
CV. MADURA
TECHNOVATION

MODUL

FRAMEWORK CODEIGNITER



Oleh :
Lembaga Kursus dan
Pelatihan (LKP)
Madura Technovation

BAB I

Perkenalan CodeIgniter 3

1.1 Penjelasan Framework

Apa itu Framework?

Framework atau dalam bahasa indonesia dapat diartikan sebagai "kerangka kerja" merupakan kumpulan dari fungsi-fungsi/prosedur-prosedur dan class-class untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang programer, tanpa harus membuat fungsi atau class dari awal.

Alasan mengapa menggunakan Framework

- Mempercepat dan mempermudah pembangunan sebuah aplikasi web.
- Relatif memudahkan dalam proses maintenance karena sudah ada pola tertentu dalam sebuah framework (dengan syarat programmer mengikuti pola standar yang ada)
- Umumnya framework menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal (misalnya validasi, ORM, pagination, multiple database, scaffolding, pengaturan session, error handling, dll)
- Lebih bebas dalam pengembangan jika dibandingkan CMS

1.2 Perkenalan CodeIgniter

Apa itu CodeIgniter?



CodeIgniter adalah sebuah web application network yang bersifat open source yang digunakan untuk membangun aplikasi php dinamis.

CodeIgniter menjadi sebuah framework PHP dengan model MVC (Model, View, Controller) untuk membangun website dinamis dengan menggunakan PHP yang dapat mempercepat pengembang untuk membuat sebuah aplikasi web. Selain ringan dan cepat, CodeIgniter juga memiliki dokumentasi yang super lengkap disertai dengan contoh implementasi kodenya. Dokumentasi yang lengkap inilah yang menjadi salah satu alasan kuat mengapa banyak orang memilih CodeIgniter sebagai framework pilihannya. Karena kelebihan-kelebihan yang dimiliki oleh CodeIgniter, pembuat PHP Rasmus Lerdorf memuji CodeIgniter di frOSCon (Agustus 2008) dengan mengatakan bahwa dia menyukai CodeIgniter karena "*it is faster, lighter and the least like a framework.*"

CodeIgniter pertama kali dikembangkan pada tahun 2006 oleh Rick Ellis. Dengan logo api yang menyala, CodeIgniter dengan cepat “membakar” semangat para web developer untuk mengembangkan web dinamis dengan cepat dan mudah menggunakan framework PHP yang satu ini.

1.3 Fungsi CodeIgniter

1. Mempercepat dan mempermudah kita dalam pembuatan website.
2. Menghasilkan struktur pemrograman yang sangat rapi, baik dari segi kode maupun struktur file phpnya.
3. Memberikan standar coding sehingga memudahkan kita atau orang lain untuk mempelajari kembali system aplikasi yang dibangun.

1.4 Kelebihan CodeIgniter

1. Berukuran sangat kecil. File download nya hanya sekitar 2MB, itupun sudah included dokumentasinya yang sangat lengkap.
2. Dokumentasi yang bagus. Saat anda mendownloadnya, telah disertakan dengan dokumentasi yang berisi pengantar, tutorial, bagaimana panduan penggunaan, serta referensi dokumentasi untuk komponen-komponennya.
3. Kompatibilitas dengan Hosting. CodeIgniter mampu berjalan dengan baik pada hampir semua platform hosting. CodeIgniter juga mendukung database-database paling umum, termasuk MySQL.
4. Tidak ada aturan coding yang ketat. Terserah anda jika anda hanya ingin menggunakan Controller, tanpa View, atau tidak menggunakan Model, atau tidak salah satu keduanya. Namun dengan menggunakan ketiga komponennya adalah pilihan lebih bijak.
5. Kinerja yang baik. Codeigniter sangat cepat bahkan mungkin bisa dibilang merupakan framework yang paling cepat yang ada saat ini.
6. Sangat mudah diintegrasikan. CodeIgniter sangat mengerti tentang pengembangan berbagai library saat ini. Karenanya CodeIgniter memberikan kemudahan untuk diintegrasikan dengan library-library yang tersedia saat ini.
7. Sedikit Konfigurasi. Konfigurasi CodeIgniter terletak di folder application/config. CodeIgniter tidak membutuhkan konfigurasi yang rumit, bahkan untuk mencoba menjalankannya, tanpa melakukan konfigurasi sedikitpun ia sudah bisa berjalan.
8. Mudah dipelajari. Disamping dokumentasi yang lengkap, ia juga memiliki berbagai forum diskusi.

1.5 Kekurangan CodeIgniter

1. CodeIgniter tidak ditujukan untuk pembuatan web dengan skala besar.
2. Library yang sangat terbatas. Hal ini dikarenakan sangat sulit mencari plugin tambahan yang terverifikasi secara resmi, karena pada situsnya CodeIgniter tidak menyediakan plugin-plugin tambahan untuk mendukung pengembangan aplikasi dengan CI.
3. Belum adanya editor khusus CodeIgniter, sehingga dalam melakukan create project dan modul-modulnya harus berpindah-pindah folder.

1.6 CodeIgniter 3

Pada bulan Juli 2013, EllisLab mengumumkan bahwa mereka mencari pemilik baru untuk CodeIgniter karena internal mereka sendiri tidak memiliki cukup fokus untuk terus mengembangkan CodeIgniter. Akhirnya pada bulan Oktober 2014, kepemilikan CodeIgniter berpindah tangan kepada British Columbia Institute of Technology, salah satu sekolah tinggi teknologi di Kanada.

Setelah hampir lima bulan lamanya sejak peralihan kepemilikan, BCIT akhirnya merilis CodeIgniter 3.0. Dan berikut adalah perubahan codeigniter 2 menjadi codeigniter 3 :

1. Codeigniter 3 memerlukan PHP versi 5.1.6 atau di atasnya
2. Penamaan model, controller Codeigniter 3 harus di awali huruf besar
3. Driver databasenya kini memiliki refactoring yang lebih luas. Sekarang default database driver nya menggunakan mysqli, tidak lagi menggunakan mysql
4. Penambahan user agent Windows 7, Windows 8, Windows 8.1, Android, Blackberry, iOS dan PlayStation 3
5. Update perbaikan di mimes.php
6. Update penulisan class dengan PHP 5 style
7. Pindah path halaman error di application/view/errors

8. Pindah Log Class di application/core
9. Update perbaikan di beberapa Library dan Helper
10. Perbaikan file dokumentas

1.7 Pengertian MVC

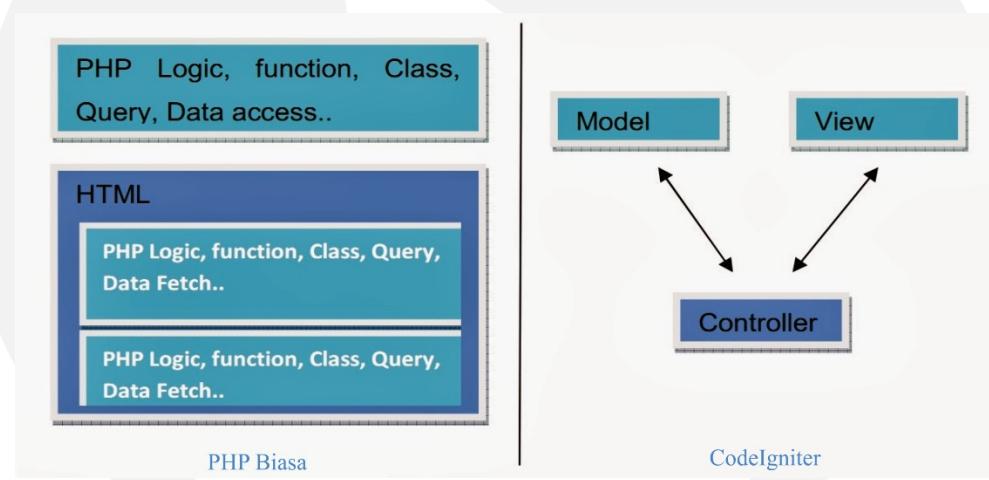
Apa itu MVC?

