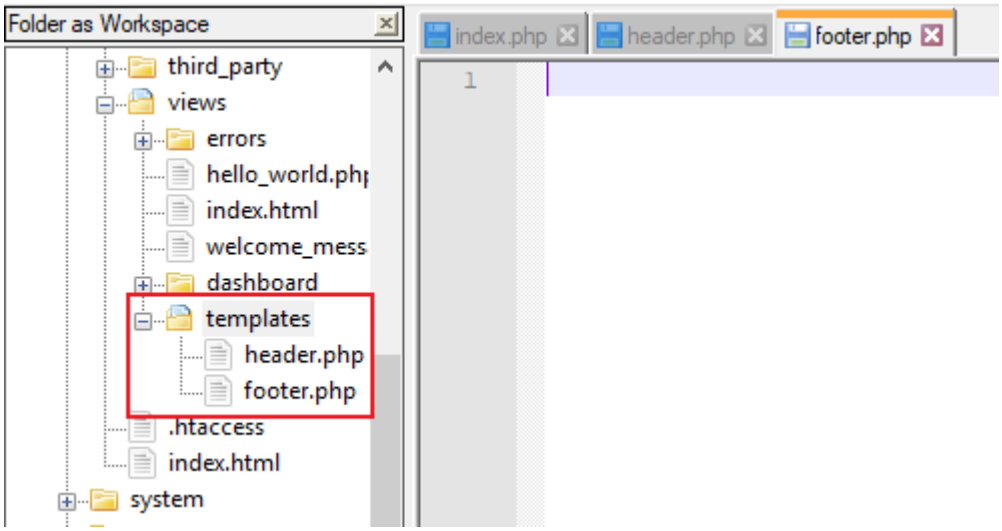
```
<html>
<head>
   <title>Web Company Profile</title>
   <link href="asset/css/style.css" rel="stylesheet" type="text/css">
</head>
<body>
<div id="satu">
   <div id="satu_con">
      <div id="satu_a">
         <span class="satu_aa">Selamat Datang Administrator</span>
         <span class="satu_ab">Creative & Innovative</span>
      </div>
      <div id="satu_b">
         <ul>
            <li><a href="#" class="pilih">Home</a></li>
            <li><a href="#">Entri Data</a>

            </li>
         </ul>
      </div>
```

```
    </div>
</div>
<div id="dua">
    <div id="dua_con">
        <div id="dua_a">
            <div id="dua_aa">
                <span id="dua_aab">Data Biodata</span>
                <!--<span id="dua_aaa">
                    <input type="button" value="Tambah Data" class="tambah">
                </span>
                -->
            </div>
            <div id="dua_ac">

            </div>
            <div id="dua_ab">
                <!--
                Script
                -->
            </div>
        </div>
    </div>
</div>
</body>
</html>
```

4. Membuat struktur template yaitu memisahkan bagian-bagian isi file index.php ke dalam file-file terpisah, dengan cara buat folder baru dengan nama **templates** di dalam folder application/views, (**application/views/templates**)



5. Buat file dengan nama **header.php** dan **footer.php** di dalam folder templates, (**application/views/templates/header.php** & **application/views/templates/footer.php**)



6. Pindahkan kode program yang ada pada file **dashboard/index.php** dimulai dari tag pembuka `<html>` hingga tag pembuka `<div id="dua_ab">` kedalam file **templates/header.php**

```
<html>

<head>
   <title>Web Company Profile</title>
   <link href="asset/css/style.css" rel="stylesheet" type="text/css">
</head>
<body>
<div id="satu">
```

```
    <div id="satu_con">
        <div id="satu_a">
            <span class="satu_aa">Selamat Datang Administrator</span>
            <span class="satu_ab">Creative & Innovative</span>
        </div>
        <div id="satu_b">
            <ul>
                <li><a href="#" class="pilih">Home</a></li>
                <li><a href="#">Entri Data</a>

                </li>
            </ul>
        </div>
    </div>
</div>
<div id="dua">
    <div id="dua_con">
        <div id="dua_a">
            <div id="dua_aa">
                <span id="dua_aab">Data Biodata</span>
                <!--<span id="dua_aaa">
                    <input type="button" value="Tambah Data" class="tambah">
                </span>
                -->
            </div>
            <div id="dua_ac">

            </div>
            <div id="dua_ab">
```

