

**PEMOGRAMAN FRAMEWORK**



**TAMUS TAHIR**

**CODEIGNITER 3 BASIC**

**(Semua Script Di Tes Dengan Menggunakan PHP 7.2)**

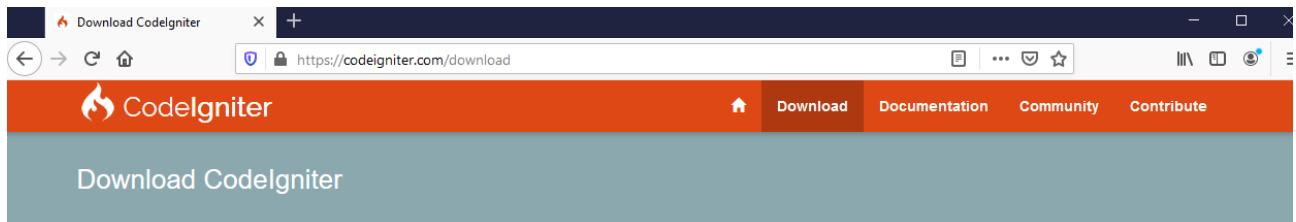
**TAMUS TAHIR**

**2020**

# CODEIGNITER

## INSTALASI

1. Download codeigniter di <https://codeigniter.com/download>
2. Kemudian download codeigniter 3.1.11



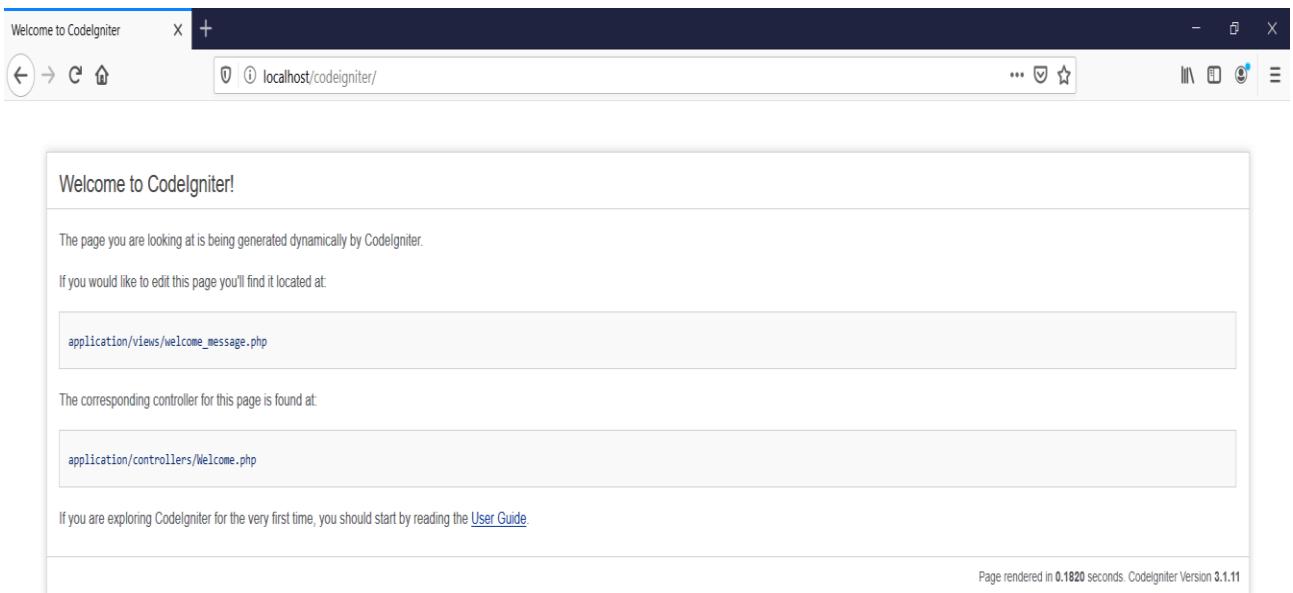
CodeIgniter comes in three flavors: CodeIgniter 3 (current), CodeIgniter 4 (future) and CodeIgniter 2 (legacy)

The screenshot shows two side-by-side sections. On the left, under "CodeIgniter 4", it says "CodeIgniter 4 is the upcoming version of the framework, intended for use with PHP 7.2. Development is underway, and the current version is v4.0.2. You \*could\* download the V4 framework using the button below, but we encourage you to check the [Installation section](#) of the User Guide, to choose from several different ways you can install the framework :)" and includes four green buttons: "Download", "Discuss", "Sources", and "Translations". On the right, under "CodeIgniter 3", it says "CodeIgniter 3.1.11 is the current version of the framework, intended for use with PHP 5.6+. There have been a number of refinements since version 2, notably with the database, session handling and encryption. Development of this version is ongoing." and includes three blue buttons: "Download", "Translations", and "Sources".

3. Ekstrak hasil download ke localhost xampp/htdocs
4. Ubah nama folder nya dari Codeigniter 3.1.11 (tergantung versi yg kamu download) menjadi nama project yang mau di buat (untuk kasus kali ini buat nama folder → codeigniter).

**Disarankan membuat nama dalam huruf kecil semua**

5. Buka folder tersebut di text editor
6. Jalankan XAMPP.
7. Jalankan pada browser <http://localhost/codeigniter/>



## STRUKTUR DIREKTORI

Tedapat dua direktori penting di dalam CI: **application** dan **system**. Selain itu terdapat juga direktori **user\_guide** dan beberapa file. Berikut ini penjelasannya:

1. **application** berisi semua kode aplikasi. Di dalam direktori inilah kita akan menulis semua kode aplikasi kita.
2. **system** berisi kode-kode inti dari Codeinitier. Jangan mengubah apapun di dalam direktori ini. Jika kita ingin upgrade versi, kita cukup me-replace direktori ini dengan yang baru.
3. **tests** berisi kode untuk melakukan unit testing.
4. **user\_guide** berisi dokumentasi codeigniter. Kita bisa menghapus direktori ini saat web sudah jadi.
5. **.editor\_config** berisi konfigurasi untuk teks editor.
6. **.gitignore** berisi daftar file dan folder yang akan diabaikan oleh Git.
7. **composer.json** adalah file yang berisi keterangan project dan keterangan library yang digunakan. File ini dibutuhkan oleh composer.
8. **contributing.md** adalah file yang berisi penjelasan cara berkontribusi di proyek CI. Kita bisa menghapus file ini, apabila web sudah jadi.

9. **license.txt** adalah file yang berisi keterangan lisensi dari CI.
10. **readme.rst** sama seperti file contributing.md file ini berisi penjelasan dan informasi tentang project CI. Kita juga bisa menghapus file ini saat web sudah selesai.
11. **index.php** adalah file utama dari CI. File yang akan dibuka pertamakali saat kita mengakses web.

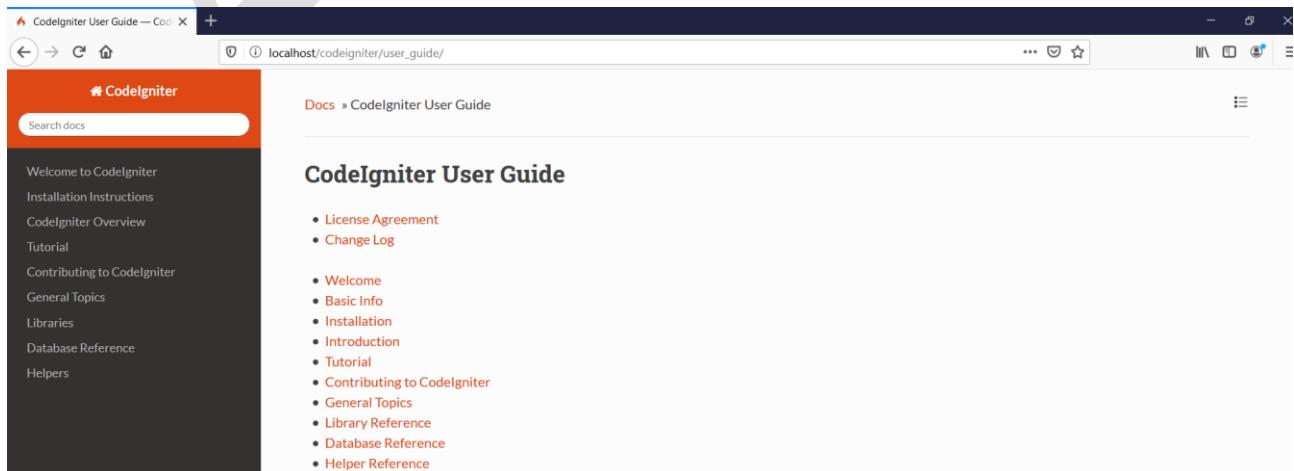
Selanjutnya silahkan buka direktori  application dan perhatikan direktori yang ada di sana.

-  cache berisi cache dari aplikasi.
-  config berisi konfigurasi aplikasi;
  -  autoload.php tempat kita mendefinisikan autoload;
  -  config.php konfigurasi aplikasi;
  -  constants.php berisi konstanta;
  -  database.php konfigurasi database aplikasi;
  -  doctypes.php berisi definisi untuk doctype HTML;
  -  foreign\_chars.php berisi karakter dan simbol;
  -  hooks.php berisi konfigurasi hooks;
  -  index.html untuk mencegah direct access;
  -  memcached.php untuk berisi konfigurasi untuk memcached;
  -  migration.php konfigurasi untuk migrasi;
  -  mimes.php berisi definisi tipe file;
  -  profiler.php konfigurasi untuk profiler;
  -  routers.php tempat kita menulis route aplikasi;
  -  smileys.php berisi kode untuk emoji;
  -  user\_agents.php berisi definisi untuk user agents.

- ❑ controller berisi kode untuk controller.
- ❑ core berisi kode untuk custom core.
- ❑ helpers berisi fungsi-fungsi helper.
- ❑ hooks berisi kode untuk script hook.
- ❑ language berisi string untuk bahasa, apabila web mendukung multibahasa.
- ❑ libraries berisi library.
- ❑ logs berisi logs dari aplikasi.
- ❑ models berisi kode untuk model.
- ❑ thrid\_party berisi library dari pihak ketiga.
- ❑ views berisi kode untuk view.
- ❑ index.html file html untuk mencegah direct access.

## USER GUIDE

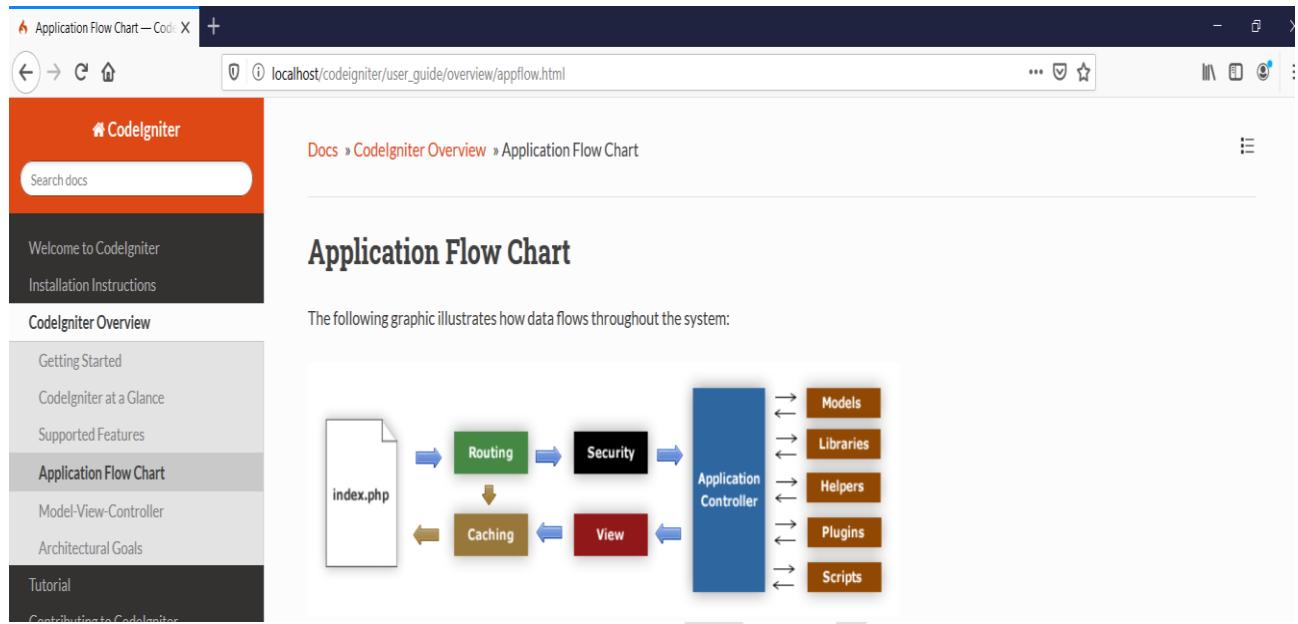
Secara default Codeigniter menyediakan panduan cara penggunaan codeigniter pada folder codeigniter yang anda download. Yaitu terdapat pada folder user\_guide dan untuk mempelajari panduan dari penggunaan codeigniter anda dapat langsung menjalankannya pada browser dengan alamat [http://localhost/codeigniter/user\\_guide/](http://localhost/codeigniter/user_guide/)



## APP FLOW CI

Untuk melihat alur kerja codeigniter kita dapat mengaksesnya pada url berikut:

[http://localhost/codeigniter/user\\_guide/overview/appflow.html](http://localhost/codeigniter/user_guide/overview/appflow.html)



Alur kerjanya seperti ini:

1. User mengirim request ke web;
2. File yang pertama kali dieksekusi adalah index.php;
3. Lalu dari index.php, request akan diteruskan oleh routers.php;
4. routers.php akan mencari cache di server, apabila terdapat cache maka cache itu yang akan dikirim sebagai balasan (response). Apabila tidak ada cache barulah request diteruskan ke Controller;
5. Sebelum request sampai pada controller akan diperiksa terlebih dahulu apakah membutuhkan security (contoh login) atau tidak.
6. Controller akan bertanggung jawab untuk mengambil data dari Model dan me-rendernya ke dalam View dengan menggunakan library, plugin, dan helper yang ada.
7. Hasil render (view) dikirim ke pengguna dan disimpan dalam cache, apabila fitur cache aktif;
8. Selesai.