MVC adalah konsep dasar yang harus diketahui sebelum mengenal CodeIgniter. MVC (Model View Controller) merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi web, berawal pada bahasa pemrograman Small Talk, yang memisahkan bisnis logic (alur piker), data logic (penyimpanan data) dan presentation logic (antarmuka aplikasi) atau secara sederhana adalah memisahkan antara desain, data dan proses. Ada 3 komponen yang membangun suatu MVC yaitu :

1. Model, biasanya 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 didalam model akan berisi class dan fungsi untuk mengambil, melakukan update dan menghapus data website. Sebuah aplikasi web biasanya menggunakan basis data dalam menyimpan data, maka pada bagian Model biasanya akan berhubungan dengan perintah-perintah query SQL.
2. View, merupakan bagian yang menangani presentation logic. Pada suatu aplikasi web bagian ini biasanya berupa file template HTML, yang diatur oleh controller. View berfungsi untuk menerima dan merepresentasikan data hasil dari model dan controller kepada user. View tidak memiliki akses langsung terhadap bagian model.

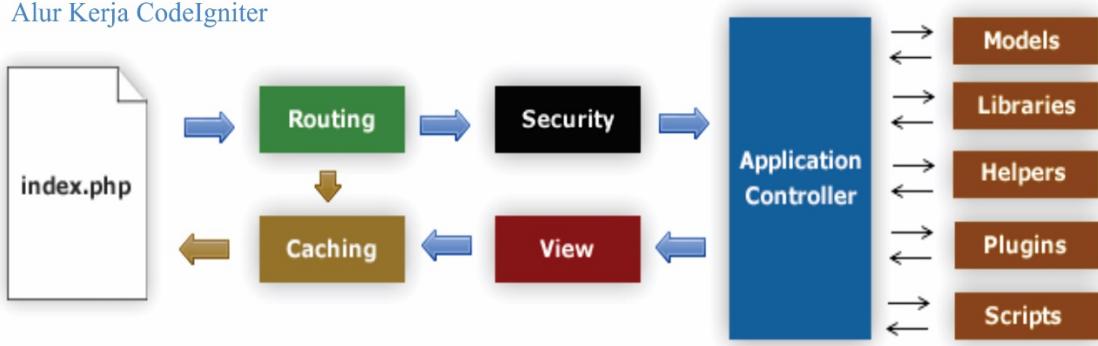
3. Controller, merupakan bagian yang mengatur hubungan antara bagian model dan bagian view. Pada controller terdapat class-class dan fungsi-fungsi yang memproses permintaan dari View ke dalam struktur data di dalam model. Controller juga tidak boleh berisi kode untuk mengakses basis data Karena tugas megakses data telah diserahkan kepada model. Tugas controller adalah menyediakan berbagai variable 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.

Perbandingan PHP Biasa dengan CodeIgniter



Alur Kerja Framework CodeIgniter

Alur Kerja CodeIgniter



1. Index.php

Index.php disini berfungsi sebagai file pertama dalam program yang akan dibaca oleh program.

2. The Router

Router akan memeriksa HTTP request untuk menentukan hal apa yang harus dilakukan oleh program.

3. Cache File

Apabila dalam program sudah terdapat "cache file" maka file tersebut akan langsung dikirim ke browser. File cache inilah yang dapat membuat sebuah website dapat dibuka dengan lebih cepat. Cache file dapat melewati proses yang sebenarnya harus dilakukan oleh program codeigniter.

4. Security

Sebelum file controller di load keseluruhan, HTTP request dan data yang disubmit oleh user akan disaring terlebih dahulu melalui fasilitas security yang dimiliki oleh codeigniter.

5. Controller

Controller akan membuka file model, core libraries, helper dan semua resources yang dibutuhkan dalam program tersebut.

6. View

Hal yang terakhir akan dilakukan adalah membaca semua program yang ada dalam view file dan mengirimkannya ke browser supaya dapat dilihat. Apabila file view sudah ada yang di “cache” maka file view baru yang belum ter-cache akan mengupdate file view yang sudah ada.

Contoh File untuk Model, View dan Controller

1. Model / buku_model.php

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

class Buku_model extends CI_Model {

    public function getBook()
    {
        return array(
            array (
                'judul' => 'Belajar Framework CI',
                'pengarang' => 'Budi Raharjo',
                'penerbit' => 'Informatika'
            ),
            array(
                'judul' => 'Belajar Photoshop',
                'pengarang' => 'Abdul Kadir',
                'penerbit' => 'Andi Offset'
            )
        );
    }
}
```

Contoh File Model / buku_model.php

2. View / buku_view.php

```

<!DOCTYPE html>
<html>

<head><title> Buku </title></head>
<body>

<table border="1">
    <tr>
        <th> No </th>
        <th> Judul </th>
        <th> Pengarang </th>
        <th> Penerbit </th>
    </tr>
    <?php
        $no = 1;
        for($i=0;$i < count($data_buku); $i++) {
            echo '<tr>';
            echo '<td>' . $no . '</td>';
            echo '<td>' . $data_buku[$i]['judul'] . '</td>';
            echo '<td>' . $data_buku[$i]['pengarang'] . '</td>';
            echo '<td>' . $data_buku[$i]['penerbit'] . '</td>';
            echo '</tr>';

            $no++;
        }
    ?>
</table>
</body>
</html>

```

Contoh File View / buku_view.php

3. Controller / buku.php

```

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

class Buku extends CI_Controller {

    public function __construct()
    {
        parent::__construct();
        //Do your magic here
        $this->load->model('buku_model');
    }

    public function index()
    {
        //get data buku
        $data['data_buku'] = $this->buku_model->getBook();