7. Pindahkan kode sisanya pada file **dashboard/index.php** dimulai dari tag penutup `</div>` hingga tag penutup html `</html>` kedalam file **templates/footer.php**

```
            </div>
        </div>
    </div>
```

```
</div>
</body>
</html>
```

8. Pada file **dashboard/index.php** tampilkan pesan selamat datang administrator

```
<p>Selamat datang Administrator</p>
```

9. Berikan keterangan nama halaman dengan cara mengubah kode program pada file **templates/header.php**

Source code

```
<span id="dua_aab">Data Biodata</span>
```

ubah menjadi

```
<span id="dua_aab"><?php echo $title; ?></span>
```

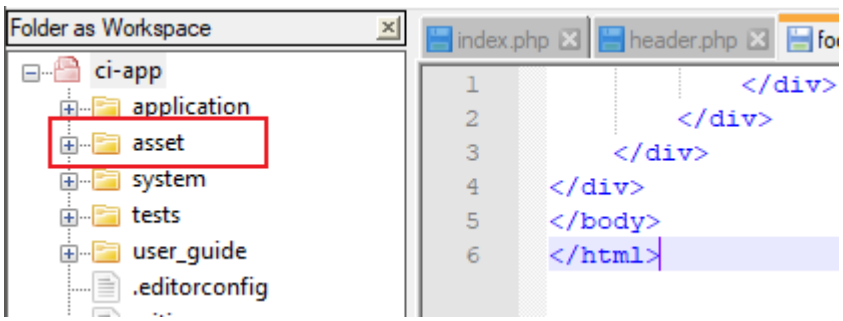
Source code

```
<title>Web Company Profile</title>
```

ubah menjadi

```
<title>Web Company Profile - <?php echo $title;
?></title>
```

10. Salin **folder asset** yang ada di modul admin ke dalam folder root aplikasi ci-app, (**ci-app/asset**)



11. Ubah link css pada file **templates/header.php** arahkan ke file css yang ada di folder asset menggunakan **base_url()**

```
<html>
<head>
   <title>Web Company Profile - <?php echo $title; ?></title>
   <link href="<?php echo base_url();?>asset/css/style.css" rel="stylesheet" type="text/css">
</head>
```

12. Fungsi base_url() merupakan helper url yang digunakan untuk menunjukkan url utama atau domain dari aplikasi yang dibuat, agar helper dapat digunakan lakukan load terlebih dahulu pada file **application/config/autoload.php**.

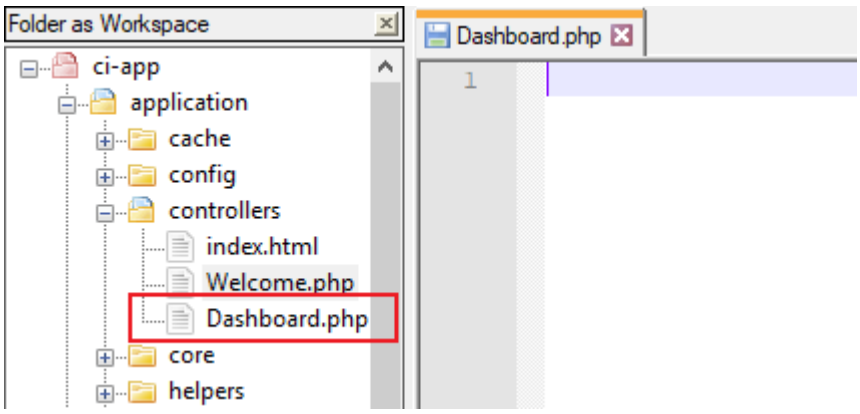
Ubah baris kode : `$autoload['helper'] = array();`