## LIBRARY

Untuk melihat library codeigniter kita dapat mengaksesnya pada url berikut:

[http://localhost/codeigniter/user\\_guide/libraries/index.html](http://localhost/codeigniter/user_guide/libraries/index.html)

The screenshot shows a browser window displaying the 'Libraries' section of the CodeIgniter 3.1.11 documentation. The URL in the address bar is 'localhost/codeigniter/user\_guide/libraries/index.html'. The page has a dark header with the 'CodeIgniter' logo and a search bar. Below the header is a sidebar with links to 'Welcome to CodeIgniter', 'Installation Instructions', 'CodeIgniter Overview', 'Tutorial', 'Contributing to CodeIgniter', and 'General Topics'. A 'Libraries' section is expanded, showing a list of available classes: Benchmarking Class, Caching Driver, Calendering Class, Shopping Cart Class, Config Class, Email Class, Encrypt Class, Encryption Library, File Uploading Class, Form Validation, FTP Class, Image Manipulation Class, Input Class, Javascript Class, Language Class, Loader Class, and Migrations Class. The main content area is titled 'Libraries' and lists the same items.

Library adalah sekumpulan kelas dan fungsi yang dibuat untuk membantu pengembang aplikasi untuk dapat membangun aplikasi dengan lebih cepat dan lebih efisien. Pada umumnya saat kita membuat aplikasi web ada beberapa kelas yang hampir selalu digunakan, sehingga kelas-kelas tersebut dapat diatur supaya secara otomatis di-load oleh sistem dan dapat langsung digunakan.

Pada CodeIgniter library dibagi menjadi 2 yaitu library yang bersifat global dan library yang dapat dibuat sendiri sesuai kebutuhan. Library global terdiri dari kelas dan fungsi-fungsi yang telah disediakan oleh CodeIgniter, dan terletak pada folder `system/libraries`. Sedangkan library yang kita buat sendiri sesuai dengan kebutuhan ditempatkan pada folder `application/libraries`.

Beberapa library yang wajib diketahui oleh pengembang di antaranya adalah:

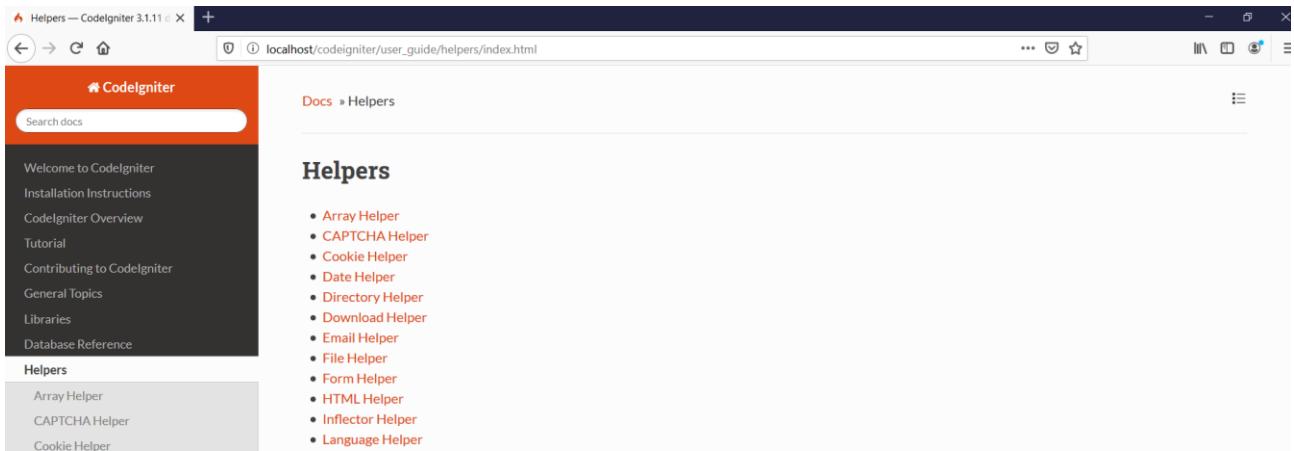
1. **Database**, library yang digunakan untuk mengakses database dan melakukan pengolahan data yang ada di dalam database. Database yang di dukung oleh CodeIgniter adalah mysql, mssql, oracle dan postgres. Sedangkan database yang tidak didukung secara langsung dapat dijembatani dengan driver odbc.

2. **Input**, library yang digunakan untuk menangani dan memproses data-data yang berasal dari form. Misalnya apabila kita menggunakan form untuk memasukan data maka library ini harus di-load supaya dapat melakukan pemrosesan data form.
3. **File Uploading**, library yang digunakan apabila kita akan membangun web yang dapat mengunggah (upload) file ke dalam web. Misalkan kita menginginkan supaya di dalam web kita ada fitur yang dapat digunakan untuk memasukkan file gambar ke dalam aplikasi web kita, maka digunakanlah library ini.
4. **Session**, library yang digunakan untuk memelihara informasi status mengenai pengguna. Sebagai contoh misalkan kita membangun suatu website dimana pengunjung website tersebut harus melakukan proses login terlebih dahulu untuk masuk ke dalam suatu halaman, maka pada situasi seperti ini, library session harus di-load supaya kita dapat memelihara state dari pengunjung, sampai pengunjung tersebut logout.
5. **URI Class**, library ini berisi fungsi-fungsi yang membantu kita untuk mendapatkan informasi dari URI pada alamat web kita.
6. **Validation**, library ini digunakan untuk melakukan validasi terhadap form input yang ada pada aplikasi web kita.
7. **Pagination**, library ini berguna pada saat kita memiliki banyak data yang harus ditampilkan. Misalkan kita memiliki 100 data, dimana ke-100 data ini akan ditampilkan ke dalam 10 halaman (10 data / halaman). Untuk membuat 10 halaman yang masing-masing memuat 10 data dan masing-masing halaman terhubung satu sama lain, maka pagination merupakan library yang tepat untuk digunakan.

## HELPER

Untuk melihat library codeigniter kita dapat mengaksesnya pada url berikut:

[http://localhost/codeigniter/user\\_guide/helpers/index.html](http://localhost/codeigniter/user_guide/helpers/index.html)



Helper juga berfungsi untuk membantu pengembang membangun aplikasi secara lebih cepat dan efisien. Setiap helper bisa terdiri dari beberapa fungsi, dimana setiap fungsi dari helper melakukan satu pekerjaan yang spesifik tanpa ada ketergantungan terhadap fungsi yang lain.

Helper biasanya disimpan dalam folder system/helpers, atau di dalam folder system/application/helpers. CodeIgniter akan terlebih dulu mencari helper di dalam folder system/application/helpers, jika helper yang dicari tidak ditemukan pada folder tersebut, baru kemudian dicari pada folder system/helpers.

Untuk menggunakan helper, ada dua cara yang dapat dilakukan, yaitu :

1. Melalui konfigurasi pada file autoload.php. Konfigurasi pada file autoload.php untuk melakukan proses autoloading terhadap helper-helper yang akan digunakan adalah sebagai berikut:

```
$autoload['helper'] = array('url','form','file');
```

2. Melakukan loading pada setiap controller yang akan menggunakan helper, dilakukan dengan sintak sebagai berikut:

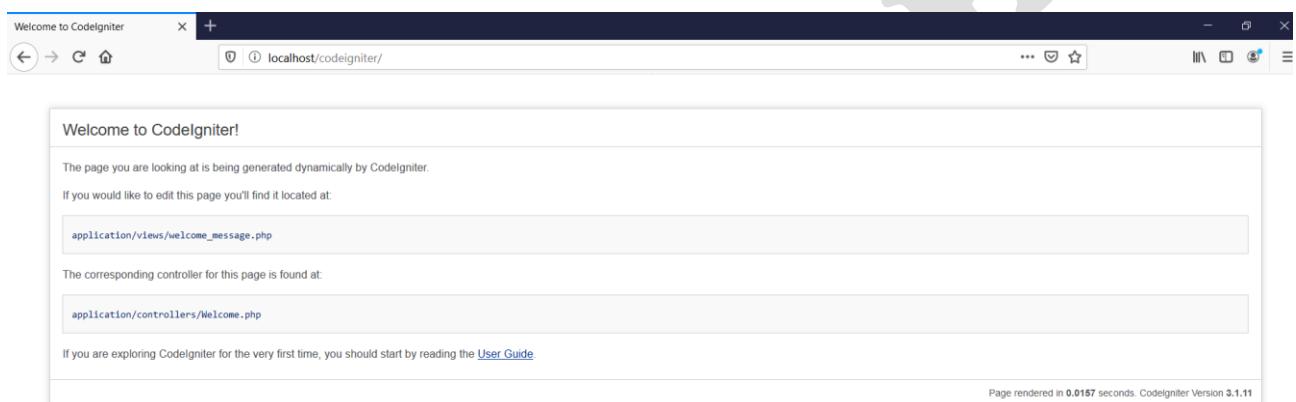
```
$this->load->helper('nama_helper');
```

## Contoh helper:

- a. URL helper : membantu dalam pembuatan link.
- b. Form helper : membantu untuk membuat element-element form.
- c. Text helper : membantu untuk pekerjaan berformat text.
- d. Cookie helper : membantu untuk penanganan cookies.
- e. File helper : membantu untuk kerja dengan file.

## HALAMAN DEFAULT

Halaman default yang tampil ketika kita telah selesai menginstal CI yaitu sebagai berikut:



Alur kerjanya adalah sebagai berikut:

1. Diawali dari file index.php yang kemudian menjalankan pengaturan routes.php yang terletak di application/config/routes.php

A screenshot of a code editor showing the "routes.php" file under the "application/config" directory. The code is as follows:

```
<?php  
defined('BASEPATH') or exit('No direct script access allowed');  
  
$route['default_controller'] = 'welcome';  
$route['404_override'] = '';  
$route['translate_uri_dashes'] = FALSE;
```

- a. `defined('BASEPATH') OR exit('No direct script access allowed');`, syntax di atas berfungsi untuk mencegah akses langsung pada file controller dan harus diakses melalui [www.namaweb.com/controller](http://www.namaweb.com/controller).
  - b. Pengaturan routes codeigniter di atas, pengaturan default\_controller di setting controller “welcome”,
  - c. Pengaturan untuk menangani halaman 404 atau halaman yang di tampilkan jika tidak di temukannya data ada url. anda dapat mengatur halaman 404 anda dengan cara memasukkan controller yang ingin anda jadikan untuk menetapkan halaman 404 pada aplikasi anda.
  - d. Pengaturan `$route['translate_uri_dashes']=false` adalah pengaturan untuk menetapkan nilai true atau false untuk izin penggunaan tanda “-” (dash) pada controller di url pada saat di jalankan.
2. File routes.php memanggil controller default yaitu welcome yang terletak di [application/controllers/welcome.php](#)



```

CODEIGNITER
application
> cache
> config
controllers
index.html
Welcome.php
> core
> helpers
> hooks
> language
> libraries
> logs
> models
> third_party
> views
.htaccess
index.html
system
user_guide
.editorconfig

routes.php
Welcome.php

1 <?php
2 defined('BASEPATH') or exit('No direct script access allowed');
3
4 class Welcome extends CI_Controller
5 {
6
7     public function index()
8     {
9         $this->load->view('welcome_message');
10    }
11 }

```

- a. Class Welcome extends CI\_Controller merupakan pendeklarasian class yang bernama *Welcome* yang meng-ekstend class Inti Codeigniter. Peraturan dalam pembuatan class/controller dalam Codeigniter yaitu, nama class harus sama dengan nama file controller dan berawalan dengan huruf besar. (ex: **welcome.php** maka class nya adalah *Class Welcome extends CI\_Controller*)

b. public function index(), merupakan pendeklarasian function dalam *class*. Sama seperti pembuatan php pada umumnya jika kita membuat index.php maka apabila kita membuka parent folder, akan langsung terhubung ke halaman index.php, begitu pula dengan public function index, jika kita mengakses suatu Controller yang memiliki function index. maka Controller yang dipilih akan langsung memproses function index

3. Kemudian akan memanggil view welcome message yang terletak di application/views/welcome\_message.php



```
<div id="container">
    <h1>Welcome to CodeIgniter!</h1>

    <div id="body">
        <p>The page you are looking at is being generated dynamically by CodeIgniter.</p>

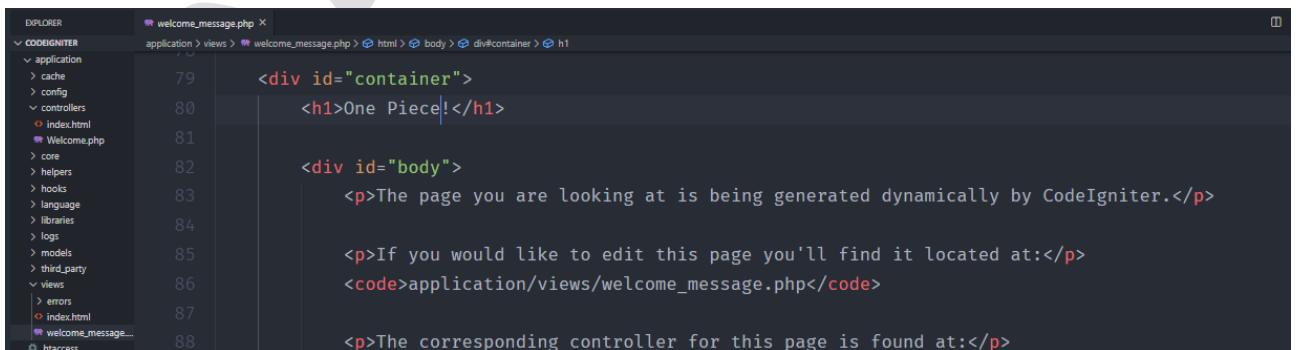
        <p>If you would like to edit this page you'll find it located at:</p>
        <code>application/views/welcome_message.php</code>

        <p>The corresponding controller for this page is found at:</p>
        <code>application/controllers/Welcome.php</code>

        <p>If you are exploring CodeIgniter for the very first time, you should start by reading
        the <a href="user_guide/">User Guide</a>.</p>
    </div>

    <p class="footer">Page rendered in <strong>{elapsed_time}</strong> seconds. <?php echo
    (ENVIRONMENT === 'development') ? 'CodeIgniter Version <strong>' . CI_VERSION . '</strong>'
    : '' ?></p>
</div>
```

4. Untuk membuktikannya kita dapat mengedit file tersebut. Contoh:



```
<div id="container">
    <h1>One Piece!</h1>

    <div id="body">
        <p>The page you are looking at is being generated dynamically by CodeIgniter.</p>

        <p>If you would like to edit this page you'll find it located at:</p>
        <code>application/views/welcome_message.php</code>

        <p>The corresponding controller for this page is found at:</p>
    </div>
```

## 5. Tampilan pada browser juga akan berubah



## MEMBUAT CONTROLLER

1. Buat file baru pada application/controllers dengan nama Home.php

```
EXPLORER   Home.php ×
application > controllers > Home.php > ...
1  <?php
2  defined('BASEPATH') or exit('No direct script access allowed');
3
4  class Home extends CI_Controller
5  {
6
7      public function index()
8      {
9          echo 'ini adalah controller home method index';
10     }
11
12     public function page()
13     {
14         echo 'ini adalah controller home method page';
15     }
16 }
```

A screenshot of a code editor showing the 'Home.php' file under the 'controllers' directory. The file contains PHP code defining a 'Home' controller that extends 'CI\_Controller'. It has two methods: 'index()' which echoes 'ini adalah controller home method index', and 'page()' which echoes 'ini adalah controller home method page'. The code editor interface includes an 'EXPLORER' sidebar on the left showing the project structure.