        //load view buku_view
        $this->load->view('buku_view', $data);
    }
}

```

Contoh File Controller / Buku.php

BAB II

PHP & Object Oriented Programming (OOP)

2.1 Sejarah PHP

Pada awalnya PHP merupakan kependekan dari *Personal Home Page* (Situs personal). PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP masih bernama *Form Interpreted* (FI), yang wujudnya berupa sekumpulan skrip yang digunakan untuk mengolah data formulir dari web.

Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP/FI. Dengan perilisan kode sumber ini menjadi sumber terbuka, maka banyak pemrogram yang tertarik untuk ikut mengembangkan PHP.

Pada November 1997, dirilis PHP/FI 2.0. Pada rilis ini, *interpreter* PHP sudah diimplementasikan dalam program C. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP/FI secara signifikan.

Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang interpreter PHP menjadi lebih bersih, lebih baik, dan lebih cepat. Kemudian pada Juni 1998, perusahaan tersebut merilis interpreter baru untuk PHP dan meresmikan rilis tersebut sebagai PHP 3.0 dan singkatan PHP diubah menjadi akronim berulang *PHP: Hypertext Preprocessing*.

Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi web kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi.

Pada Juni 2004, Zend merilis PHP 5.0. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi objek. Server web bawaan ditambahkan pada versi 5.4 untuk mempermudah pengembang menjalankan kode PHP tanpa menginstall software server.

Versi terbaru dan stabil dari bahasa pemrograman PHP saat ini adalah versi 7.0.16 dan 7.1.2 yang resmi dirilis pada tanggal 17 Februari 2017.

2.2 Pengertian PHP

Apa itu PHP?

PHP adalah singkatan dari "PHP: Hypertext Preprocessor", yaitu bahasa pemrograman disisi server yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah situs web dan bisa digunakan bersamaan dengan HTML.

Ketika anda mengakses sebuah URL, maka web browser akan melakukan request ke sebuah web server.

2.3 Pengertian OOP

Apa itu OOP?

OOP (Object Oriented Programming) adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, nah objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi. Saya ambil contoh Pesawat, Pesawat adalah sebuah objek. Pesawat itu sendiri terbentuk dari beberapa objek yang lebih kecil lagi seperti mesin, roda, baling-baling, kursi, dll. Pesawat sebagai objek yang terbentuk dari objek-objek yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada objek-objek yang lainnya. Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, objek-objek itu saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain.

Konsep OOP

1. Abstrak Class

- Kelas merupakan deskripsi abstrak informasi dan tingkah laku dari sekumpulan data.
- Kelas dapat diilustrasikan sebagai suatu cetak biru(blueprint) atau prototipe yang digunakan untuk menciptakan objek.
- Kelas merupakan tipe data bagi objek yang mengenkapsulasi data dan operasi pada data dalam suatu unit tunggal.
- Kelas mendefinisikan suatu struktur yang terdiri atas data kelas (data field), prosedur atau fungsi (method), dan sifat kelas (property).

2. Encapsulation

- Istilah enkapsulasi sebenarnya adalah kombinasi data dan fungsionalitas dalam sebuah unit tunggal sebagai bentuk untuk menyembunyikan detail informasi.
- Proses enkapsulasi memudahkan kita untuk menggunakan sebuah objek dari suatu kelas karena kita tidak perlu mengetahui segala hal secara rinci.
- Enkapsulasi menekankan pada antarmuka suatu kelas, atau dengan kata lain bagaimana menggunakan objek kelas tertentu.
- Contoh: kelas mobil menyediakan antarmuka fungsi untuk menjalankan mobil tersebut, tanpa kita perlu tahu komposisi bahan bakar, udara dan kalor yang diperlukan untuk proses tersebut.

3. Inheritance

- Kita dapat mendefinisikan suatu kelas baru dengan mewarisi sifat dari kelas lain yang sudah ada.
- Penurunan sifat ini bisa dilakukan secara bertingkattingkat, sehingga semakin ke bawah kelas tersebut menjadi semakin spesifik.
- Sub kelas memungkinkan kita untuk melakukan spesifikasi detail dan perilaku khusus dari kelas supernya.

- Dengan konsep pewarisan, seorang programmer dapat menggunakan kode yang telah ditulisnya pada kelas super berulang kali pada kelas-kelas turunannya tanpa harus menulis ulang semua kode-kode itu.
4. Polymorphism
- Polimorfisme merupakan kemampuan objek-objek yang berbeda kelas namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama.
 - Polimorfisme juga dapat dikatakan kemampuan sebuah objek untuk memutuskan method mana yang akan diterapkan padanya, tergantung letak objek tersebut pada jenjang pewarisan.
 - Method overriding.
 - Method name overloading.

BAB III

Installasi dan Konfigurasi CodeIgniter

3.1 Installasi CodeIgniter

Agar dapat menggunakan CodeIgniter, yang harus dilakukan adalah menginstall dan melakukan konfigurasi terhadap CodeIgniter terlebih dahulu. Installasi CodeIgniter sangatlah mudah. Hal-hal yang harus dipersiapkan dalam menginstall CodeIgniter adalah mempersiapkan web server.

Banyak sekali aplikasi web server yang beredar, salah satu web server yang sangat terkenal dan juga bersifat bebas adalah web server Apache, sebuah web server yang digunakan pada sebagian server yang ada di internet. Pada tutorial dibawah ini menggunakan XAMPP sebagai aplikasi web server. Selain web server siapkan juga Code Editor.

Langkah-langkah menginstall CodeIgniter :

1. Yang pertama adalah download terlebih dahulu CodeIgniter. CodeIgniter dapat di download di link <https://codeigniter.com/download>.

CodeIgniter 3.x

CodeIgniter 3.1.5 is the current version of the framework.

There have been a number of refinements since version 2.x, notably with the database, session handling and encryption. Development of this version is ongoing.

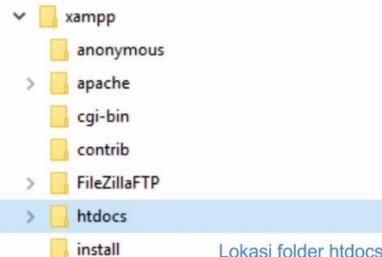
[View CodeIgniter 3 on Github](#)

[Download CodeIgniter 3](#)

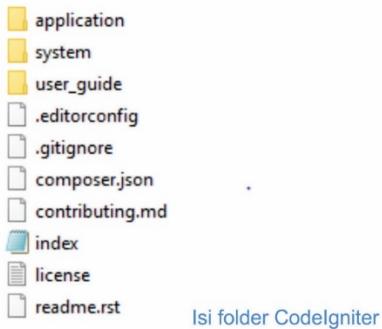
[Download System Message Translations](#)

Halaman download CodeIgniter 3.x

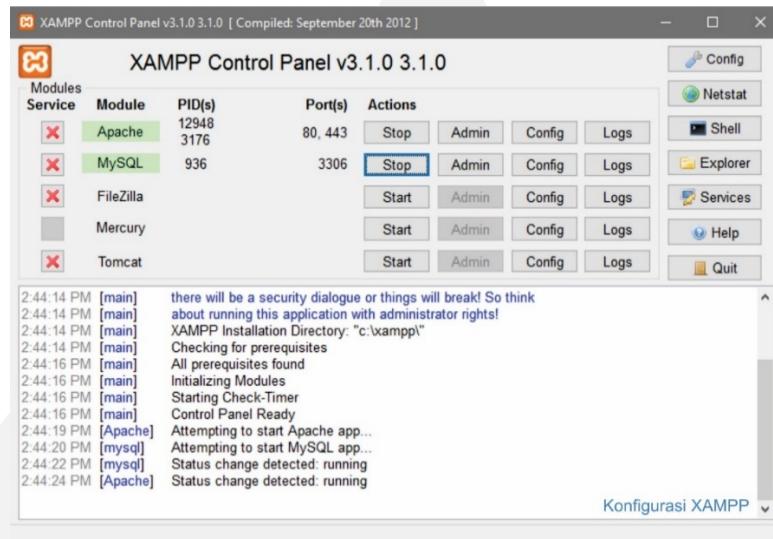
2. Ekstrak paketan CodeIgniter, lalu copy folder CodeIgniter ke htdocs di *C:\xampp\htdocs* (atau sesuai dimana XAMPP di install). Nama folder dapat diubah sesuai keinginan.



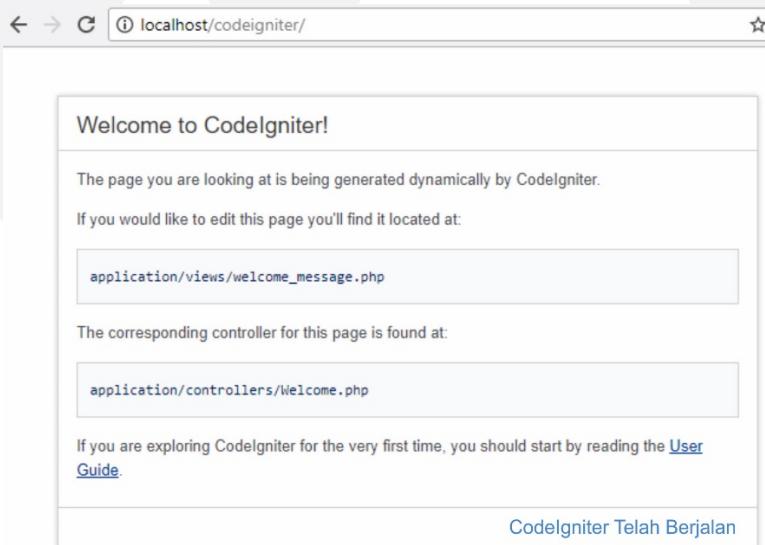
Isi dari paket CodeIgniter adalah :



3. Nyalakan aplikasi XAMPP seperti berikut (Apache dan MySQL) :



4. Setelah itu akses folder yang ada pada folder htdocs dan akan muncul seperti dibawah ini.
Dan CodeIgniter telah sukses berjalan di aplikasi Anda.



3.2 Konfigurasi CodeIgniter

Walaupun CodeIgniter dapat berjalan dengan konfigurasi default, tetapi untuk sebuah aplikasi yang nyata kita harus tetap melakukan konfigurasi, setidaknya pada bagian base_url dan router. Pengaturan base_url dan router sangat berguna ketika proses pengembangan aplikasi yang banyak menggunakan helper dan library.

File konfigurasi terletak dalam folder application/config. File yang terdapat pada direktori tersebut yang sering digunakan adalah file autoload.php, config.php, database.php dan routes.php.

1. File autoload.php. Konfigurasi pada file ini bertujuan untuk mementukan sumber daya apa yang akan diload secara otomatis.