Menjadi : `$autoload['helper'] = array('url');`

4.2 Menampilkan Template Admin

Atur controller untuk melihat hasil dari template admin yang telah dibuat, adapun caranya adalah

1. Buat file controller baru dengan nama **Dashboard.php** di dalam folder application/controllers (**application/controllers/Dashboard.php**)



2. Buat class dan method pada file Dashboard.php, serta load view template yang sudah dibuat. Perlu di ingat nama file controller harus sama dengan nama classnya

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Dashboard extends CI_Controller {

    public function index()
    {
        $data['title']  = 'Halaman Dashboard';

        $this->load->view('templates/header', $data);
        $this->load->view('dashboard/index');
        $this->load->view('templates/footer');
    }
}
```

3. Ubah default controller pada file **application/config/routes.php** arahkan ke controller dashboard

```
$route['default_controller'] = 'dashboard';
```

4. Arahkan URL browser ke http://localhost/ci-app, jika yang tampil adalah dashboard admin berarti templating berhasil dilakukan.

4.3 Koneksi Database

CodeIgniter mendukung banyak jenis database misalnya MySql, PostGre SQL, Oracle. CodeIgniter memiliki sebuah file konfigurasi yang memungkinkan menyimpan konfigurasi untuk melakukan koneksi ke database (username, password, nama database, dan lain-lain).

Untuk memulai melakukan koneksi database terlebih dahulu buat database beserta tabel-tabel yang dibutuhkan, pada contoh ini akan menggunakan database yang telah dibuat pada pertemuan sebelumnya.

Pertama lakukan konfigurasi pada file database yang terletak di **application/config/database.php.** Sesuaikan hostname, username, password, dan nama database dengan setingan database masing-masing.

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

$active_group = 'default';
$query_builder = TRUE;

$db['default'] = array(
    'dsn'   => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'sekolah1',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => FALSE,
    'db_debug' => (ENVIRONMENT !== 'production'),
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'failover' => array(),
    'save_queries' => TRUE
);
```

Karena database termasuk library di dalam CodeIgniter, untuk connect ke database ada beberapa cara yang disediakan oleh CodeIgniter diantaranya :

1. **Menambahkan Database Library Sebagai Autoload Library**

   Untuk connect ke database bisa dengan menambahkan database sebagai autoload library di file **application/config/autoload.php**. Cara ini sangat sederhana, cukup menambahkan kata "database" ke dalam autoload library, sehingga menjadi :

   ```
   $autoload['libraries'] = array('database');
   ```

2. **Mengaktifkan Manual Dari Library Database**

   Jika hanya ada beberapa halaman website yang memerlukan konektivitas database, maka untuk optimalisasi lakukan koneksi ke database secara manual, cukup dengan menambahkan baris kode di bawah ini pada tiap fungsi yang membutuhkan koneksi kedatabase atau dalam konstruktor kelas untuk membuat database tersedia secara global di kelas. Cara load library koneksi database menggunakan perintah seperti di bawah ini

   ```
   $this->load->database();
   ```

   Karena yang menangani data adalah model, sebaiknya load koneksi database di lakukan didalam kelas Model.

4.4 Model

Model pada CodeIgniter adalah sebuah kelas php yang berfungsi untuk menangani data. Data bukan hanya dari database tetapi juga bisa dari File Text, Web Service atau layanan-layanan data lainnya. Semua Model harus diletakkan di dalam folder **application/models**. Biasanya pembuatan model mengikuti desain database atau dapat diartikan satu tabel dapat diwakili oleh satu model. Model tersebutlah yang bertanggung jawab pada semua operasi pada satu tabel tersebut.