2. Perhatikan pada controller yang kita buat di atas. Pertama kali yang harus di lakukan adalah meng-extends controller baru ini dengan CI\_Controller.

*class Home extends CI\_Controller*

3. Nama class harus di awali dengan huruf besar
4. Nama class harus sesuai dengan nama file controller
5. Untuk menjalankan method index anda bisa mengaksesnya dengan alamat

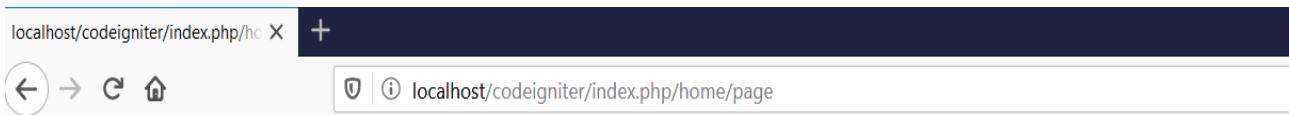
<http://localhost/codeigniter/index.php/home>



ini adalah controller home method index

6. Untuk menjalankan method index anda bisa mengaksesnya dengan alamat

<http://localhost/codeigniter/index.php/home/page>



## MENGHILANGKAN INDEX.PHP

1. Untuk mengakses sebuah controller kita harus menambahkan index.php contoh:

<http://localhost/codeigniter/index.php/home>, hal ini membuat url menjadi tidak menarik

2. Jika kita tidak menggunakan index.php maka hasilnya akan eror contoh:

<http://localhost/codeigniter/home>



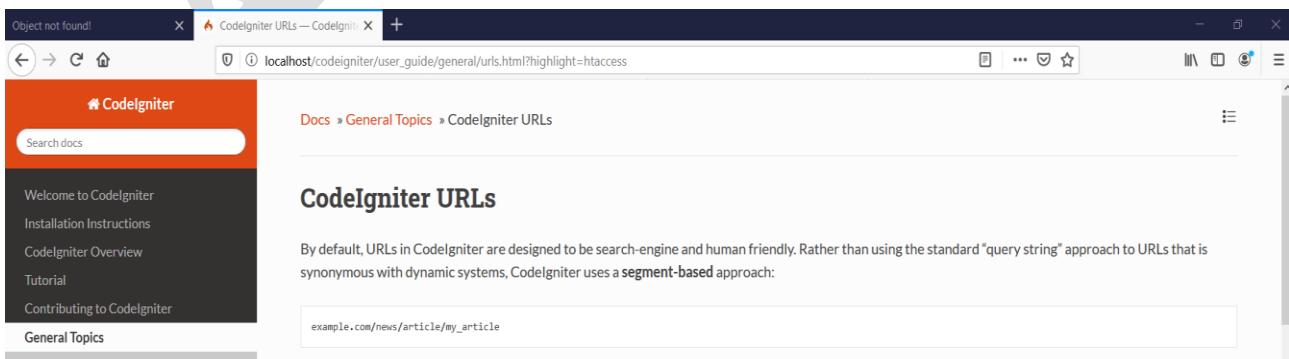
## Object not found!

The requested URL was not found on this server. If you entered the URL manually please check your spelling and try again.

If you think this is a server error, please contact the [webmaster](#).

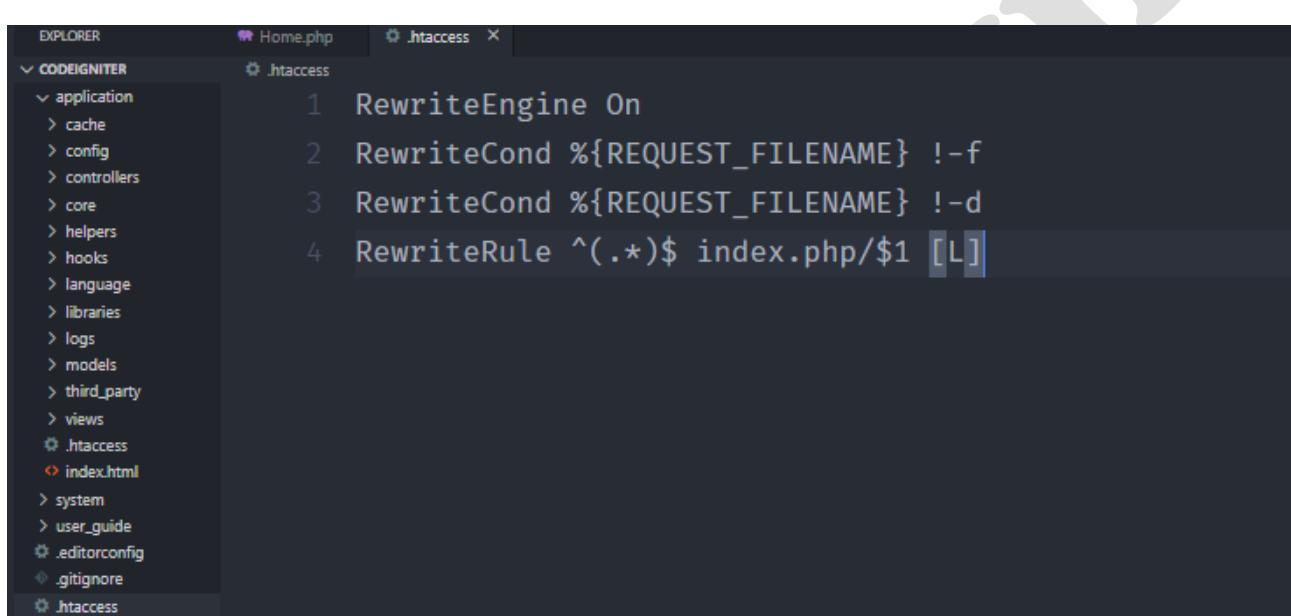
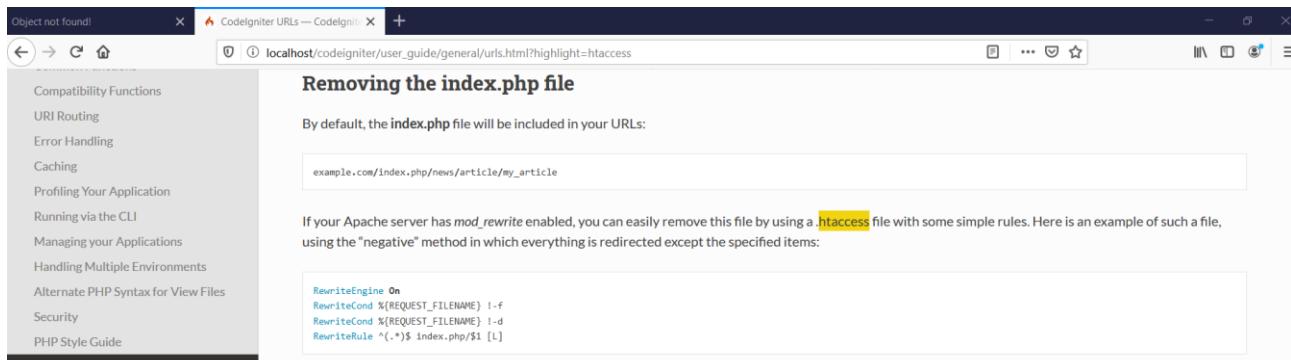
3. Untuk menghilangkan index.php agar alamat url kita terlihat menarik kita perlu membuat sebuah file .htaccess dapat dilihat pada

[http://localhost/framework/user\\_guide/general/urls.html?highlight=htaccess](http://localhost/framework/user_guide/general/urls.html?highlight=htaccess)



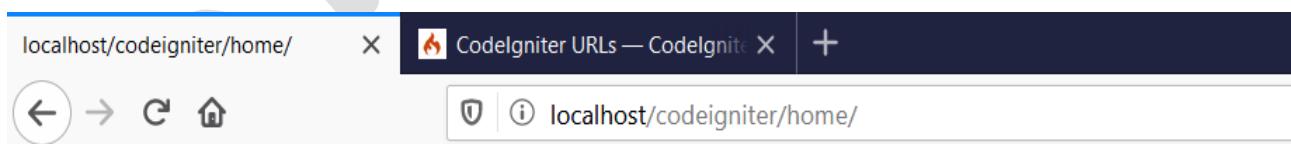
4. Buatfile .htaccess pada directory root codeigniter atau folder paling luar

5. Kemudian copy script berikut



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

6. Coba jalankandi url: <http://localhost/codeigniter/home/>



ini adalah controller home method index

## MENGGANTI CONTROLLER DEFAULT

1. Pada saat kita amengakses <http://localhost/codeigniter/>, controller yang pertama kali diakses adalah controller welcome

The screenshot shows a browser window titled "CodeIgniter URLs — CodeIgniter". The address bar shows "localhost/codeigniter/". The page content displays the "One Piece!" welcome message, which includes the following text:  
The page you are looking at is being generated dynamically by CodeIgniter.  
If you would like to edit this page you'll find it located at:  
`application/views/welcome_message.php`  
The corresponding controller for this page is found at:

2. Kita akan menggantinya dengan controller Home. Buka dan edit file application/config/routes.php

The screenshot shows a code editor with the "routes.php" file open. The left sidebar shows the project structure under "CODEIGNITER":  
application  
cache  
config  
constants.php  
database.php  
doctypes.php  
foreign\_chars.php  
hooks.php  
index.html  
memcached.php  
migration.php  
mimes.php  
profiler.php  
routes.php  
smilieus.php  
The main pane shows the PHP code for the routes file:

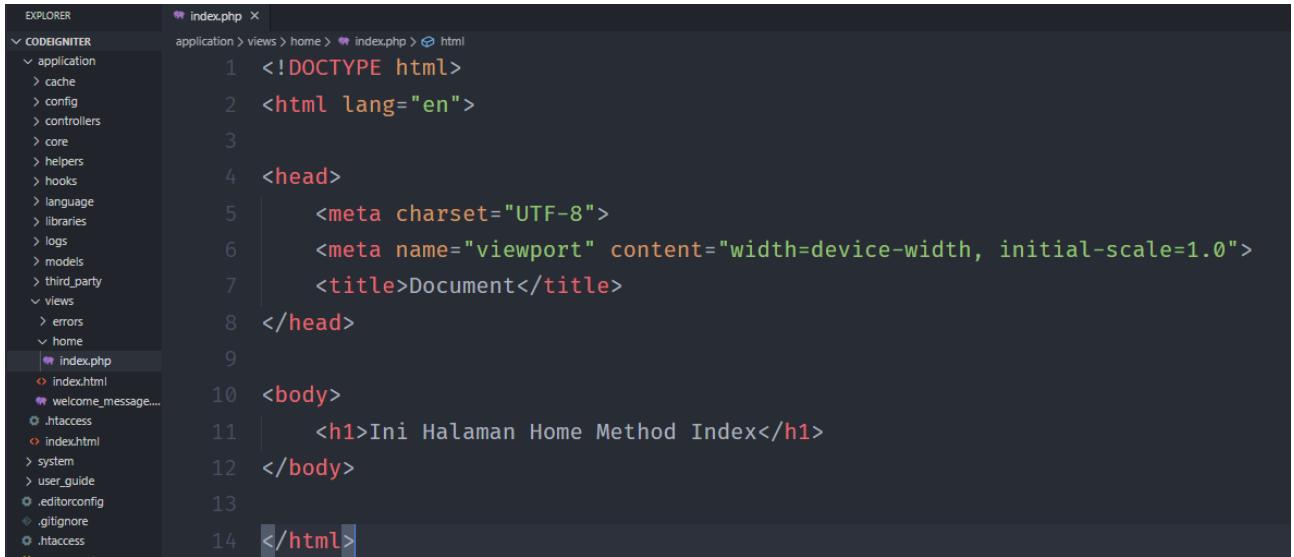
```
<?php  
defined('BASEPATH') or exit('No direct script access allowed');  
  
$route['default_controller'] = 'home';  
$route['404_override'] = '';  
$route['translate_uri_dashes'] = FALSE;
```

3. Lakukan pengetesan di browser, url: <http://localhost/codeigniter/>

The screenshot shows a browser window titled "CodeIgniter URLs — CodeIgniter". The address bar shows "localhost/codeigniter/". The page content displays the modified welcome message: "ini adalah controller home method index"

## MEMBUAT VIEW

1. Buat folder baru pada application/views dengan nama home dan buat file baru index.php



```
EXPLORER index.php
CODEIGNITER application > views > home > index.php > html
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9
10 <body>
11     <h1>Ini Halaman Home Method Index</h1>
12 </body>
13
14 </html>
```