```
$autoload['libraries'] = array();  
/*  
 | Auto-load Libraries  
 |-----  
 | These are the classes located in system/libraries/ or your  
 | application/libraries/ directory, with the addition of the  
 | 'database' library, which is somewhat of a special case.  
 |  
 | Prototype:  
 |  
 | $autoload['libraries'] = array('database', 'email', 'session');  
 |  
 | You can also supply an alternative library name to be assigned  
 | in the controller:  
 |  
 | $autoload['libraries'] = array('user_agent' => 'ua');  
 */  
File autoload.php bagian libraries
```

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

2. File config.php. Pada file konfigurasi config.php berisi konfigurasi secara umum mengenai CodeIgniter.

```
$config['base_url'] = '';
/*
----- Base Site URL -----
----- URL to your CodeIgniter root. Typically this will be your base URL,
WITH a trailing slash:
http://example.com/
WARNING: You MUST set this value!
If it is not set, then CodeIgniter will try guess the protocol and path
your installation, but due to security concerns the hostname will be set
to $_SERVER['SERVER_ADDR'] if available, or localhost otherwise.
The auto-detection mechanism exists only for convenience during
development and MUST NOT be used in production!
If you need to allow multiple domains, remember that this file is still
a PHP script and you can easily do that on your own.
*/
File config.php bagian base url
```

`$config['base_url']` – Konfigurasi ini berisi alamat url sebuah aplikasi yang dibuat. Pada tutorial sebelumnya instalasi CodeIgniter di folder “`C:xampp/htdocs/codeigniter`” maka untuk konfigurasi base url-nya adalah seperti berikut.

```
$config['base_url'] = 'localhost/codeigniter';
Contoh konfigurasi base url
```

3. File database.php. Disini akan mengisi konfigurasi untuk koneksi ke database. Pada tutorial disini menggunakan database mysql yang terpaket di dalam aplikasi XAMPP.

```
$db['default'] = array(
    'dsn'      => '',
    'hostname' => 'localhost',
    'username' => '',
    'password' => '',
    'database' => '',
    '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
);
File database.php
```

Hostname : menggunakan localhost karena di sini kita menggunakan database mysql dari aplikasi xampp yang terinstal di computer local.

Username : secara default username database mysql dari aplikasi xampp adalah root.

Password : secara default ini bisa dikosongi.

Database : ini berisikan nama database yang dibuat untuk aplikasi.

4. File routes.php. Konfigurasi routing digunakan untuk memetakan permintaan atau request kedalam controller didalam website yang dibuat. Misalnya kita membuka alamat <http://www.nama-website.com>, permintaan tersebut tidak menyertakan nama controller yang ingin dibuka tetapi kita bisa secara default mengarahkannya agar secara otomatis akan membuka controller sesuai yang definisikan.

Untuk melakukan konfigurasi routing buka file konfigurasi routes.php. Settingan utama yang ada adalah sebagai berikut :

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

File routes.php

Artinya secara default semua permintaan yang tidak menyertakan nama controllernya akan diarahkan untuk membuka controller “welcome”. Sehingga saat alamat <http://www.nama-website.com> dibuka secara otomatis akan membuka <http://www.nama-website.com/index.php/welcome>. Dan file tersebut dapat di edit sesuai keinginan.

BAB IV

Hello CodeIgniter

4.1 CodeIgniter Library dan Helper

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

- **Library**

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.

Pada umumnya ada banya library yang dapat digunakan pada CodeIgniter. Tetapi untuk tahap awal library diatas yang wajib diketahui sebab library tersebut pada umumnya sering digunakan.

Untuk menggunakan library yang ada pada folder system/libraries, ada dua cara yang dapat dilakukan, yaitu :

1. Mengatur pada file *system/config/autoload.php*.

Contoh :

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

Contoh konfigurasi library

2. Dengan melakukan loading terhadap library yang kita inginkan pada controller dimana library ini akan digunakan. Biasanya library ini di-load pada konstruktor dari controller yang bersangkutan. Berikut sintaknya:

```
$this->load->library('nama_library');
```

Helper

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 :

1. URL helper : membantu dalam pembuatan link.
2. Form helper : membantu untuk membuat element-element form.
3. Text helper : membantu untuk pekerjaan berformat text.
4. Cookie helper : membantu untuk penanganan cookies.
5. File helper : membantu untuk kerja dengan file.

4.2 Mempercantik URL CodeIgniter

Saat mengakses url Codeigniter, pasti dilihat ada index.php. Contohnya <http://localhost/project/index.php/home>. Sebenarnya index.php dapat dihilangkan dengan mudah sehingga aplikasi web yang dibuat tidak perlu menggunakan index.php pada urlnya.

Cara menghilangkan index.php pada CodeIgniter :

1. Buat file “.htaccess” di dalam folder project CodeIgniter dan ketikan script berikut:

```
<IfModule mod_rewrite.c>

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

</IfModule>
```

2. Edit file *config.php* yang berada di folder */application/config/* dan cari *index_page* dan hapus tulisan *index.php*, sehingga seperti berikut :

```
$config['index_page'] = 'index.php';
```

Edit file config.php

BAB V

Database

CodeIgniter mendukung banyak jenis database misalnya MySQL, PostGre SQL, Oracle dan lain-lain. Dukungan database dari CodeIgniter berupa penyediaan beberapa driver database yang sekaligus juga memiliki fungsi sekuriti, caching dan active record.

5.1 Connect ke Database

Agar dapat melakukan koneksi dengan database, yang harus dilakukan adalah konfigurasi database pada file application/config/database.php seperti berikut :

```
$db['default'] = array(
    'dsn'      => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'database_name',
    '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
);
```

Contoh konfigurasi database

Diatas menggunakan array multi dimensi Karena lebih simple dan dapat menyimpan lebih dari satu konfigurasi koneksi.

Cara untuk dapat mengkoneksikan database :

CodeIgniter memiliki sebuah file konfigurasi yang memungkinkan Anda menyimpan konfigurasi untuk melakukan koneksi ke database (username, password, nama database dan lain-lain). File konfigurasi terletak di *application/config/database.php*.

Untuk connect ke database ada beberapa cara yang dapat dilakukan diantaranya:

1. Menambahkan Database Library sebagai Autoload Library

Untuk connect ke database bisa dengan cara menambahkan database sebagai autoload library di file *application/config/autoload.php*.

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

2. Mengaktifkan Manual Dari Library Database

Jika hanya halaman website yang memerlukan koneksi database, maka untuk optimalisasi, lakukan koneksi ke database secara manual, dengan menambahkan baris kode di bawah ini pada tiap fungsi tempat yang membutuhkan koneksi database atau dalam konstruktor kelas untuk membuat database yang tersedia secara global di kelas.

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

Code diatas jika tidak berisikan informasi apapun pada parameter pertama maka akan menyambung pada group konfigurasi yang aktif. Untuk memilih kelompok tertentu dari file konfigurasi, dapat melakukan seperti hal berikut ini (berguna untuk aplikasi yang memiliki 2 database).

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

Group_name adalah nama grup konfigurasi dari file konfigurasi Anda. Contoh penggunaannya :

```

$db['default'] = array(
    'dsn'      => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'database_satu',
    '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
);

$db['database_dua'] = array(
    'dsn'      => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'database_dua',
    '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
);

```

Contoh konfigurasi 2 database

Cara memanggilnya pada file model di sisi construct adalah :

```
$this->db2 = $this->load->database('database_dua', TRUE);
```

Perbedaan koneksi ke database 1 dan database 2, terletak pada `$this->db` atau `$this->db2`.

3. Mengaktifkan Manual Dari Model

Cara yang ketiga adalah mengaktifkan database pada saat loading model. Caranya dengan mengeset TRUE pada parameter ketiga load model. Contoh :