Agar dapat menggunakan model maka harus me-load model tersebut. Adapun perintah yang dapat digunakan untuk meload sebuah model adalah

```
$this->load->model('Model_name');
```

Ketika sudah berhasil me-load sebuah Model di dalam kelas Controller maka model tersebut akan menjadi sebuah property dari class controller yang me-loadnya. Melalui property itulah digunakan untuk mengakses semua fungsi yang ada di dalam Model. Contohnya untuk mengakses fungsi **get_data()** didalam class Model dengan nama **Model_name** caranya adalah

```
$this->Model_name->get_data();
```

4.5 Query Builder

CodeIgniter menyediakan cara yang mudah untuk melakukan query dengan database yang disebut dengan query builder. Query builder pada codeigniter merupakan sebuah fungsi yang sudah disediakan untuk melakukan create, read, update, dan delete pada database, sehingga memungkinkan developer untuk melakukan crud data tanpa harus menuliskan perintah querynya. Perhatikan contoh dibawah ini.

Mengambil data tanpa menggunakan query builder

```
$query = $this->db->query('SELECT * FROM biodata');

return $query->result_array();
```

Menggunakan query builder

```
$query = $this->db->get('biodata');

return $query->result_array();
```

Perintah query diatas sama-sama akan mengambil semua data pada tabel biodata dan mengembalikan hasilnya dalam bentuk array. Lebih dalam tentang penggunaan query builder bisa di lihat pada dokumentasi atau user guide CodeIgniter.

4.6 Menampilkan Data

1. Buat Controller baru dengan nama **Biodata.php** di dalam folder **application/controllers/Biodata.php**, isi dengan source code seperti dibawah ini

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Biodata extends CI_Controller {

    public function __construct()
    {
        parent::__construct();
        $this->load->model('Biodata_model'); //load Model
    }

    public function index()
    {
        $data['title']  = 'Halaman List Biodata';
        $data['menu']   = 'biodata';
        $data['biodatas'] = $this->Biodata_model->get_biodata_list();

        $this->load->view('templates/header', $data);
        $this->load->view('biodata/index', $data);
        $this->load->view('templates/footer');
    }
}
```

   Jangan lupa untuk melakukan load Model seperti yang dilakukan fungsi `__construct()` pada source code controller Biodata diatas

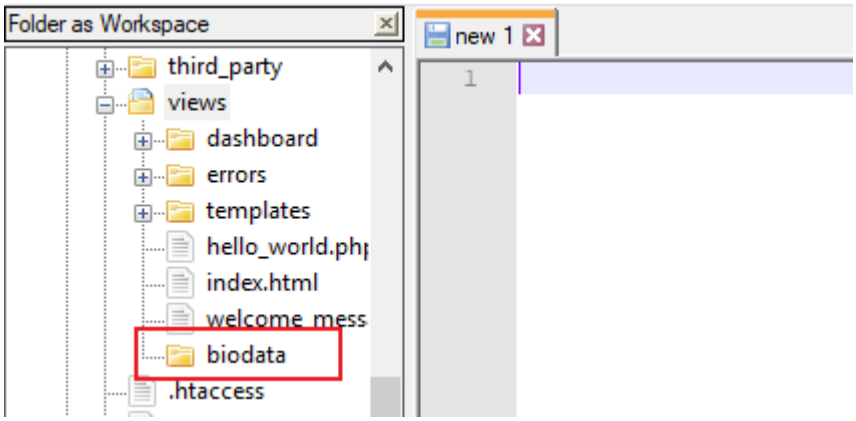
2. Koneksi kedatabase bisa menggunakan konfigurasi autoload yang ada di folder **application/config/autoload.php**, ubah menjadi `$autoload['libraries'] = array('database');` atau meload database pada saat membuat Model, seperti yang dilakukan pada contoh di tahap selanjutnya

3. Membuat Model **Biodata_model.php** untuk tabel biodata di dalam folder **application/models/Biodata_model.php** serta buat fungsi untuk mengambil isi tabel biodata seperti pada contoh dibawah ini

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Biodata_model extends CI_Model
{
    function __construct()
    {
        parent::__construct();
        $this->load->database(); //load koneksi Database
    }

    function get_biodata_list()
    {
        $query = $this->db->get('biodata');
        return $query->result_array();
    }
}
```