2. Pada controller home di application/controllers/Home.php lakukan pemanggilan view yang akan dibuat



```
EXPLORER index.php Home.php
CODEIGNITER application > controllers > Home.php > ...
1  <?php
2  defined('BASEPATH') or exit('No direct script access allowed');
3
4  class Home extends CI_Controller
5  {
6
7      public function index()
8      {
9          $this->load->view('home/index');
10     }
11 }
```

3. Jalankan di browser: <http://localhost/codeigniter/>



**Ini Halaman Home Method Index**

# CRUD

## KONFIGURASI DATABSE

- Buat database baru dengan nama codeigniter dan table baru dengan nama mahasiswa dengan struktur table sebagai berikut:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<b>id</b>	int(11)			No	None		AUTO_INCREMENT	Change  Drop  More
2	<b>nama</b>	varchar(128)	utf8mb4_general_ci		No	None			Change  Drop  More
3	<b>nim</b>	int(11)			No	None			Change  Drop  More

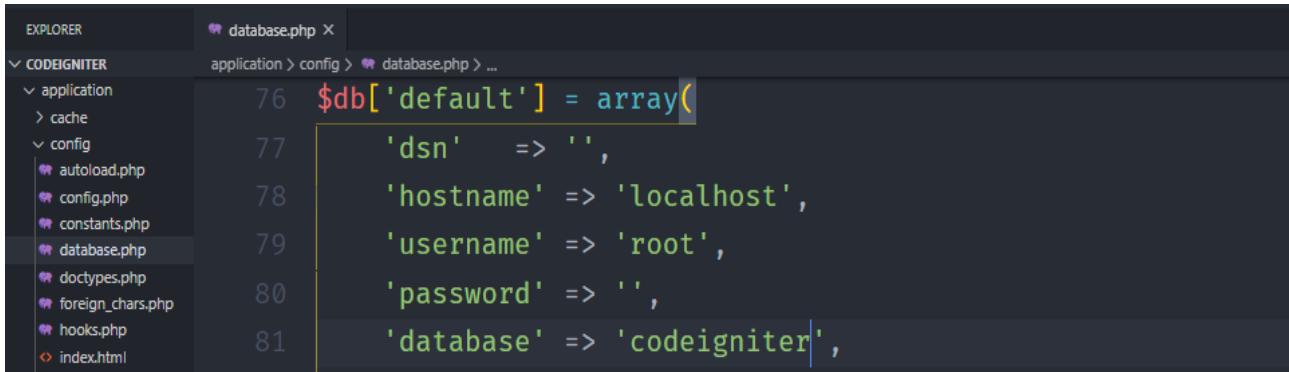
- Kemudian pilih insert pada table tersebut

The screenshot shows the MySQL Workbench interface with the 'Insert' tab selected. There are two rows of input fields for the 'mahasiswa' table. The first row contains: id (empty), nama (Luffy), nim (2020054001). The second row contains: id (empty), nama (empty), nim (2020054002). A 'Go' button is visible at the bottom right of each row.

- Pilih go

+ Options		← T →	▼	id	nama	nim
<input type="checkbox"/>		Edit		1	Luffy	2020054001
<input type="checkbox"/>		Edit		2	Zoro	2020054002

- Pada application/config/database.php isi sesuai database pengaturan anda



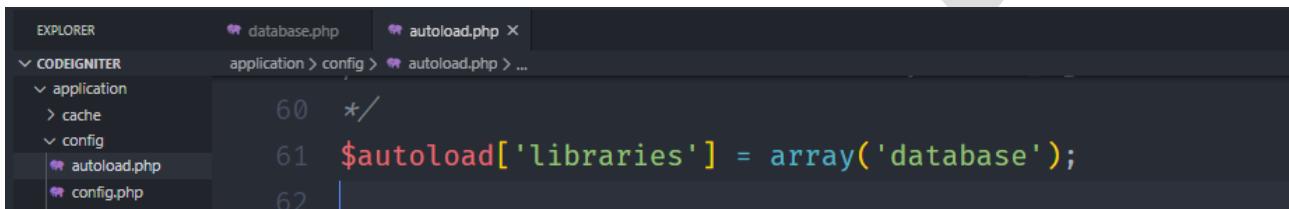
```

EXPLORER          database.php X
CODEIGNITER
  application
    > cache
    > config
      autoload.php
      config.php
      constants.php
      database.php
      doctypes.php
      foreign_chars.php
      hooks.php
    index.html

76 $db['default'] = array(
77     'dsn'      => '',
78     'hostname' => 'localhost',
79     'username' => 'root',
80     'password' => '',
81     'database' => 'codeigniter',

```

- Untuk menggunakan library database kita perlu meload library tersebut, application/config/autoload.php



```

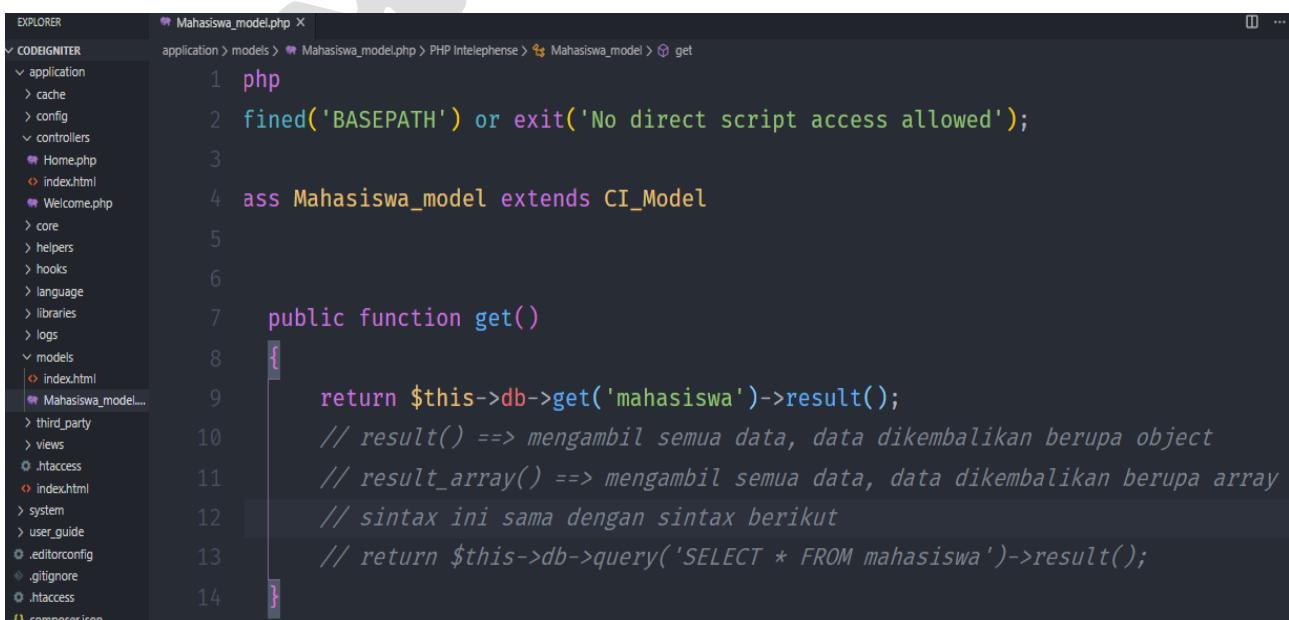
EXPLORER          database.php X  autoload.php X
CODEIGNITER
  application
    > cache
    > config
      autoload.php
      config.php
    index.html

60 */
61 $autoload['libraries'] = array('database');
62

```

## READ

- Buat file baru pada application/models dengan nama Mahasiswa\_model. Disini kita tambahkan kata model untuk membedakan antara model dan controller



```

EXPLORER          Mahasiswa_model.php X
CODEIGNITER
  application
    > cache
    > config
    > controllers
      Home.php
      index.html
      Welcome.php
    > core
    > helpers
    > hooks
    > language
    > libraries
    > logs
    > models
      index.html
      Mahasiswa_model...
    > third_party
    > views
      .htaccess
      index.html
    > system
    > user_guide
      .editorconfig
      .gitignore
      .htaccess
    composer.json

application > models > Mahasiswa_model.php > PHP Intelephense > Mahasiswa_model > get

1  php
2  fined('BASEPATH') or exit('No direct script access allowed');
3
4  ass Mahasiswa_model extends CI_Model
5
6
7  public function get()
8  {
9      return $this->db->get('mahasiswa')->result();
10     // result() ==> mengambil semua data, data dikembalikan berupa object
11     // result_array() ==> mengambil semua data, data dikembalikan berupa array
12     // sintax ini sama dengan sintax berikut
13     // return $this->db->query('SELECT * FROM mahasiswa')->result();
14 }


```

2. Class Mahasiswa\_model kita harus extend ke class CI\_Model
3. Kemudian kita buat fungsi get untuk menagmbil semua data pada table mahasiswa.
4. Buat controller baru dengan nama Mahasiswa.php



EXPLORER      Mahasiswa\_model.php      Mahasiswa.php

```

CODEIGNITER
application > controllers > Mahasiswa.php > PHP Intelephense > Mahasiswa > __construct
1  <?php
2  defined('BASEPATH') or exit('No direct script access allowed');
3
4  class Mahasiswa extends CI_Controller
5  {
6      public function __construct()
7      {
8          // menggunakan construct dari construcrnya CI_controller
9          parent::__construct();
10         // memanggil / load file Mahasiswa_model
11         $this->load->model('Mahasiswa_model');
12     }
13
14     public function index()
15     {
16         // $data['mahasiswa'] ==> akan menerima data dari yang dikirimkan dari
17         // mahasiswa model dengan method get
18         $data['mahasiswa'] = $this->Mahasiswa_model->get();
19
20         // $data ==> kemudian datanya kita kirimkan ke halaman index
21         $this->load->view('mahasiswa/index', $data);
22     }

```

5. Buat folder baru di views dengan nama mahasiswa dan file baru dengan nama index.php



EXPLORER      Mahasiswa\_model.php      Mahasiswa.php      index.php

```

CODEIGNITER
application > views > mahasiswa > index.php > html > body
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9
10 <body>
11     <h1>Data Mahasiswa</h1>
12     <table border="1" cellpadding="10" cellspacing="0">
13         <tr>
14             <th>#</th>
15             <th>Nama</th>
16             <th>NIM</th>
17         </tr>
18         <?php $i = 1; ?>
19         <?php foreach ($mahasiswa as $m) : ?>
20             <tr>
21                 <td><?= $i; ?></td>
22                 <td><?= $m->nama; ?></td>
23                 <td><?= $m->nim; ?></td>
24             </tr>
25             <?php $i++; ?>
26         <?php endforeach; ?>
27     </table>
28 </body>

```

6. Jalankan di browser: <http://localhost/codeigniter/mahasiswa>

A screenshot of a web browser window. The address bar shows 'localhost / 127.0.0.1 / codeigniter'. The title bar says 'Document' and 'CodeIgniter URLs — CodeIgnite'. Below the address bar are standard navigation buttons (back, forward, home). The main content area displays the heading 'Data Mahasiswa' and a table with two rows of data.

## Data Mahasiswa

#	Nama	NIM
1	Luffy	2020054001
2	Zoro	2020054002

7. Kembali ke model, kita akan menangani jika data yang dikembalikan berupa array

A screenshot of a code editor interface. On the left is an 'EXPLORER' sidebar showing the project structure under 'CODEIGNITER'. The 'models' folder contains 'Mahasiswa\_model.php'. The main editor area shows the PHP code for the 'Mahasiswa\_model' class, specifically the 'get()' method which returns a result array from the database.

8. Pada views/mahasiswa/index akan error karena terdapat perbedaan cara mencetak data

A screenshot of a web browser window. The address bar shows 'localhost / 127.0.0.1 / codeigniter'. The title bar says 'Document' and 'CodeIgniter URLs — CodeIgnite'. Below the address bar are standard navigation buttons. The main content area displays the heading 'Data Mahasiswa' and an empty table with two columns: 'Nama' and 'NIM'.

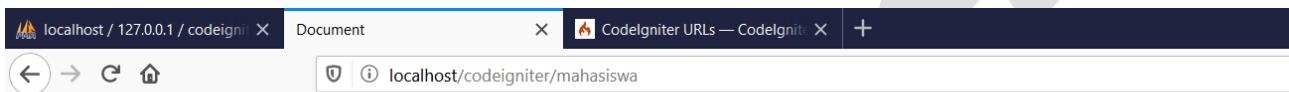
## Data Mahasiswa

#	Nama	NIM
	A PHP Error was encountered Severity: Notice Message: Trying to get property 'nama' of non-object Filename: mahasiswa/index.php	A PHP Error was encountered Severity: Notice Message: Trying to get property 'nim' of non-object Filename: mahasiswa/index.php

## 9. Views/mahasiswa/index.php

```
> hooks          17 | </tr>
> language       18 | <?php $i = 1; ?>
> libraries      19 | <?php foreach ($mahasiswa as $m) : ?>
> logs           20 |   <tr>
> models          21 |     <td><?= $i; ?></td>
> third_party    22 |     <td><?= $m['nama']; ?></td>
> views           23 |     <td><?= $m['nim']; ?></td>
> errors          24 |   </tr>
> home            25 |   <?php $i++; ?>
> mahasiswa      26 | <?php endforeach; ?>
```