```
$this->load->model('nama_model','TRUE');
```

5.2 CodeIgniter Model

Model pada CodeIgniter adalah sebuah kelas php yang berfungsi untuk menangani data. Kelas model di-extend ketika hendak menggunakan fitur database pada CodeIgniter saja. Semua file model harus diletakkan di dalam folder application/models. Untuk me-load model kita dapat menggunakan perintah berikut :

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

Jika sudah berhasil me-load sebuah model maka model tersebut akan menjadi sebuah property, dari property itulah akan dapat menggunakan semua fungsi yang ada di dalam file model.

5.3 Melakukan Query pada Database

Query dilakukan untuk dapat mengambil data pada database.

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

Query diatas belum menghasilkan data apapun. Keluarannya hanya berupa Object(true) atau false. Ketika keluarannya False berarti query yang dilakukan gagal. Tetapi jika true atau mengembalikan sebuah object maka query yang dilakukan telah berhasil.

Dari object tersebut (variable \$query, mengacu pada contoh diatas) dapat mengambil data yang diinginkan. Contohnya :

```
$query=$this->db->query('SELECT nama, judul, email FROM tabel');
```

```
foreach ($query->result() as row)
{
    echo $row->nama;
    echo $row->judul;
    echo $row->email;
}
```

BAB VI

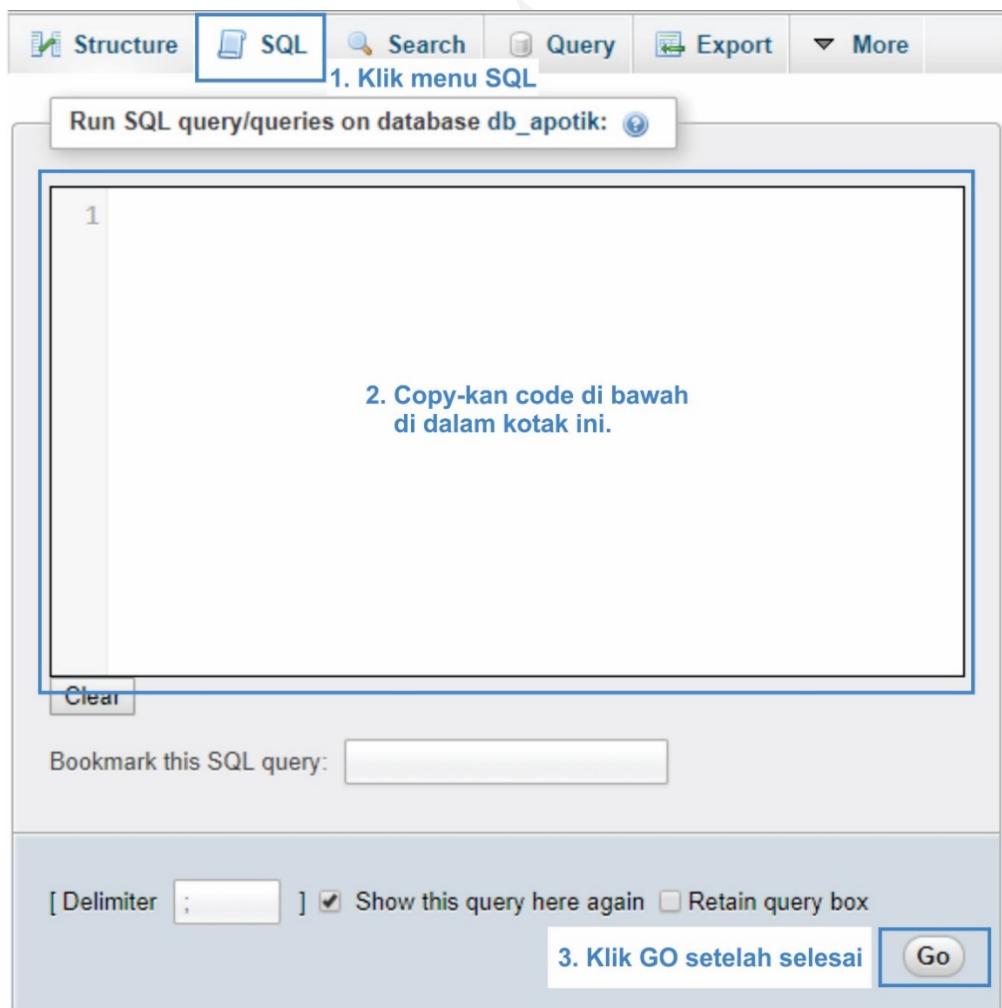
Contoh Program

Pada contoh program di bawah ini akan ditunjukkan bagaimana membuat program apotik disertakan pembuatan database, login, validation form dan Create, Read, Update, Delete (CRUD).

6.1 Persiapan Database

Langkah pertama adalah mendesain dan membuat database. Mengapa? Karena database akan sangat berhubungan dengan jalannya data program yang akan dibuat. Pada tutorial berikut ini akan digunakan aplikasi server XAMPP.

Buat database dengan nama db_apotik atau yang lain, lalu copy-kan code di bawah dengan intruksi seperti di bawah ini :



Tabel : ADMIN

```
create table ADMIN
(
    USERNAME          varchar(225) not null,
    PASSWORD          varchar(225),
    primary key (USERNAME)
);
```

Tabel : DETIL_TRANSAKSI

```
create table DETIL_TRANSAKSI
(
    KODE_DETIL        int not null,
    KODE_TRANSAKSI    int,
    KODE_OBAT         int not null,
    SUB_TOTAL         int,
    JUMLAH            int,
    primary key (KODE_DETIL)
);
```

Table: OBAT

```
create table OBAT
(
    KODE_OBAT          int not null,
    KODE_SUPPLIER       int not null,
    KODE_DETIL          int,
    NAMA_OBAT          varchar(225),
    PRODUSEN           varchar(225),
    HARGA               int,
    JML_STOK            int,
    FOTO                longblob,
    primary key (KODE_OBAT)
);
```

Tabel : SUPPLIER

```
create table SUPPLIER
(
    KODE_SUPPLIER       int not null,
    NAMA_SUPPLIER       varchar(225),
    ALAMAT              varchar(225),
    KOTA                varchar(225),
    TELP                int,
    primary key (KODE_SUPPLIER)
);
```

Tabel : TRANSAKSI

```
create table TRANSAKSI
(
    KODE_TRANSAKSI          int not null,
    KODE_DETIL                int not null,
    USERNAME                  varchar(225) not null,
    NAMA PEMBELI            varchar(225),
    TGL_TRANSAKSI             date,
    SUB_TOTAL                 int,
    TOTAL                      int,
    primary key (KODE_TRANSAKSI)
);

alter table DETIL_TRANSAKSI add constraint FK_MEMILIKI2 foreign key
(KODE_TRANSAKSI)
references TRANSAKSI (KODE_TRANSAKSI) on delete restrict on update restrict;

alter table DETIL_TRANSAKSI add constraint FK_MEMPUNYAI foreign key
(KODE_OBAT)
references OBAT (KODE_OBAT) on delete restrict on update restrict;

alter table OBAT add constraint FK_MEMPUNYAI2 foreign key (KODE_DETIL)
references DETIL_TRANSAKSI (KODE_DETIL) on delete restrict on update restrict;

alter table OBAT add constraint FK_MENYUPLAI foreign key (KODE_SUPPLIER)
references SUPPLIER (KODE_SUPPLIER) on delete restrict on update restrict;

alter table SUPPLIER add constraint FK_MENGELOLA foreign key (USERNAME)
references ADMIN (USERNAME) on delete restrict on update restrict;

alter table TRANSAKSI add constraint FK_MELAKUKAN foreign key (USERNAME)
references ADMIN (USERNAME) on delete restrict on update restrict;