Jangan lupa melakukan load koneksi database seperti yang ditunjukkan source code diatas pada fungsi `__construct()`

4. Buat folder **biodata** di dalam folder **application/views** untuk menyimpan semua halaman view dari tabel biodata



5. Buat file **index.php** didalam folder biodata (**application/views/biodata/index.php**), kemudian tampilkan isi biodata di dalam file index.php menggunakan table, seperti contoh dibawah ini

```
<p>[ <a href="">Tambah Data</a> ] </p>
<table width="650px" border="1">
  <tr style="background:#ccc">
    <th width="10%">ID</th>
    <th width="22%">Nama</th>
    <th width="50%">Alamat</th>
    <th>Aksi</th>
  </tr>

  <?php foreach ($biodatas as $biodata) : ?>
    <tr>
      <td align="center"><?= $biodata['id']; ?></td>
      <td><?= $biodata['nama']; ?></td>
      <td><?= $biodata['alamat']; ?></td>
      <td>
        <a href="">
          Detail
        </a>
        <a href="">
          Ubah
        </a>
        <a href=""
          onClick="return cekHapus();">
          Hapus
        </a>
      </td>
    </tr>
  <?php endforeach; ?>
</table>

<script language="JavaScript">
  function cekHapus(){
    if(confirm("Yakin? Ciyus mau hapus data ini?")){
      return true;
```

```
    } else {
        return false;
    }
  }
</script>
```

6. Arahkan link menu biodata ke controller biodata, buka file **header.php** yang ada di folder **application\views\templates\header.php** lakukan perubahan source code pada bagian link menu seperti dibawah ini

```
<div id="satu_b">
  <ul>
    <li><a href="<?= base_url(); ?>" class="<?= $menu=='home'?'pilih':'' ?>">Home</a></li>
    <li><a href="<?= base_url(); ?>biodata" class="<?= $menu=='biodata'?'pilih':'' ?>">Biodata</a>

    </li>
  </ul>
</div>
```

7. Fungsi **base_url()** akan menampilkan url utama dari aplikasi web yang sebelumnya sudah di konfigurasi di file **config.php** di dalam folder **application/config/config.php**. fungsi base_url() termasuk ke dalam helper codeigniter sehingga perlu melakukan load terlebih dulu agar bisa digunakan. Karena banyak link dan halaman yang akan menggunakan fungsi base_url() maka lakukan load helper url di file **autoload.php** yang ada di dalam folder **application/config/autoload.php**, cari helper dan tambahkan url seperti contoh source code dibawah ini

```
/*
| -------------------------------------
|  Auto-load Helper Files
| -------------------------------------
| Prototype:
|
|   $autoload['helper'] = array('url', 'file');
```

```
*/
$autoload['helper'] = array('url');
```

8. Simpan perubahan dan reload browser, pastikan ketika menu Biodata di klik maka muncul tabel berisi data dari tabel biodata

4.7 Menambah Data

Menambahkan data memerlukan bantuan form validation untuk memvalidasi inputan user, form validation termasuk ke dalam library codeigniter sehingga perlu me-loadnya terlebih dahulu sebelum menggunakannya.

1. Load library form_validation di dalam fungsi **__construct** class controller Biodata di folder **application/controllers/Biodata.php** seperti pada source code dibawah ini

```
public function __construct()
{
    parent::__construct();
    $this->load->model('Biodata_model'); //load Model
    $this->load->library('form_validation');              //load
form_validation
}
```