## 10. Tes pada browser



## Data Mahasiswa

#	Nama	NIM
1	Luffy	2020054001
2	Zoro	2020054002

11. Kedua cara ini memiliki hasil yang sama, untuk tutorial ini datanya akan kita kembalikan berupa object

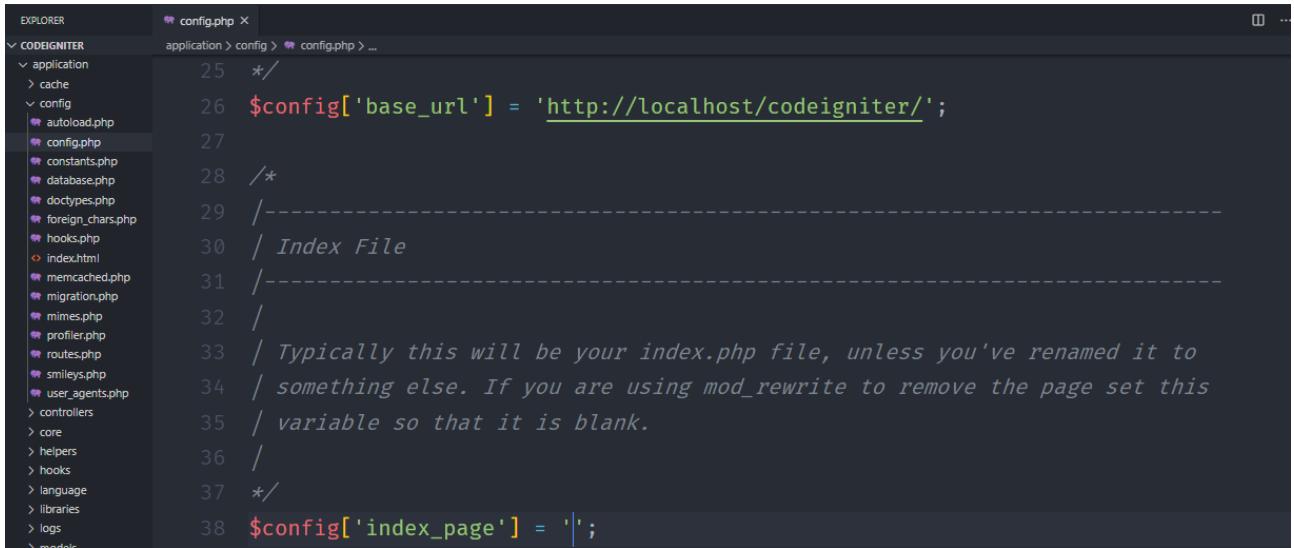
## 12. Kembalikan seperti sebelumnya

```
> models          6 | 
> models          7 |     public function get()
> models          8 |     {
> third_party    9 |         return $this->db->get('mahasiswa')->result();
> views           10 |     }
```

```
> models          18 | <?php $i = 1; ?>
> models          19 | <?php foreach ($mahasiswa as $m) : ?>
> views           20 |   <tr>
> errors          21 |     <td><?= $i; ?></td>
> home            22 |     <td><?= $m->nama; ?></td>
> mahasiswa      23 |     <td><?= $m->nim; ?></td>
> errors          24 |   </tr>
> errors          25 |   <?php $i++; ?>
> errors          26 | <?php endforeach; ?>
```

## BASE URL

1. Set base\_url yaitu pada application/config/config.php tambahkan alamat url kita dan delete index.php pada index\_page



```
EXPLORER config.php x
CODEIGNITER application > config > config.php > ...
25 */
26 $config['base_url'] = 'http://localhost/codeigniter/';
27 /*
28 -----
29 / ----- Index File -----
30 / -----
31 / -----
32 /
33 / Typically this will be your index.php file, unless you've renamed it to
34 / something else. If you are using mod_rewrite to remove the page set this
35 / variable so that it is blank.
36 /
37 */
38 $config['index_page'] = '|';
```

2. Untuk menggunakan base url, kita harus meload helper url



```
EXPLORER config.php x autoload.php x
CODEIGNITER application > config > autoload.php > ...
91 */
92 $autoload['helper'] = array('url');
93
```

## CREATE

1. Pada application/views/home/index.php tambahkan link untuk tambah



```
EXPLORER index.php x
application > views > mahasiswa > index.php > html > body > table > tr > th
5 <meta charset="UTF-8">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>Document</title>
8 </head>
9
10 <body>
11 <h1>Data Mahasiswa</h1>
12 <hr>
13
14 <a href="= base_url('mahasiswa/tambah'); ?&gt;"&gt;Tambah&lt;/a&gt;
15 &lt;br&gt;&lt;br&gt;</pre
```

2. Coba jalankan di browser dan klik kanan view page source

A screenshot of a web browser window titled "Document". The address bar shows "localhost/codeigniter/mahasiswa". The main content area displays a table with two rows:

#	Nama	NIM
1	Luffy	2020054001
2	Zoro	2020054002

A screenshot of a browser window showing the page source code. The address bar shows "view-source:localhost/codeigniter/mahasiswa". The source code includes an anchor tag with a href attribute pointing to "http://localhost/codeigniter/mahasiswa/tambah". This specific line is highlighted with a red rectangular box.

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <h1>Data Mahasiswa</h1>
12   <hr>
13
14   <a href="http://localhost/codeigniter/mahasiswa/tambah">Tambah</a>
15 <br><br>
```

3. Disini alamat url tambah telah terbentuk

4. Buat method tambah pada controller mahasiswa

A screenshot of a code editor showing the "Mahasiswa.php" file under the "controllers" directory. The code defines a public function "tambah()". A portion of the code is highlighted with a red box.

```
EXPLORER Mahasiswa.php X
CODEIGNITER application > controllers > Mahasiswa.php ...
22
23   public function tambah()
24   {
25     $this->load->view('mahasiswa/tambah');
26 }
```

5. Buat file baru tambah.php pada views/mahasiswa

The screenshot shows a code editor with the file 'tambah.php' open. The file is located in the 'views/mahasiswa' directory of a CodeIgniter application. The code contains HTML and PHP code for a form to add student data. It includes meta tags, a title, a body section with a heading, a form opening tag, labels for 'Nama' and 'NIM', and a submit button.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Tambah Data Mahasiswa</h1>
    <hr>
    <?=> form_open(); ?>
    <label for="nama">Nama</label>
    <input type="text" name="nama" id="nama">
    <br>
    <label for="nim">NIM</label>
    <input type="text" name="nim" id="nim">
    <br>
    <button type="submit">Tambah</button>
</form>
```

6. Untuk menggunakan form\_open(), kita harus meload helper form

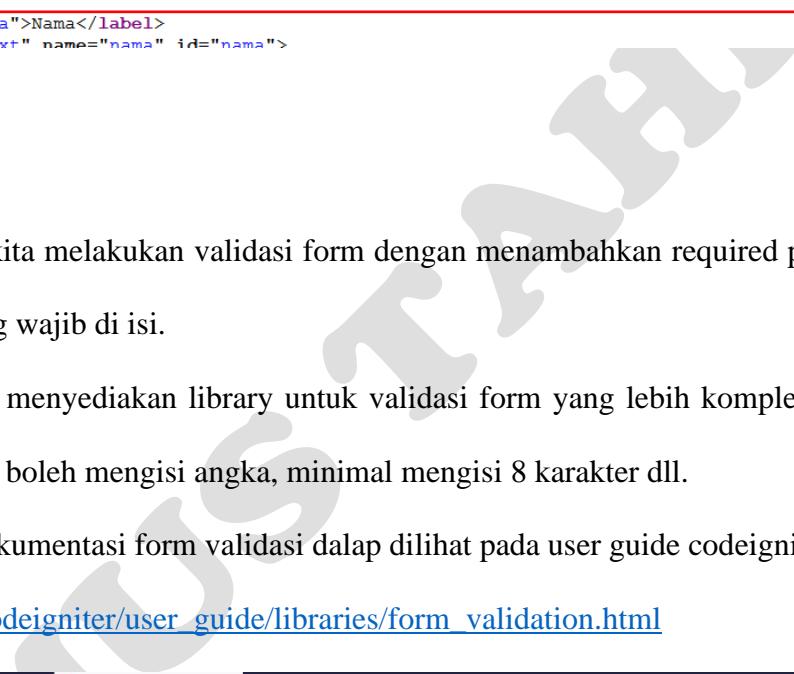
The screenshot shows a code editor with the file 'autoload.php' open. The file is located in the 'config' directory of a CodeIgniter application. It contains PHP code that defines the 'helper' autoload array, which includes the 'url' and 'form' helpers.

```
$autoload['helper'] = array('url', 'form');
```

7. Test pada browser



8. Klik kanan pada browser dan view page source untuk melihat hasil dari form\_open()



Document X http://localhost/codeigniter/mahasiswa +

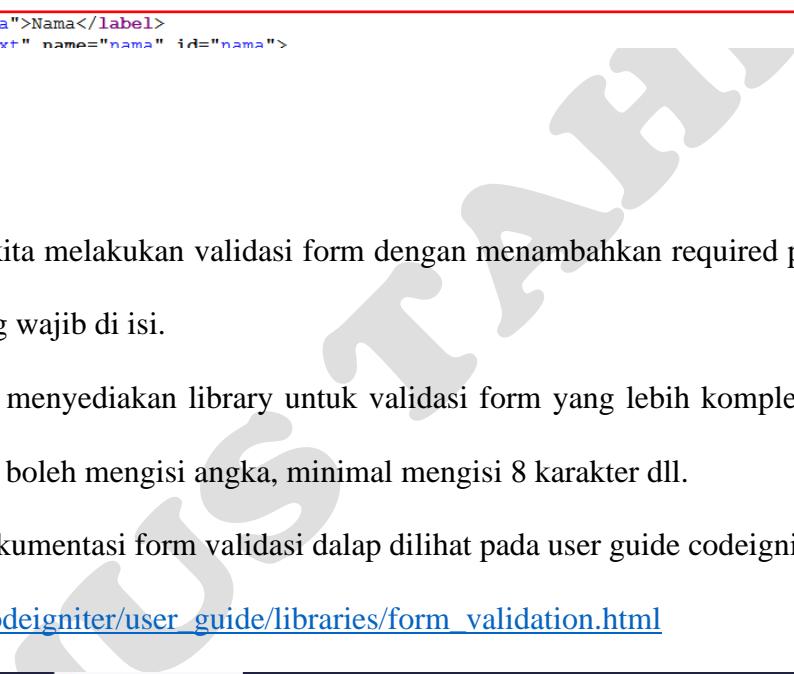
view-source:http://localhost/codeigniter/mahasiswa/tambah

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <h1>Tambah Data Mahasiswa</h1>
12   <hr>
13
14
15   <form action="http://localhost/codeigniter/mahasiswa/tambah" method="post" accept-charset="utf-8">
16
17     <label for="nama">Nama</label>
18     <input type="text" name="nama" id="nama">
```

## VALIDASI

1. Pada php native kita melakukan validasi form dengan menambahkan required pada masing-masing input yang wajib di isi.
2. Codeigniter telah menyediakan library untuk validasi form yang lebih kompleks, misalnya wajib diisi, hanya boleh mengisi angka, minimal mengisi 8 karakter dll.
3. Untuk melihat dokumentasi form validasi dalap dilihat pada user guide codeigniter,

[http://localhost/codeigniter/user\\_guide/libraries/form\\_validation.html](http://localhost/codeigniter/user_guide/libraries/form_validation.html)



Document X Form Validation — CodeIgniter X Form Validation — CodeIgniter +

localhost/codeigniter/user\_guide/libraries/form\_validation.html

Docs » Libraries » Form Validation

## Form Validation

CodeIgniter provides a comprehensive form validation and data prepping class that helps minimize the amount of code you'll write.

Page Contents

- Form Validation
  - Overview
  - Form Validation Tutorial
    - The Form
    - The Success Page
    - The Error Page

## 4. Berikut beberapa contoh rule pada form validation

### Rule Reference

The following is a list of all the native rules that are available to use:

Rule	Parameter	Description	Example
required	No	Returns FALSE if the form element is empty.	
matches	Yes	Returns FALSE if the form element does not match the one in the parameter.	matches[form_item]
regex_match	Yes	Returns FALSE if the form element does not match the regular expression.	regex_match[/regex/]
differs	Yes	Returns FALSE if the form element does not differ from the one in the parameter.	differs[form_item]
is_unique	Yes	Returns FALSE if the form element is not unique to the table and field name in the parameter. Note: This rule requires <b>Query Builder</b> to be enabled in order to work.	is_unique[table.field]

## 5. Untuk menggunakannya kita harus meload library form validation



A screenshot of a code editor showing the CodeIgniter project structure. The left sidebar shows the 'EXPLORER' view with 'CODEIGNITER' selected, showing the 'application' folder containing 'cache', 'config', 'autoload.php', and 'config.php'. The main editor area shows the 'autoload.php' file with the following code:

```
60 */  
61 $autoload['libraries'] = array('database', 'form_validation');
```

## 6. Validasi dilakukan pada controller mahasiswa



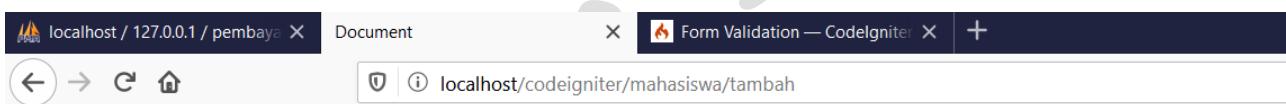
A screenshot of a code editor showing the 'Mahasiswa.php' controller file. The left sidebar shows the 'EXPLORER' view with 'CODEIGNITER' selected, showing the 'application' folder containing various files like 'Home.php', 'Mahasiswa.php', and 'Welcome.php'. The main editor area shows the 'Mahasiswa.php' file with the following code:

```
23 public function tambah()  
24 {  
25     // set_rules 3 parameter  
26     // 1 ==> name dari form input  
27     // 2 ==> tulisan yang akan dicetak jika error  
28     // 3 ==> rulennya apa (boleh lebih dari 1)  
29     $this->form_validation->set_rules('nama', 'Nama', 'required');  
30     $this->form_validation->set_rules('nim', 'Nim', 'required');  
31  
32     // jika validasi bernilai false tampilkan halaman tambah mahasiswa  
33     // validasinya bernilai false ketika pertama membuka halaman tambah  
34     if ($this->form_validation->run() == FALSE) {  
35         $this->load->view('mahasiswa/tambah');  
36     } else {  
37         // jika bernilai true  
38     }  
39 }
```

## 7. Error dari validasi akan dicetak di halaman view tambah mahasiswa

```
EXPLORER Mahasiswa.php tambah.php
CODEIGNITER application > views > mahasiswa > tambah.php > html > body > label
14
15     <?= form_open(); ?>
16
17     <label for="nama">Nama</label>
18     <input type="text" name="nama" id="nama">
19     <!-- error akan dicetak ketika validasinya tidak sesuai rule -->
20     <?= form_error('nama') ?>
21     <br>
22
23     <label for="nim">NIM</label>
24     <input type="text" name="nim" id="nim">
25     <?= form_error('nim') ?>
26     <br>
27
28     <button type="submit">Tambah</button>
29
30 </form>
```

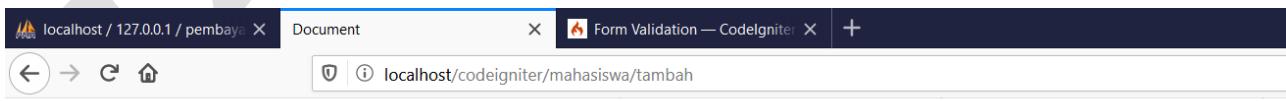
## 8. Test browser



## Tambah Data Mahasiswa

Nama   
NIM

## 9. Biarkan formnya kosong dan tekan tombol tambah



## Tambah Data Mahasiswa

Nama

The Nama field is required.

NIM

The Nim field is required.

## 10. Errornya bisa kita ganti dengan Bahasa Indonesia

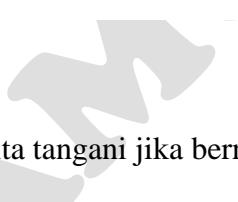
- ➔ system/language/English/form\_validation.php
- ➔ disarankan backup filenya terlebih dahulu
- ➔ {field} jangan diubah



A screenshot of a code editor showing the file `form_validation_lang.php`. The code defines language messages for form validation. Lines 40 and 41 show the Indonesian translation for 'form\_validation\_required': `$lang['form_validation_required'] = 'Input {field} wajib di isi.';`. Lines 42 and 43 show the Indonesian translation for 'form\_validation\_isset': `$lang['form_validation_isset'] = 'The {field} field must have a value.'`.