alter table TRANSAKSI add constraint FK_MEMILIKI foreign key (KODE_DETIL)
references DETIL_TRANSAKSI (KODE_DETIL) on delete restrict on update restrict;
```

Setelah Anda copy dan klik GO maka table yang terbuat seperti berikut :

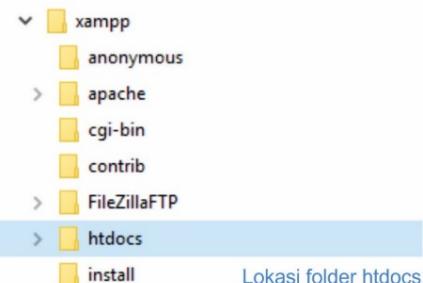
Table	Action
admin	Browse Structure Search Insert Empty Drop
detil_transaksi	Browse Structure Search Insert Empty Drop
obat	Browse Structure Search Insert Empty Drop
supplier	Browse Structure Search Insert Empty Drop
transaksi	Browse Structure Search Insert Empty Drop
5 tables	Sum

Tabel - tabel database yang terbuat

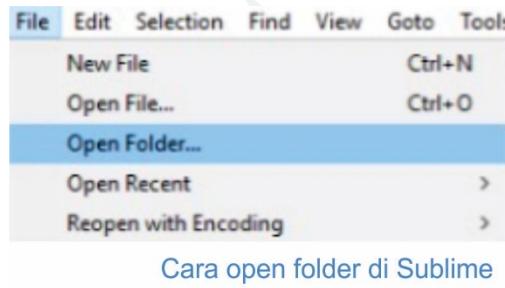
6.2 Templating

Pada tutorial ini akan digunakan template sb admin 2. Untuk mendapatkan template sb admin 2 dapat di download pada link berikut, <https://startbootstrap.com/template-overviews/sb-admin-2/>.

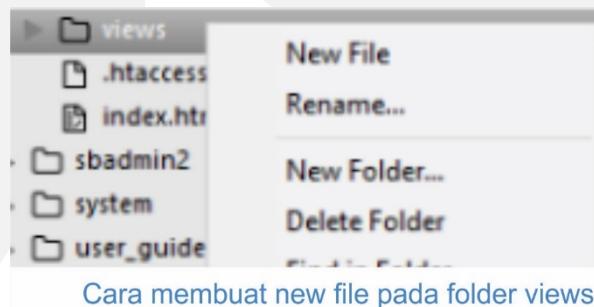
1. Siapkan paket CodeIgniter, lalu ekstrak dan copy-kan kedalam folder C:\xampp\htdocs atau sesuai tempat Anda menginstall XAMPP-nya. (Anda dapat mengganti nama foldernya). Disini saya memberi nama *codeigniter*.



2. Ekstrak juga paket sb admin 2 yang telah Anda download. Copy-kan ke dalam folder codeigniter pada folder htdocs yang tadi.
3. Buka folder codeigniter di sublime.



4. Konfigurasi 4 file CodeIgniter. Anda dapat lihat pada bab 3 poin 3.2, untuk database sesuaikan dengan nama database yang Anda buat.
5. Buat file baru dengan nama template.php pada folder *application/views/*.



6. Sekarang buka file blank.html pada folder *sbadmin2/pages/*. Dan copy-kan semua code kedalam file template.php.
7. Pada file template.php cari <div> yang berada pada komentar `<!-- Page Content -->`. Hapus 1 <div> tersebut.

```
<!-- Page Content -->
<div id="page-wrapper">
    <div class="container-fluid">
        <div class="row">
            <div class="col-lg-12">
                <h1 class="page-header">Blank</h1>
            </div>
        <!-- /.col-lg-12 -->
    </div>
    <!-- /.row -->
</div>
<!-- /.container-fluid -->
</div>
<!-- #page-wrapper -->      Code yang dihapus pada file template.php
```

Ganti codingan diatas dengan coding dibawah ini :

```
<!-- Page Content -->
<div id="page-wrapper">
    <div class="container-fluid">
        <div class="row">
            <div class="col-lg-12">
                <?php
                    $this->load->view($main_view);
                ?>
            </div>
        </div>
    </div>
</div>                                Isi page content
```

main_view nantinya akan dipanggil dengan file view lainnya yang berisi sebagai content dari halaman web-nya. Sehingga kita tidak perlu membuat header dan footer di setiap view yang dibuat.

6.2 Login

Setelah melakukan templating, selanjutnya kita coba untuk membuat halaman login untuk masuk ke dalam aplikasi. Untuk template halaman login dapat Anda download sendiri sesuai yang Anda inginkan.

Setelah download template untuk halaman login Anda dapat membuat content-nya seperti pada sintaks di bawah ini. Yang perlu diperhatikan pada tag form / <form>. Placeholder

digunakan untuk nama bayangan pada form inputan. Dan perhatikan pula name pada setiap tag input /<input>.

Untuk pengecekan nantinya Anda dapat memasukan manual pada databasenya (table Admin) dan berikan MD5 untuk passwordnya.

```
<div class="container">
<div class="row">
<div class="col-md-4 col-md-offset-4">
<div class="login-panel panel panel-default">
<div class="panel-heading">
<h3 class="panel-title">Please Sign In</h3>
</div>
<div class="panel-body">

<?php
    if(!empty($notif)){
        echo '<div class="alert alert-danger">';
        echo $notif;
        echo '</div>';
    }
?>

<form role="form" action="<?php echo base_url(); ?>index.php/Login/dologin" method="post">
<fieldset>
    <div class="form-group">
        <input class="form-control" placeholder="Username" name="Username" autofocus>
    </div>
    <div class="form-group">
        <input class="form-control" placeholder="Password" name="Password" type="password" value="">
    </div>
    <div class="checkbox">
        <label>
            <input name="remember" type="checkbox" value="Remember Me">Remember Me
        </label>
    </div>
    <!-- Change this to a button or input when using this as a form -->
    <input class="btn btn-lg btn-success btn-block" name="submit" type="submit" value="Login">
</fieldset>
</form>
</div>
</div>
</div>
</div>
```

File views login_view.php

Kita juga perlu membuat file Controller dari halaman login ini. Buat file pada folder controllers dengan nama Login.php. Adapun sintaks yang perlu dituliskan pada file Login.php sebagai berikut :

```

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

class Login extends CI_Controller {

    public function __construct()
    {
        parent::__construct();
        $this->load->model('Login_model');
        //Do your magic here
    }

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

    public function doLogin(){
        if($this->input->post('submit')){           Sintaks untuk validasi form
            $this->form_validation->set_rules('Username', 'Username', 'trim|required');
            $this->form_validation->set_rules('Password', 'Password', 'trim|required');

            if($this->form_validation->run()){
                if($this->Login_model->login() == TRUE){
                    redirect('Admin/');
                }else{
                    $data['notif'] = 'Gagal';
                    $this->load->view('login_view', $data);
                }
            }else{
                $data['notif'] = validation_errors();
                $this->load->view('login_view', $data);
            }
        }
    }
}

```

File Controller Login.php

Selanjutnya, buat file Model pada folder models dengan nama Login_model.php . Model pada login ini digunakan untuk memeriksa user di database. Adapun sintaks yang perlu dituliskan pada file Login_model.php sebagai berikut :

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

class Login_model extends CI_Model {

    public function login(){
        $username = $this->input->post('Username');
        $password = $this->input->post('Password');

        $query = $this->db->where('username', $username)
                            ->where('password', md5($password))
                            ->get('admin');

        if($query->num_rows() > 0){
            $data = array(
                'username' => $username,
                'logged_in' => TRUE
            );

            $this->session->userdata($data);

            return TRUE;
        }else{
            return FALSE;
        }
    }
}
```

Halaman Model Login_model.php

Setelah selesai membuat file view, controller dan model maka kode program siap untuk dijalankan. Buka browser Anda, dan jangan lupa untuk mengaktifkan XAMPP (Apache dan MySQL).

Hasil :

Tampilan awal saat mengakses halaman login. Ada 2 inputan yaitu Username dan Password.

Please Sign In

 Remember Me
Login

Tampilan Halaman Login

Terdapat validasi form pada halaman login ini. Saat mengklik button Login tersebut, tanpa anda mengisikan Username dan Password (kosong), validasi form tersebut akan muncul, seperti gambar di bawah ini.

Please Sign In

The Username field is required.
The Password field is required.

 Remember Me
Login

Tampilan halaman login dengan validation form

6.4 CRUD

Langkah selanjutnya kita akan membuat CRUD pada aplikasi apotiknya. Disini akan dicontohkan CRUD pada data-data obat saja, sehingga untuk data supplier dapat Anda gunakan untuk latihan mandiri.