2. Membuat fungsi untuk menampilkan form tambah data pada class Controller Biodata, tambahkan source code di bawah ini di bawah method `index()`.

```
public function tambah()
{
  $data['title']  = 'Halaman Tambah Data';
  $data['menu']   = 'biodata';

  $this->form_validation->set_rules('txtNama', 'Nama', 'required');
  $this->form_validation->set_rules('txtAlamat', 'Alamat', 'required');

  if ($this->form_validation->run() == FALSE)
  {
```

```
        $this->load->view('templates/header', $data);
        $this->load->view('biodata/tambah');
        $this->load->view('templates/footer');
    }
    else
    {
        $this->Biodata_model->create_new_biodata();
        $this->session->set_flashdata('flash', 'Ditambahkan');
        redirect('biodata');
    }
}
```

3. Membuat file baru **tambah.php** untuk menampilkan form tambah data pada views (**application/views/biodata/tambah.php**), tambahkan source code dibawah ini untuk membuat form

```
<?php if( validation_errors() ) { ?>
    <div id="eror"><?= validation_errors(); ?></div>
<?php } ?>

<form action="" method="post">
    Nama :  <input type="text" name="txtNama" />
    <br/><br/>
    Alamat : <textarea name="txtAlamat"></textarea>
    <br /><br/>
    <input type="submit" value="Submit" />
</form>

<p>[ <a href="<?= base_url(); ?>biodata">Tampil Data</a> ] </p>
```

4. Aktifkan library session untuk membuat pesan flash di file **autoload.php** folder **application/config/autoload.php**, tambahkan library menjadi

```
$autoload['libraries'] = array('session');
```

5. Membuat method **create_new_biodata()** untuk insert data pada class Model Biodata yang berada di folder **application/models/Biodata_model.php**

```
function create_new_biodata()
{
   $data = [
        "nama" => $this->input->post('txtNama', true),
        "alamat" => $this->input->post('txtAlamat', true)
   ];

   $this->db->insert('biodata', $data);
}
```

6. Mengarahkan link tambah data di file **index.php** folder **application/views/biodata/index.php** ke halaman form tambah data

```
<p>[ <a href="<?= base_url(); ?>biodata/tambah">Tambah Data</a> ] </p>
```

7. Menampilkan pesan flash pada file **index.php** folder **application/views/biodata/index.php**

```
<?php if( $this->session->flashdata('flash') ){ ?>
   <div id="succes">Biodata <strong>Berhasil</strong> <?= $this->session->flashdata('flash'); ?></div>
<?php } ?>

<p>[ <a href="<?= base_url(); ?>biodata/tambah">Tambah Data</a> ] </p>
```

8. Merubah bahasa pesan error validasi dengan cara copy file **form_validation_lang.php** yang berada di folder **system\language\english\form_validation_lang.php** pastekan ke **application\language\english\form_validation_lang.php**, kemudian rubah bahasa pesan validasi required menggunakan bahasa indonesia

```
$lang['form_validation_required']       = 'Field {field} tidak boleh kosong.';
```

9. Tambahkan data baru menggunakan form input yang telah dibuat.

4.8 Menghapus Data

1. Tambahkan method **hapus()** di dalam class Controller **Biodata.php** yang menerima parameter berupa id data yang akan dihapus, seperti source code dibawah ini

```
public function hapus($id)
{
   $this->Biodata_model->delete_biodata($id);
   $this->session->set_flashdata('flash', 'Dihapus');
   redirect('biodata');

}
```

2. Buat method baru delete_biodata di dalam class Model **Biodata_model.php** untuk menghapus data dengan id tertentu pada tabel biodata

```
function delete_biodata($id)
{
   $this->db->where('id', $id);
   $this->db->delete('biodata');
}
```

3. Kirimkan id biodata yang akan dihapus melalui tombol **hapus** di kolom aksi tabel biodata file **index.php** yang ada di folder **application/views/biodata/index.php**

```
<a href="<?= base_url(); ?>biodata/hapus/<?= $biodata['id'] ?>"
   onClick="return cekHapus();">
```

```
    Hapus
</a>
```