```
EXPLORER Mahasiswa.php tambah.php form_validation_lang.php ...
CODEIGNITER application > system > language > english > form_validation_lang.php > ...
36 * @since Version 1.0.0
37 * @filesource
38 */
39 defined('BASEPATH') or exit('No direct script access allowed');
40 // {field} ==> jangan diubah
41 $lang['form_validation_required'] = 'Input {field} wajib di isi.';
42 $lang['form_validation_isset'] = 'The {field} field must have a value.'
43 $lang['form_validation_valid_email'] = 'The {field} field must be a valid email address.'
```

## 11. Test pada browser



A screenshot of a web browser showing a form titled "Tambah Data Mahasiswa". The form has two fields: "Nama" and "NIM". Both fields are empty. Below the "Nama" field is the error message "Input Nama wajib di isi.". Below the "NIM" field is the error message "Input Nim wajib di isi.". At the bottom right is a "Tambah" button.

localhost / 127.0.0.1 / pembaya Document Form Validation — CodeIgniter

Tambah Data Mahasiswa

Nama

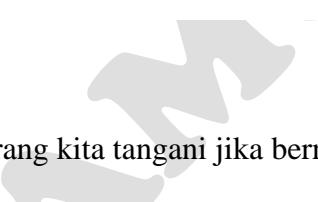
Input Nama wajib di isi.

NIM

Input Nim wajib di isi.

**Tambah**

## 12. Sekarang kita tangani jika bernilai true, datanya akan kita kirim ke database



A screenshot of a code editor showing the file `Mahasiswa.php` in the `controllers` folder. The code contains logic for form validation. It checks if validation is false (error) and then loads the add view. If validation is true (success), it inserts data into the database using the `Mahasiswa_model` and then redirects to the main `mahasiswa` page.

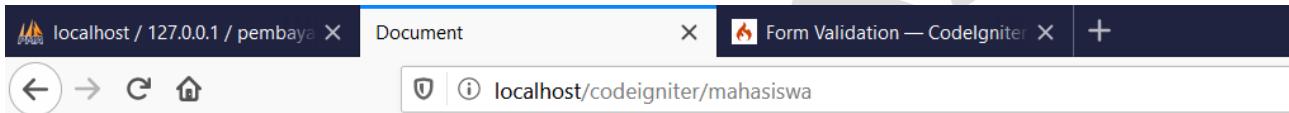
```
EXPLORER Mahasiswa.php tambah.php ...
CODEIGNITER application > controllers > Mahasiswa.php > PHP Intelephense > Mahasiswa
32 // jika validasi belum true maka tampilkan halaman tambah mahasiswa
33 // validasinya bernilai false ketika pertama membuka halaman tambah
34 if ($this->form_validation->run() == FALSE) {
35     $this->load->view('mahasiswa/tambah');
36 } else {
37     // jika bernilai true
38     $this->Mahasiswa_model->insert();
39     // redirect = mengalihakan ke halaman lain
40     redirect('mahasiswa');
41 }
42 }
```

### 13. Buat method insert pada mahasiswa model



```
EXPLORER Mahasiswa.php Mahasiswa_model.php tambah.php  
CODEIGNITER application > models > Mahasiswa_model.php > PHP Intelephense > Mahasiswa_model > insert  
application > cache > config > controllers > core > helpers > hooks > language > libraries > logs > models > index.html > Mahasiswa_model... > third_party > views > errors > home > mahasiswa  
16 public function insert()  
17 {  
18     // tampung hasil input ke dalam sebuah variabel data  
19     $data = [  
20         'nama' => $this->input->post('nama'),  
21         'nim' => $this->input->post('nim'),  
22     ];  
23     // insert (nama tabel, datanya apa)  
24     $this->db->insert('mahasiswa', $data);  
25 }
```

### 14. Test pada browser, jika tidak ada error datanya akan terkirim ke database



## Data Mahasiswa

Tambah

#	Nama	NIM
1	Luffy	2020054001
2	Zoro	2020054002
3	Sanji	2020054003

### 15. Disini masih terdapat kekurangan karena tidak terdapat notifikasi jika datanya berhasil terkirim

16. Untuk itu kita butuh flash data → pesan yang muncul pada kondisi tertentu dan akan hilang ketika browsernya di refresh

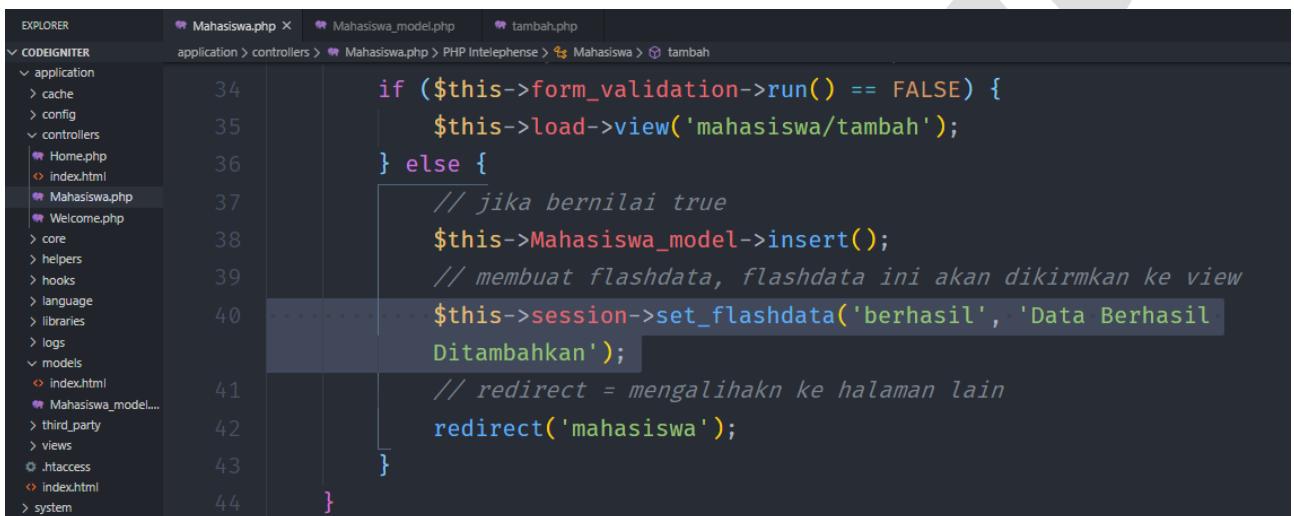
## FLASH DATA

1. Flashdata bersifat sementara mempunyai 2 parameter yaitu nama flashdata dan pesan yang akan ditampilkan
2. Untuk menggunakan flashdata kita perlu meload library session



```
EXPLORER Mahasiswa.php Mahasiswa_model.php autoload.php ...  
CODEIGNITER application > config > autoload.php ...  
application > cache > config > autoload.php  
config > config.php  
60 */  
61 $autoload['libraries'] = array('database', 'form_validation', 'session');  
62
```

3. Kemudian kita set flashdatanya pada controller



```
EXPLORER Mahasiswa.php Mahasiswa_model.php tambah.php ...  
CODEIGNITER application > controllers > Mahasiswa.php > PHP Intelephense > Mahasiswa > tambah  
application > cache > config > controllers > Home.php > index.html > Mahasiswa.php > Welcome.php  
core > helpers > hooks > language > libraries > logs > models > index.html > Mahasiswa_model... > third_party > views > .htaccess > index.html > system  
34 if ($this->form_validation->run() == FALSE) {  
35     $this->load->view('mahasiswa/tambah');  
36 } else {  
37     // jika bernilai true  
38     $this->Mahasiswa_model->insert();  
39     // membuat flashdata, flashdata ini akan dikirimkan ke view  
40     $this->session->set_flashdata('berhasil', 'Data Berhasil  
Ditambahkan');  
41     // redirect = mengalihakan ke halaman lain  
42     redirect('mahasiswa');  
43 }  
44 }
```

4. Cetak flashdatanya pada view index



```
EXPLORER Mahasiswa.php Mahasiswa_model.php tambah.php index.php ...  
CODEIGNITER application > views > mahasiswa > index.php > html > body  
application > cache > config > controllers > core > helpers > hooks > language > libraries > logs > models > third_party > views > errors > home > mahasiswa > index.php > tambah.php  
11 <h1>Data Mahasiswa</h1>  
12 <hr>  
13  
14 <!-- operator ternary -->  
15 <!-- jika ada flasdata berhasil cetak flasdata, jika tidak kosongkan -->  
16 <?= $this->session->flashdata('berhasil') ? $this->session->flashdata  
('berhasil') : ''; ?>  
17  
18 <a href="<?= base_url('mahasiswa/tambah'); ?>">Tambah</a>  
19 <br><br>
```

5. Lakukan pengujian di browser

A screenshot of a web browser window. The address bar shows 'localhost/codeigniter/mahasiswa'. The page title is 'Data Mahasiswa'. Below the title, a message says 'Data Berhasil Ditambahkan [Tambah](#)'. A table with columns '#', 'Nama', and 'NIM' is shown, but it is empty.

## Data Mahasiswa

Data Berhasil Ditambahkan [Tambah](#)

#	Nama	NIM

6. Flashdatanya berhasil dicetak dan ketika direfresh browser pesannya akan hilang

## UPDATE

1. Pada views/mahasiswa/index.php tambahkan kolom aksi dan link untuk ubah dan hapus

A screenshot of a CodeIgniter application structure. The 'views' folder contains 'mahasiswa' which has 'index.php' and 'tambah.php'. The 'index.php' file contains PHP code for displaying a table of student data with edit and delete links. The 'tambah.php' file is also visible.

```
<table border="1" cellpadding="10" cellspacing="0">
    <tr>
        <th>#</th>
        <th>Nama</th>
        <th>NIM</th>
        <th>Aksi</th>
    </tr>
    <?php $i = 1; ?>
    <?php foreach ($mahasiswa as $m) : ?>
        <tr>
            <td><?= $i; ?></td>
            <td><?= $m->nama; ?></td>
            <td><?= $m->nim; ?></td>
            <td>
                <a href=<?= base_url('mahasiswa/ubah/' . $m->id); ?>>Ubah</a>
                <a href=<?= base_url('mahasiswa/hapus/' . $m->id); ?>" onclick="return confirm ('Anda Yakin?')>Hapus</a>
            </td>
        </tr>
    <?php $i++; ?>
<?php endforeach; ?>
```

2. Jalankan pada browser

A screenshot of a web browser window. The address bar shows 'localhost/codeigniter/mahasiswa'. The page title is 'Data Mahasiswa'. Below the title, there is a link '[Tambah](#)'. A table displays student data with columns '#', 'Nama', 'NIM', and 'Aksi'. Each row has two buttons: 'Ubah' and 'Hapus'.

## Data Mahasiswa

[Tambah](#)

#	Nama	NIM	Aksi
1	Luffy	2020054001	<a href="#">Ubah</a> <a href="#">Hapus</a>
2	Zoro	2020054002	<a href="#">Ubah</a> <a href="#">Hapus</a>

### 3. Pada controller mahasiswa buat method ubah

The screenshot shows a code editor with the file 'Mahasiswa.php' open. The code implements a 'ubah' method that performs validation and updates the database if validation passes.

```
// $id untuk menampung id yang dikirimkan dari index
public function ubah($id)
{
    $this->form_validation->set_rules('nama', 'Nama', 'required');
    $this->form_validation->set_rules('nim', 'Nim', 'required');

    if ($this->form_validation->run() == FALSE) {
        // kemudian $id tersebut dikirimkan ke mahasiswa model method get row
        $data['mahasiswa'] = $this->Mahasiswa_model->getRow($id);
        $this->load->view('mahasiswa/ubah', $data);
    } else {
        // jika validation bernilai true jalankan method update
        $this->Mahasiswa_model->update();
        $this->session->set_flashdata('berhasil', 'Data Berhasil Diubah');
        redirect('mahasiswa');
    }
}
```

### 4. Kita buat method getRow pada model mahasiswa

The screenshot shows a code editor with the file 'Mahasiswa\_model.php' open. It contains a 'getRow' method that retrieves a specific row from the 'mahasiswa' table based on the provided 'id'.