1. File views, dengan nama add_obat_view.php

File ini digunakan untuk form inputan data obat yang juga menggunakan fasilitas upload gambar.

```
<div class="row">
    <div class="col-lg-12">
        <h1 class="page-header">Tambah Obat</h1>
    </div>
</div>

<div class="row">
    <div class="col-lg-12">
        <div class="panel panel-default">
            <div class="panel-heading">
                Tambahkan Data Obat
            </div>

            <div class="panel-body">
                <div class="row">
                    <div class="col-lg-6">

                        <?php
                            if(!empty($notif)){
                                echo '<div class="alert alert-danger">';
                                echo $notif;
                                echo '</div>';
                            }
                        ?>
                    Untuk upload file/foto
                <form role="form" method="post" enctype="multipart/form-data"
                    action="php echo base_url('index.php/Obat/simpan'); ?&gt;"&gt;

                    &lt;div class="form-group"&gt;
                        &lt;label&gt;Kode Obat&lt;/label&gt;
                        &lt;input class="form-control" placeholder="Kode Obat" name="kode_obat"&gt;
                    &lt;/div&gt;

                    &lt;div class="form-group"&gt;
                        &lt;label&gt;Kode Supplier&lt;/label&gt;
                        &lt;select class="form-control" name="kode_supplier"&gt;
                            &lt;option&gt;Pilih Kode Supplier&lt;/option&gt;
                            &lt;?php
                                foreach ($kode_supplier as $data) {
                                    echo '&lt;option&gt;' . $data-&gt;KODE_SUPPLIER . '&lt;/option&gt;';
                                }
                            ?&gt;
                        &lt;/select&gt;
                    &lt;/div&gt;
                </pre

Halaman view add_obat_view.php


```

```
<div class="form-group">
    <label>Nama Obat</label>
    <input class="form-control" placeholder="Nama Obat" name="nama_obat">
</div>

<div class="form-group">
    <label>Produsen</label>
    <input class="form-control" placeholder="Produsen" name="produsen">
</div>

<div class="form-group">
    <label>Harga</label>
    <input class="form-control" placeholder="Harga" name="harga">
</div>

<div class="form-group">
    <label>Jumlah Stok</label>
    <input class="form-control" placeholder="Jumlah Stok" name="jml_stok">
</div>

<div class="form-group">
    <label>File input</label>
    <input type="file" name="foto">
</div>

<input type="submit" class="btn btn-success" name="submit" value="Submit">
<button type="reset" class="btn btn-warning">Reset</button>
</form>
</div>
</div>
</div>
</div>
```

Lanjutan halaman view add_obat_view.php

2. File view, data_obat_view.php.

File ini digunakan untuk menampilkan data-data yang telah di inputkan pada database biasanya juga disebut select data.

```
<?php
    if (!empty($notif)) {
        echo '<div class=alert alert-danger>';
        echo $notif;
        echo'</div>';
    }
?>

<div class="row">
    <div class="col-lg-12">
        <h1 class="page-header">Data Obat</h1>
    </div>
</div>

<div class="row">
    <div class="col-lg-12">
        <div class="panel panel-default">
            <div class="panel-heading">
                Data - Data Obat
            </div>
            <div class="panel-body">
                <div class="table-responsive">
                    <table class="table table-striped table-bordered table-hover">
                        Halaman view data_obat_view.php

```

```

<thead>
    <tr>
        <th>Kode Obat</th>
        <th>Kode Supplier</th>
        <th>Nama Obat</th>
        <th>Produsen</th>
        <th>Harga</th>
        <th>Jumlah Stok</th>
        <th>Foto</th>
        <th>Aksi</th>
    </tr>
</thead>
<tbody>
<?php
    $no = 1;
    foreach ($obat as $data) {
        echo'
            <tr>
                <td>' . $data->KODE_OBAT . '</td>
                <td>' . $data->KODE_SUPPLIER . '</td>
                <td>' . $data->NAMA_OBAT . '</td>
                <td>' . $data->PRODUSEN . '</td>
                <td>' . $data->HARGA . '</td>
                <td>' . $data->JML_STOK . '</td>
                <td>' . $data->FOTO . '</td>
                <td>
                    <div class="col-md-4 col-sm-4 col-xs-4 col-lg-4">
                        <a href="'.base_url().'index.php/Obat/delete_obat/' . $data->KODE_OBAT . '">
                            <i class="glyphicon glyphicon-trash"> </i> Hapus </a>
                    </div>
                    <div class="col-md-4 col-sm-4 col-xs-4 col-lg-4">
                        <a href="'.base_url().'index.php/Obat/ambil_obat/' . $data->KODE_OBAT . '">
                            <i class="glyphicon glyphicon-search"> </i> Edit </a>
                    </div>
                </td>
            </tr> ';
    }
}
?>
</tbody>
</table>
</div>
</div>
</div>
</div>

```

Lanjutan halaman view data_obat_view.php

3. File edit_obat_view.php

File ini digunakan untuk menampilkan halaman edit data obat atau update data obat.

```
<div class="row">
    <div class="col-lg-12">
        <h1 class="page-header">Tambah Obat</h1>
    </div>
</div>

<div class="row">
    <div class="col-lg-12">
        <div class="panel panel-default">
            <div class="panel-heading">
                Tambahkan Data Obat
            </div>
            <div class="panel-body">
                <div class="row">
                    <div class="col-lg-12">

                        <form role="form" method="post" enctype="multipart/form-data"
                            action=<?php echo base_url('index.php/Obat/update_obat/'.$data->KODE_OBAT); ?>>
                            <?php
                                echo'
                            <div class="col-md-9 col-sm-9 col-xs-9 col-lg-9">
                                <div class="form-group">
                                    <label>Kode Obat</label>
                                    <input class="form-control" name="kode_obat" value="'.$data->KODE_OBAT.'">
                                </div>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

Halaman view edit_obat_view.php

```
<div class="form-group">
    <label>Kode Supplier</label>
    <input class="form-control" name="kode_supplier" value="'.$data->KODE_SUPPLIER.'">
</div>

<div class="form-group">
    <label>Nama Obat</label>
    <input class="form-control" name="nama_obat" value="'.$data->NAMA_OBAT.'">
</div>

<div class="form-group">
    <label>Produsen</label>
    <input class="form-control" name="produsen" value="'.$data->PRODUSEN.'">
</div>

<div class="form-group">
    <label>Harga</label>
    <input class="form-control" name="harga" value="'.$data->HARGA.'">
</div>

<div class="form-group">
    <label>Jumlah Stok</label>
    <input class="form-control" name="jml_stok" value="'.$data->JML_STOK.'">
</div>

<div class="form-group">
    <label>FOTO</label>
    <input class="form-control" name="foto" value="'.$data->FOTO.'">
</div>
```

Lanjutan halaman view edit_obat_view.php

```
<div class="col-md-3 col-sm-3 col-xs-3 col-lg-3" >
    <div id="foto">
        
    </div>
    <br>
    '>
    ?>

    <div class="col-md-4 col-sm-4 col-xs-4 col-lg-4">
        <input type="submit" class="btn btn-block btn-danger" name="submit" value="update">
    </div>

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

Lanjutan halaman view edit_obat_view.php

4. File controller Obat.php

File ini digunakan sebagai pengendali terhadap desain dan juga kontruksi dari penggunaan database. Kadang dibutuhkan controller lain untuk digunakan pada salah satu controller lainnya.