4. Lakukan penghapusan salah satu data jika muncul pesan data berhasil dihapus maka metode hapus data berhasil

4.9 Melihat Detail Data

1. Membuat fungsi **detail()** di Controller **Biodata.php** folder **application/controllers/Biodata.php** untuk mengambil data berdasarkan id yang diterima dari parameter URL dan me-load view untuk menampilkan datanya

```
public function detail($id)
{
    $data['title']  = 'Halaman Detail Biodata';
    $data['menu']   = 'biodata';
    $data['biodata'] = $this->Biodata_model->get_biodata_by_id($id);

    $this->load->view('templates/header', $data);
    $this->load->view('biodata/detail', $data);
    $this->load->view('templates/footer');
}
```

2. Membuat fungsi get_biodata_by_id() didalam class Model **Biodata_model.php** untuk mengambil isi biodata dari database berdasarkan id yang diminta

```
function get_biodata_by_id($id)
{
    return $this->db->get_where('biodata', ['id'=>$id])->row_array();
}
```

3. Membuat file **detail.php** pada folder **application/views/biodata/detail.php** untuk menampilkan isi biodata

```
<p>ID : <b><?= $biodata['id']; ?></b></p>
<p>Nama : <b><?= $biodata['nama']; ?></b></p>
<p>Alamat : <b><?= $biodata['alamat']; ?></b></p>
<p>[ <a href="<?= base_url(); ?>biodata">Tampil Data</a> ] </p>
```

4. Kirimkan id biodata yang akan ditampilkan melalui tombol **detail** kolom aksi tabel biodata di file **index.php** folder **application/views/biodata/index.php**

```
<a href="<?= base_url(); ?>biodata/detail/<?= $biodata['id'] ?>">
   Detail
</a>
```

4.10 Update Data

1. Buat method ubah didalam class Controller **Biodata.php**, method ubah akan mengirimkan data yang akan diubah berdasarkan id ke tampilan form ubah, selanjutnya akan menjalankan method update biodata untuk menerima perubahan data

```
public function ubah($id)
{
   $data['title']  = 'Halaman Form Ubah Data';
   $data['menu']   = 'biodata';
   $data['biodata'] = $this->Biodata_model->get_biodata_by_id($id);

   $this->form_validation->set_rules('txtNama', 'Nama', 'required');
   $this->form_validation->set_rules('txtAlamat', 'Alamat', 'required');

   if ($this->form_validation->run() == FALSE)
   {
      $this->load->view('templates/header', $data);
```

```
        $this->load->view('biodata/ubah', $data);
        $this->load->view('templates/footer');
    }
    else
    {
        $this->Biodata_model->update_biodata();
        $this->session->set_flashdata('flash', 'Diubah');
        redirect('biodata');
    }
}
```

2. Membuat tampilan form ubah data pada file baru **ubah.php** di **application/views/biodata/ubah.php**

```
<?php if( validation_errors() ) { ?>
    <div id="eror"><?= validation_errors(); ?></div>
<?php } ?>

<form action="" method="post">
    <input type="hidden" name="id" value="<?= $biodata['id'] ?>" />
    Nama :  <input type="text" name="txtNama" value="<?= $biodata['nama'] ?>" />
    <br/><br/>
    Alamat : <textarea name="txtAlamat"><?= $biodata['alamat'] ?></textarea>
    <br /><br/>
    <input type="submit" value="Submit" />
</form>

<p>[ <a href="<?= base_url(); ?>biodata">Tampil Data</a> ] </p>
```

3. Membuat method update biodata pada class Model Biodata_model.php di folder **application/models/Biodata_model.php**

```
function update_biodata()
{
    $data = [
```

```
        "nama" => $this->input->post('txtNama', true),
        "alamat" => $this->input->post('txtAlamat', true)
    ];

    $this->db->where('id', $this->input->post('id'));
    $this->db->update('biodata', $data);
}
```

4. Mengirimkan id biodata yang akan di update melalui aksi tombol ubah di file **index.php** folder **application/views/biodata/index.php**

```
<a href="<?= base_url(); ?>biodata/ubah/<?= $biodata['id'] ?>">
    Ubah
</a>
```