```
public function getRow($id)
{
    $where = ['id' => $id];
    // ambil data pada tabel mahasiswa dimana id = id yang dikirimkan
    return $this->db->get_where('mahasiswa', $where)->row();
}
```

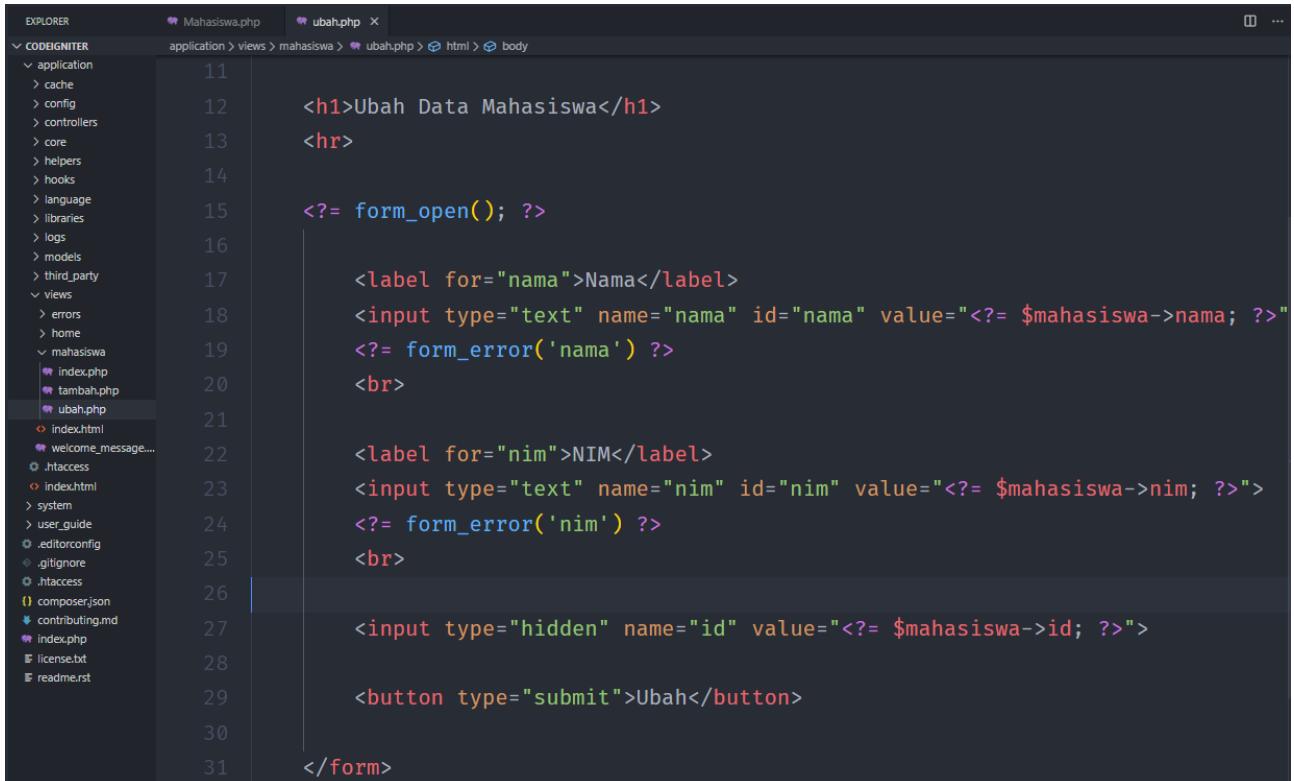
### 5. Kita buat method update pada model mahasiswa

The screenshot shows a code editor with the file 'Mahasiswa\_model.php' open. It contains an 'update' method that performs an update query on the 'mahasiswa' table using the provided data and id.

```
public function update()
{
    $data = [
        'nama' => $this->input->post('nama'),
        'nim' => $this->input->post('nim')
    ];
    $id = $this->input->post('id');

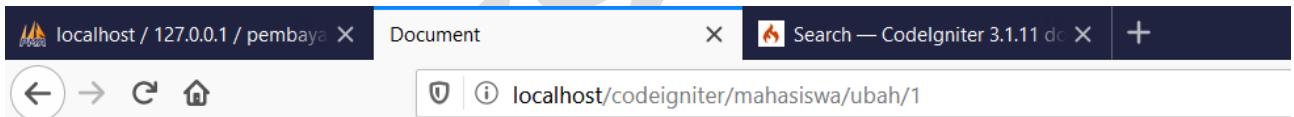
    // update pada tabel mahasiswa set datanya = $data dimana id = $id
    $this->db->where('id', $id);
    $this->db->update('mahasiswa', $data);
}
```

## 6. Buat view ubah pada folder mahasiswa



```
EXPLORER Mahasiswa.php ubah.php
CODEIGNITER application > views > mahasiswa > ubah.php > html > body
11 <h1>Ubah Data Mahasiswa</h1>
12 <hr>
13
14
15 <?= form_open(); ?>
16
17 <label for="nama">Nama</label>
18 <input type="text" name="nama" id="nama" value="<?= $mahasiswa->nama; ?>">
19 <?= form_error('nama') ?>
20 <br>
21
22 <label for="nim">NIM</label>
23 <input type="text" name="nim" id="nim" value="<?= $mahasiswa->nim; ?>">
24 <?= form_error('nim') ?>
25 <br>
26
27 <input type="hidden" name="id" value="<?= $mahasiswa->id; ?>">
28
29
30
31 </form>
```

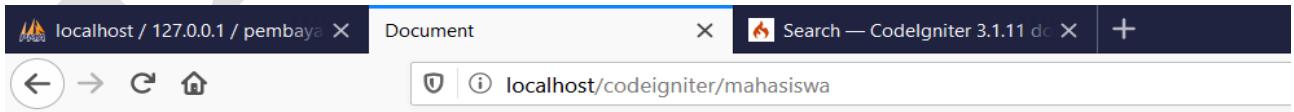
## 7. Test pada browser



## Ubah Data Mahasiswa

Nama

NIM



## Data Mahasiswa

Data Berhasil Diubah [Tambah](#)

#	Nama	NIM	Aksi
1	Monkey D luffy	2020054001	<a href="#">Ubah</a> <a href="#">Hapus</a>
2	Zoro	2020054002	<a href="#">Ubah</a> <a href="#">Hapus</a>

## DELETE

1. Tambahkan method hapus pada controller home

The screenshot shows the 'Mahasiswa.php' controller file in the CodeIgniter application's controllers directory. The code defines a 'hapus' method that calls the 'delete' method from the 'Mahasiswa\_model' model. It also sets a success message in the session and redirects to the 'mahasiswa' page.

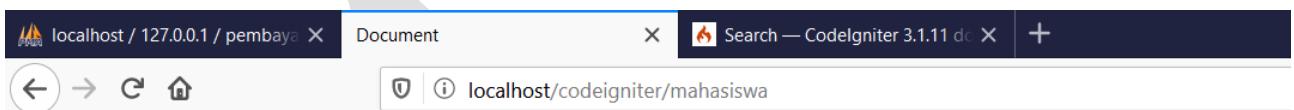
```
public function hapus($id)
{
    $this->Mahasiswa_model->delete($id);
    if ($this->db->affected_rows() > 0) {
        $this->session->set_flashdata('berhasil', 'Data Berhasil Dihapus');
        redirect('mahasiswa');
    }
}
```

2. Buat method delete pada Mahasiswa model

The screenshot shows the 'Mahasiswa\_model.php' model file in the CodeIgniter application's models directory. The code defines a 'delete' method that uses the database to find a record by ID and then deletes it.

```
public function delete($id)
{
    // delete pada tabel mahasiswa dimana id = $id
    $this->db->where('id', $id);
    $this->db->delete('mahasiswa');
}
```

3. Lakukan pengujian di browser



## Data Mahasiswa

Data Berhasil Dihapus [Tambah](#)

#	Nama	NIM	Aksi
1	Monkey D luffy	2020054001	<a href="#">Ubah</a> <a href="#">Hapus</a>
2	Zoro	2020054002	<a href="#">Ubah</a> <a href="#">Hapus</a>
3	sanji	2020054003	<a href="#">Ubah</a> <a href="#">Hapus</a>

## UPLOAD

1. Tambah field baru pada table mahasiswa , buka table mahasiswa → structure → go

The screenshot shows the MySQL Workbench interface with the 'Structure' tab selected for the 'mahasiswa' table. The table structure is as follows:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)	utf8mb4_general_ci		No	None		AUTO_INCREMENT	Change  Drop  More
2	nama	varchar(128)	utf8mb4_general_ci		No	None			Change  Drop  More
3	nim	int(11)	utf8mb4_general_ci		No	None			Change  Drop  More

At the bottom, there is a 'Go' button and a dropdown menu for adding columns.

2. Berikut fieldnya, pilih save

The screenshot shows the MySQL Workbench interface with the 'Structure' tab selected for the 'mahasiswa' table. A new column 'foto' is being added with the following properties:

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I	Comments
foto	VARCHAR	255	None						

At the bottom right, there are 'Preview SQL' and 'Save' buttons.

3. Structure

The screenshot shows the MySQL Workbench interface with the 'Structure' tab selected for the 'mahasiswa' table. The table now includes the 'foto' column:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)	utf8mb4_general_ci		No	None		AUTO_INCREMENT	Change  Drop  More
2	nama	varchar(128)	utf8mb4_general_ci		No	None			Change  Drop  More
3	nim	int(11)	utf8mb4_general_ci		No	None			Change  Drop  More
4	foto	varchar(255)	utf8mb4_general_ci		No	None			Change  Drop  More

4. Pada form tambah mahaasiswa, ubah form\_open menjadi form\_open\_multipart agar form bisa menangani file upload. Dan Tambahkan input tipe file untuk field foto

```

<?=> form_open_multipart(); ?>

    <label for="nama">Nama</label>
    <input type="text" name="nama" id="nama">
    <!-- error akan dicetak ketika validasinya tidak sesuai rule -->
    <?=> form_error('nama') ?>
    <br>

    <label for="nim">NIM</label>
    <input type="text" name="nim" id="nim">
    <?=> form_error('nim') ?>
    <br>

    <label for="foto">Foto</label>
    <input type="file" name="foto" id="foto">
    <br>

    <button type="submit">Tambah</button>

</form>

```

## 5. Tambahkan field foto pada model mahasiswa method insert

```

EXPLORER tambah.php Mahasiswa_model.php X
CODEIGNITER application > models > Mahasiswa_model.php > PHP Inteliphense > Mahasiswa_model > insert
16 public function insert()
17 {
18     // $url => ketika terjadi error pada saat upload, halaman akan di redirect ke alamat ini
19     // nantinya halaman ubah akan mempunyai url sendiri
20     $url = 'mahasiswa/tambah';
21     // tampung hasil input ke dalam sebuah variabel data
22     $data = [
23         'nama' => $this->input->post('nama'),
24         'nim' => $this->input->post('nim'),
25         // $this->upload($url) akan mengembalikan nama file hasil upload
26         'foto' => $this->upload($url)
27     ];
28     // insert (nama tabel, datanya apa)
29     $this->db->insert('mahasiswa', $data);
30 }

```

## 6. Buat method upload pada model



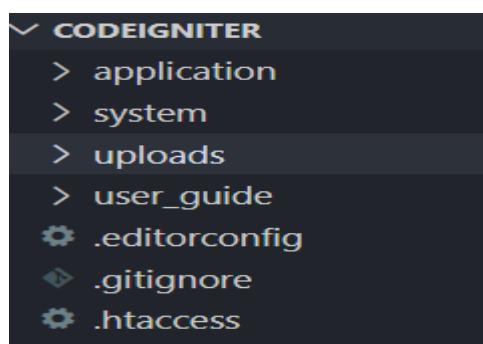
```
EXPLORER tambah.php Mahasiswa_model.php X
CODEIGNITER application > models > Mahasiswa_model.php > PHP Intelephense > Mahasiswa_model > upload
< application
  > cache
  > config
  > controllers
  > core
  > helpers
  > hooks
  > language
  > libraries
  > logs
  > models
    > index.html
    < Mahasiswa_model_
      > third_party
      > views
        < .htaccess
        < index.html
      > system
      > uploads
      > user_guide
        < .editorconfig
        < .gitignore
        < .htaccess
      < composer.json
      < contributing.md
      < index.php
    < license.txt
    < readme.rst

public function upload($url)
{
    // folder dimana file akan disimpan
    $config['upload_path'] = './uploads/';
    // type file yang diizinkan
    $config['allowed_types'] = 'jpeg|jpg|png';
    // maximum size dalam kb
    $config['max_size'] = 200;
    // mengganti nama file (nama file tidak boleh sama)
    $config['encrypt_name'] = TRUE;

    // memanggil settingan class upload
    $this->load->library('upload', $config);

    // jika uploadnya error
    if (!$this->upload->do_upload('foto')) {
        // tampung pesan errornya dalam variabel $error
        $error = array('error' => $this->upload->display_errors());
        // buat flasdata untuk pesan errornya
        $this->session->set_flashdata('error', $error['error']);
        // mengalihkan ke halaman ==> tergantung variabel url yang dikirimkan
        redirect($url);
    } else {
        // jika tidak ada error upload filenya
        $data = $this->upload->data();
        // kembalikan nama file ke method yang memanggilnya
        // nantinya nama file akan masuk ke database
        // filenya akan masuk ke dalam folder sesuai dengan upload_path
        return $data['file_name'];
    }
}
```

## 7. Buat folder uploads pada root folder



8. Tambahkan flashdata untuk mencetak notifikasi error, pada views/mahasiswa/tambah.php

```
EXPLORER          tambah.php ×
CODEIGNITER
  application
    > cache
    > config
    > controllers
    > core
    > helpers
    > hooks
    > language
    > libraries
    > logs
    > models
    > third_party
  views
    > errors
    > home
      mahasiswa
        index.php
        tambah.php
        ubah.php