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

class Obat extends CI_Controller {

    public function __construct()
    {
        parent::__construct();
        $this->load->model('Obat_model');
    }      Nama file model yang digunakan
```

Halaman controller Obat.php

```

public function index()
{
    $data['obat'] = $this->Obat_model->get_data_obat();
    $data['main_view'] = 'data_obat_view';
    $this->load->view('template', $data);
}

public function add_obat(){
    $data['kode_supplier'] = $this->Obat_model->getKodeSupplier();
    $data['main_view'] = 'add_obat_view';
    $this->load->view('template', $data);
}

public function simpan(){
    if($this->input->post("submit")){
        $this->form_validation->set_rules('kode_obat', 'Kode Obat','trim|required');
        $this->form_validation->set_rules('kode_supplier', 'Kode Supplier', 'trim|required');
        $this->form_validation->set_rules('nama_obat', 'Nama Obat', 'trim|required');
        $this->form_validation->set_rules('produsen', 'Produsen', 'trim|required');
        $this->form_validation->set_rules('harga', 'Harga', 'trim|required');
        $this->form_validation->set_rules('jml_stok', 'Jumlah Stok', 'trim|required');

        if ($this->form_validation->run() == TRUE) {
            $config['upload_path'] = './image/';
            $config['allowed_types'] = 'jpg|png';
            $config['max_size'] = 2000;
            $this->load->library('upload', $config);

            if($this->upload->do_upload('foto')){
                if($this->Obat_model->insert($this->upload->data()) == TRUE){
                    $data['notif'] = 'Penambahan Data Berhasil';
                    $data['main_view'] = 'add_obat_view';
                    $this->load->view('template',$data);
                }else{
                    $data['notif'] = 'Penambahan Data Gagal!';
                    $data['main_view'] = 'add_obat_view';
                    $this->load->view('template',$data);
                }
            }else{
                $data['main_view'] = 'add_obat_view';
                $data['notif'] = $this->upload->display_errors();
                $this->load->view('template',$data);
            }
        }
    }
}

```

```

        }else{
            $data['notif'] = validation_errors();
            $data['main_view'] = 'add_obat_view';
            $this->load->view('template',$data);
        }
    }

public function delete_obat($obat){
    if ($this->Obat_model->delete_obat($obat)) {
        $data['notif'] = 'Hapus Berhasil';
        $data['main_view'] = 'data_obat_view';
        $this->load->view('template', $data);
        redirect('Obat/add_obat');
        $data['supplier'] = $this->Supplier_model->get_data_obat();

    }
}

```

Lanjutan halaman controller Obat.php

```

public function ambil_obat($kode_obat){
    $data['main_view'] = "edit_obat_view";
    $data['data'] = $this->Obat_model->detil_obat($kode_obat);
    $this->load->view('template', $data);
}

public function update_obat($kode_obat)
{
    $data['main_view'] = 'data_obat_view';
    $this->Obat_model->edit_obat($kode_obat);
    $this->load->view('template', $data);
    redirect('Obat');
}

```

Lanjutan halaman controller Obat.php

Untuk fungsi upload gambar, dibutuhkan folder untuk penyimpanan gambar-gambar yang diupload. Maka dari itu sesuai path yang telah decoding di atas buat folder di *xampp/htdocs/nama_foler_projek/image/*.

5. File Models Obat_model.php

File model ini digunakan untuk berhubungan dengan data dan interaksi ke database yang terhubung dengan projek yang dibuat.

```

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

class Obat_model extends CI_Model {

    public function insert($foto){
        $data = array(
            'kode_obat'      => $this->input->post('kode_obat'),
            'kode_supplier'  => $this->input->post('kode_supplier'),
            'nama_obat'      => $this->input->post('nama_obat'),
            'produsen'       => $this->input->post('produsen'),
            'harga'          => $this->input->post('harga'),
            'jml_stok'       => $this->input->post('jml_stok'),
            'foto'           => $foto['file_name']
        );
        $this->db->insert('obat', $data);

        if($this->db->affected_rows() > 0){
            return TRUE;
        }else{
            return FALSE;
        }
    }

    public function getKodeSupplier(){
        //MENGAMBIL KODE SUPPLIER YANG ADA DI TABEL SUPLIER
        return $this->db->order_by('kode_supplier')
                    ->get('supplier')
                    ->result();
    }

    public function get_data_obat(){
        return $this->db->get('obat')
                    ->result();
    }
}

```

Halaman model Obat_model.php

```

public function delete_obat($obat){
    $this->db->where('kode_obat', $obat)
        ->delete('obat');

}

public function detil_obat($kode_obat){
    //ngambil 1 data
    return $this->db->where('KODE_OBAT', $kode_obat)
        ->get('obat')
        ->row();
}

public function edit_obat($kode_obat)
{
    $data = array(
        'kode_obat'      => $this->input->post('kode_obat'),
        'kode_supplier'  => $this->input->post('kode_supplier'),
        'nama_obat'      => $this->input->post('nama_obat'),
        'produsen'       => $this->input->post('produsen'),
        'harga'          => $this->input->post('harga'),
        'jml_stok'       => $this->input->post('jml_stok'),
        'foto'           => $this->input->post('foto')
    );

    $this->db->where('kode_obat',$kode_obat);
    $this->db->update('obat',$data );
}

```

Lanjutan halaman model Obat_model.php

Setelah selesai semua, file-file di atas siap untuk di eksekusi. Berikut adalah tampilan dari kode program diatas :

APOTIK FARMA

- [**Obat**](#)
- [**Data Obat**](#)
- [**Tambah Obat**](#) (active)
- [**Supplier**](#)
- [**Transaksi**](#)

Tambah Obat

Tambahkan Data Obat

Kode Obat

Kode Supplier

Nama Obat

Produsen

Harga

Jumlah Stok

File input

 No file chosen

Submit
Reset

Halaman Tambah Obat

Obat

- [**Supplier**](#)
- [**Transaksi**](#)

Tambah Obat

Tambahkan Data Obat

Penambahan Data Berhasil

Kode Obat

Kode Supplier

Nama Obat

Produsen

Harga

Jumlah Stok

File input

 No file chosen

Submit
Reset

Notifikasi ketika berhasil tambah data

APOTIK FARMA

Obat	Data Obat	Tambah Obat	Supplier	Transaksi														
	Data Obat																	
	Data - Data Obat <table border="1"> <thead> <tr> <th>Kode Obat</th> <th>Kode Supplier</th> <th>Nama Obat</th> <th>Produsen</th> <th>Harga</th> <th>Jumlah Stok</th> <th>Foto</th> <th>Aksi</th> </tr> </thead> <tbody> <tr> <td>123</td> <td>123</td> <td>Panadol</td> <td>Yayuk</td> <td>5000</td> <td>20</td> <td>Panadol_Extra.png</td> <td> Hapus Edit</td> </tr> </tbody> </table>	Kode Obat	Kode Supplier	Nama Obat	Produsen	Harga	Jumlah Stok	Foto	Aksi	123	123	Panadol	Yayuk	5000	20	Panadol_Extra.png	Hapus Edit	
Kode Obat	Kode Supplier	Nama Obat	Produsen	Harga	Jumlah Stok	Foto	Aksi											
123	123	Panadol	Yayuk	5000	20	Panadol_Extra.png	Hapus Edit											

Data obat yang telah dimasukkan

APOTIK FARMA

Obat	Supplier	Transaksi

Edit Data Obat

Tambahkan Data Obat

Kode Obat	123
Kode Supplier	123
Nama Obat	Panadol
Produsen	Yayuk
Harga	5000
Jumlah Stok	20
FOTO	

update

Halaman Edit Data Obat

Daftar Pustaka

- <http://www.hakayuci.com/2016/01/install-codeigniter-3-di-xampp.html>
- <https://andrawisata.wordpress.com/php-pemograman/codeigniter-2-0-x/4-installasi-dan-konfigurasi-codeigniter/>
- <https://samsoleh.wordpress.com/category/codeigniter/>
- <https://id.wikipedia.org/wiki/CodeIgniter>
- <http://www.simplecodedaily.com/2014/03/cara-modifikasi-url-codeigniter-sesuka.html>
- https://id.wikipedia.org/wiki/PHP#Sejarah_PHP
- <http://bersamahani.blogspot.co.id/2012/06/sejarah-singkat-php.html>
- <https://fatihamaliah.wordpress.com/2013/04/02/pengertian-konsep-oop-object-oriented-programming/>
- <https://www.sinaryuda.web.id/codeigniter/codeigniter-dengan-database.html>
- <http://otakscript.blogspot.co.id/2015/06/koneksi-multi-database-di-codeigniter-3.html>
- <http://itcodetutorial.blogspot.co.id/2015/05/routing-codeigniter.html>
- <https://www.codepolitan.com/codeigniter-3-0-akhirnya-dirilis>
- <http://www.jurnalweb.com/codeigniter-3-0-akhirnya-resmi-di-rilis>
- <https://bugzilla.wordpress.com/2012/11/09/codeigniter-flow-chart-diagram-alur-codeigniter/>