11
12      <h1>Tambah Data Mahasiswa</h1>
13      <hr>
14
15      <?= $this->session->flashdata('error') ? $this->session->flashdata('error') : ''; ?>
16
17      <?= form_open_multipart(); ?>
18
19      <label for="nama">Nama</label>
20      <input type="text" name="nama" id="nama">
```

9. Lakukan pengetesan di browser

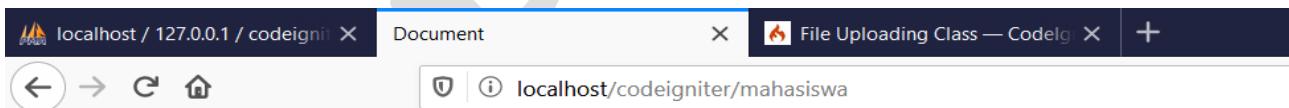
- a. Jika filenya tidak sesuai dengan tipe yang diijinkan

The filetype you are attempting to upload is not allowed.

- b. Jika type file sesuai tapi sizenya melebihi size yang diizinkan

The file you are attempting to upload is larger than the permitted size.

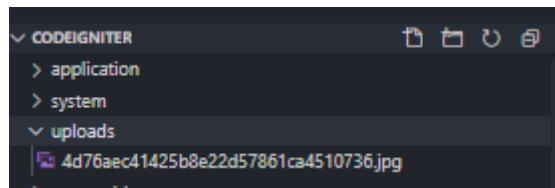
- c. Jika semuanya sesuai



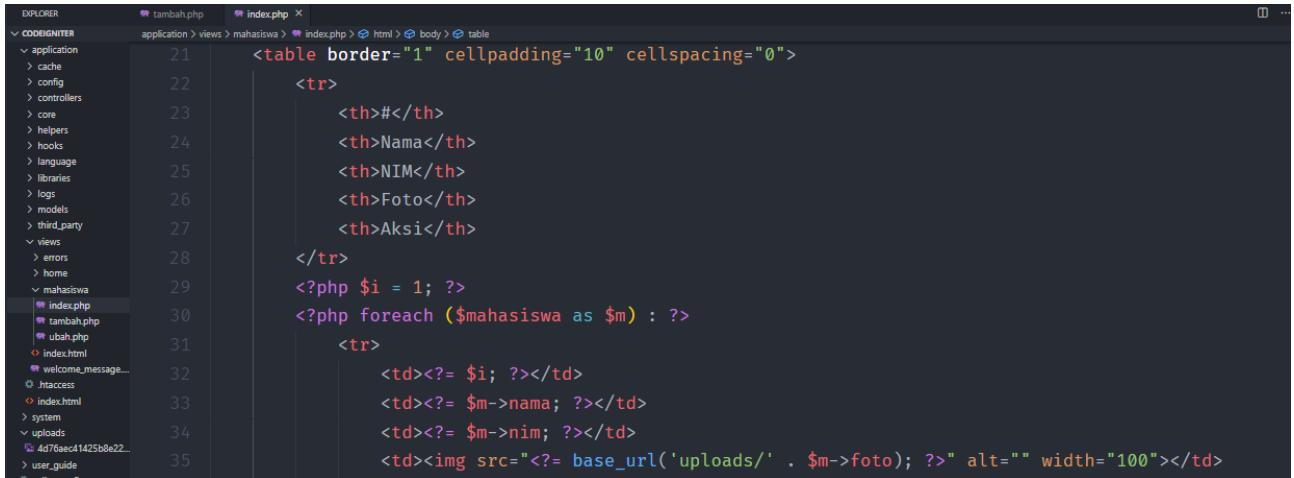
## Data Mahasiswa

Data Berhasil Ditambahkan [Tambah](#)

10. Perhatikan nama filenya sudah diubah



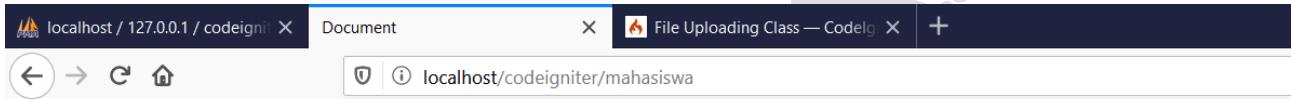
## 11. Tampilkan gambarnya pada tabel



The screenshot shows a code editor with a file named 'index.php' open. The code is a PHP script that generates an HTML table. It includes columns for '#', 'Nama', 'NIM', 'Foto', and 'Aksi'. The 'Foto' column contains an image tag pointing to a file in the 'uploads' directory. The code uses a foreach loop to iterate through an array of student data (\$mahasiswa).

```
<table border="1" cellpadding="10" cellspacing="0">
<tr>
<th>#</th>
<th>Nama</th>
<th>NIM</th>
<th>Foto</th>
<th>Aksi</th>
</tr>
<?php $i = 1; ?>
<?php foreach ($mahasiswa as $m) : ?>
<tr>
<td><?= $i; ?></td>
<td><?= $m->nama; ?></td>
<td><?= $m->nim; ?></td>
<td></td>
<td><a href="#">Ubah</a> <a href="#">Hapus</a></td>
</tr>
<?php endforeach; ?>
<tr>
<th colspan="5" style="text-align: center;">Total Data Mahasiswa: <?= count($mahasiswa); ?>
</tr>
</table>
```

## 12. Test browser



## Data Mahasiswa

[Tambah](#)

#	Nama	NIM	Foto	Aksi
1	Monkey D luffy	2020054001		<a href="#">Ubah</a> <a href="#">Hapus</a>
2	Zoro	2020054002		<a href="#">Ubah</a> <a href="#">Hapus</a>
3	sanji	2020054003		<a href="#">Ubah</a> <a href="#">Hapus</a>
4	Usop	2020054004		<a href="#">Ubah</a> <a href="#">Hapus</a>

## EDIT FILE UPLOAD

### 1. Views/mahasiswa/ubah.php

```
EXPLORER ubah.php X index.php
CODEIGNITER application > views > mahasiswa > ubah.php > html > body

application
> cache
> config
> controllers
> core
> helpers
> hooks
> language
> libraries
> logs
> models
> third_party
views
> errors
> home
mahasiswa
index.php
tambah.php
ubah.php
index.html
welcome_message...
htaccess
index.html
system
uploads
4d76aec41425b8e22...
user_guide
.editorconfig
.gitignore
.htaccess
composer.json
contributing.md
index.php
license.txt
readme.rst

12 <h1>Ubah Data Mahasiswa</h1>
13 <hr>
14
15 <?= $this->session->flashdata('error') ? $this->session->flashdata('error') : ''; ?>
16
17 <?= form_open_multipart(); ?>
18
19     <label for="nama">Nama</label>
20     <input type="text" name="nama" id="nama" value="<?= $mahasiswa->nama; ?>">
21     <?= form_error('nama') ?>
22     <br>
23
24     <label for="nim">NIM</label>
25     <input type="text" name="nim" id="nim" value="<?= $mahasiswa->nim; ?>">
26     <?= form_error('nim') ?>
27     <br>
28
29     <label for="nim">Foto Lama</label><br>
30     
31     <br>
32
33     <label for="foto">Foto</label>
34     <input type="file" name="foto" id="foto">
35     <br>
36
37     <input type="hidden" name="foto_lama" value="<?= $mahasiswa->foto; ?>">
38     <input type="hidden" name="id" value="<?= $mahasiswa->id; ?>">
39
40     <button type="submit">Ubah</button>
41
42 </form>
```

2. Kemudian pada method update\_data buat kondisi jika ada gambar baru jalankan method upload image, jika tidak gambar akan diambil dari gambar lama



```

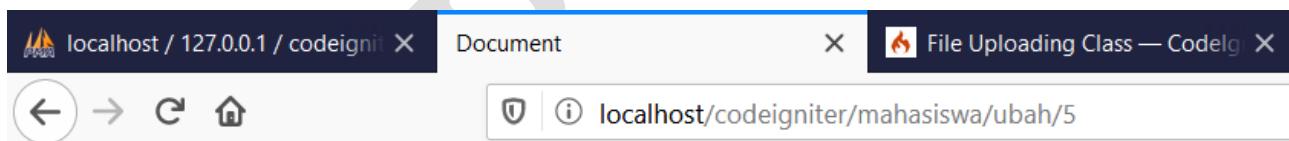
EXPLORER Mahasiswa_model.php
application > models > Mahasiswa_model.php > PHP Intelephense > Mahasiswa_model > delete

public function update()
{
    $url = 'mahasiswa/ubah';
    $fotoLama = $this->input->post('foto_lama');
    $data = [
        'nama' => $this->input->post('nama'),
        'nim' => $this->input->post('nim'),
        /*
            jika input tipe file yang namanya foto tidak kosong (ada foto
            yang diupload) maka jalankan method upload, sebaliknya foto
            diisi dengan foto lama
        */
        'foto' => !empty($_FILES["foto"]["name"]) ? $this->upload($url) :
        $fotoLama
    ];
    $id = $this->input->post('id');

    // update pada tabel mahasiswa set datanya = $data dimana id = $id
    $this->db->where('id', $id);
    $this->db->update('mahasiswa', $data);
}

```

3. Jalankan di browser



## Ubah Data Mahasiswa

Nama   
 NIM

Foto Lama

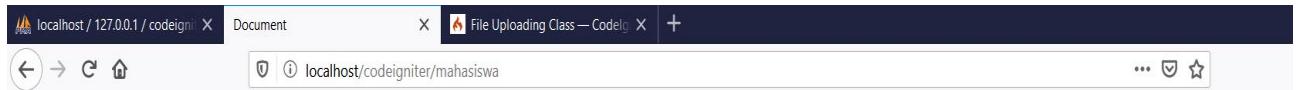


Foto  No file selected.

4. Lakukan pengujian

- a. Jika tidak ada file yang diupload
- b. Jika foto baru diupload

5. Lakukan ubah data pada semua data yang fotonya tidak ada



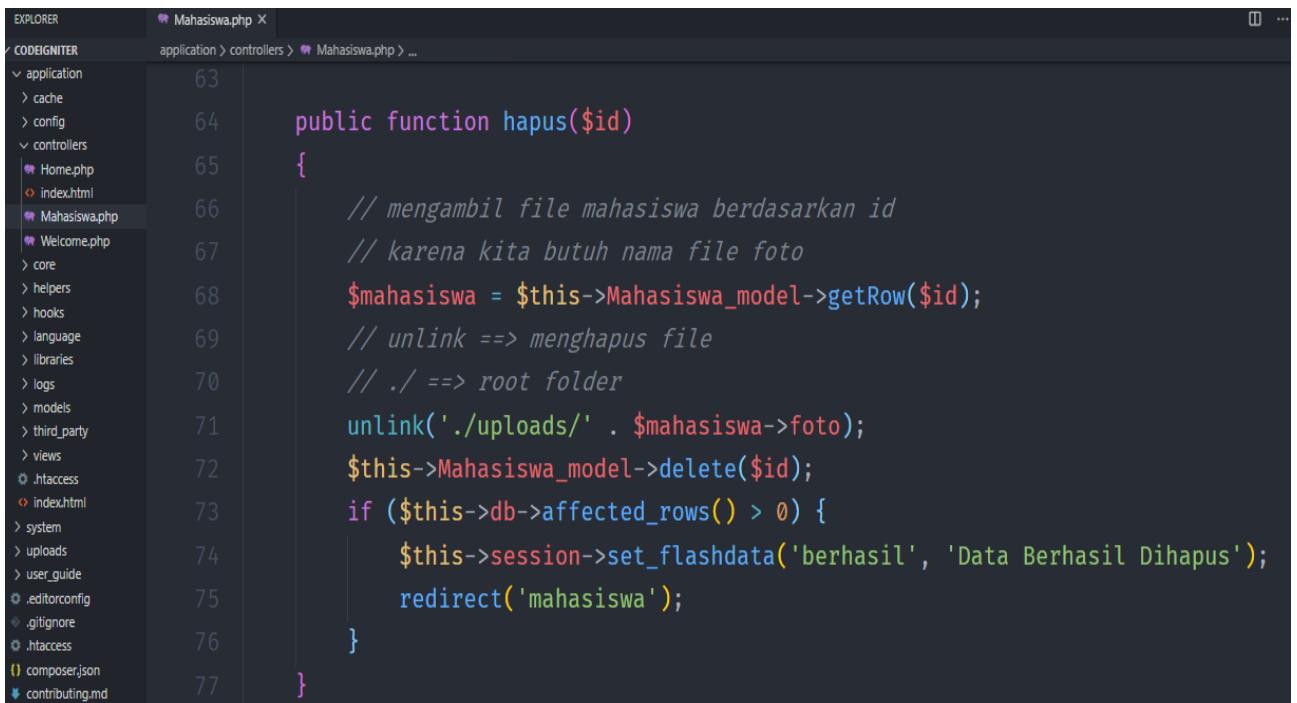
## Data Mahasiswa

Data Berhasil Diubah [Tambah](#)

#	Nama	NIM	Foto	Aksi
1	Monkey D luffy	2020054001		<a href="#">Ubah</a> <a href="#">Hapus</a>
2	Zoro	2020054002		<a href="#">Ubah</a> <a href="#">Hapus</a>
3	sanji	2020054003		<a href="#">Ubah</a> <a href="#">Hapus</a>
4	Usop	2020054004		<a href="#">Ubah</a> <a href="#">Hapus</a>

## DELETE FILE

1. Untuk script yang sekarang jika kita melakukan hapus data, maka datanya akan terhapus pada database tapi file fotonya tidak terhapus pada folder
2. Kita harus unlink foto tersebut pada method delete controller mahasiswa



```
EXPLORER CODEIGNITER Mahasiswa.php X application > controllers > Mahasiswa.php > ...
application
> cache
> config
controllers
  < Home.php
  < index.html
  < Mahasiswa.php
  < Welcome.php
> core
> helpers
> hooks
> language
> libraries
> logs
> models
> third_party
> views
  < .htaccess
  < index.html
system
> uploads
> user_guide
.editorconfig
.gitignore
.htaccess
composer.json
contributing.md

public function hapus($id)
{
    // mengambil file mahasiswa berdasarkan id
    // karena kita butuh nama file foto
    $mahasiswa = $this->Mahasiswa_model->getRow($id);
    // unlink ==> menghapus file
    // ./ ==> root folder
    unlink('./uploads/' . $mahasiswa->foto);
    $this->Mahasiswa_model->delete($id);
    if ($this->db->affected_rows() > 0) {
        $this->session->set_flashdata('berhasil', 'Data Berhasil Dihapus');
        redirect('mahasiswa');
    }
}
```

3. Lakukan pengujian di browser, hapus data dan cek pada folder uploads

#	Nama	NIM	Foto	Aksi
1	Monkey D luffy	2020054001		<a href="#">Ubah</a> <a href="#">Hapus</a>
2	Zoro	2020054002		<a href="#">Ubah</a> <a href="#">Hapus</a>
3	sanji	2020054003		<a href="#">Ubah</a> <a href="#">Hapus</a>

