



# Framework Codeigniter

---

**Sebuah Panduan dan Best Practice**

# Framework CodeIgniter

Sebuah Panduan dan Best Practice

# Kata Pengantar

---

Codeigniter merupakan salah satu framework terbaik yang ada saat ini. Banyak website-website besar telah menggunakan codeigniter sebagai framwork utama dalam membangun website maupun aplikasi mereka.

Selain kemudahan dalam menggunakan, codeigniter juga cepat, mudah di kustomisasi dan integrasikan dengan library atau framework lainnya. Framework ini juga sempat menjadi perhatian pembuat php – Rasmus Lerdorf

*"I like CodeIgniter because it is faster, lighter and the least like a framework." – Rasmus Lerdorf*

Dalam buku ini akan dibahas mengenai cara-cara penggunaan Codeigniter, mulai dari pengenalan tentang controller, model dan view, penggunaan codeigniter dan form, penggunaan database di codeigniter, studi-studi kasus hingga penggunaanya dengan library-library lain seperti jQuery dan jQuery UI.

Akhir kata penulis menyadari bahawa penulisan dalam buku ini masih jauh dari sempurna. Oleh karena itu pertanyaan, kritik dan saran dapat di emailkan ke [xibnoe@gmail.com](mailto:xibnoe@gmail.com) atau dapat mengunjungi blog penulis di <http://www.koder.web.id>

Pekanbaru, Juni 2011

Ibnu Daqiqil Id, M.Ti

# DAFTAR ISI

Perkenalan CodeIgniter 2.0 .....	1
Kenapa Menggunakan Framework? .....	2
Apa itu Codeigniter?.....	3
Apa sih Kelebihan CodeIgniter?.....	3
CodeIgniter 2.0 .....	4
Apa itu MVC?.....	5
Jangan Belajar CodeIgniter!!.....	7
PHP & Object Oriented Programming .....	9
Apasih PHP?.....	9
Sejarah PHP .....	9
Menggunakan PHP .....	9
Apa sih Object Oriented Programming (OOP)?.....	10
Apa itu Object .....	10
Apa itu Class?.....	10
Inheritance .....	12
Installasi dan Konfigurasi CodeIgniter .....	13
Mempersiapkan Web Server.....	13
Instalasi CodeIgniter .....	13
Konfigurasi CodeIgniter.....	16
Kesepakatan Coding ( <i>Coding Standart</i> ) CodeIgniter.....	18
Hello CodeIgniter .....	21
Apaan sih Controller?.....	21
Controller dan View.....	24
Mempercantik URL Codeigniter .....	26
CodeIgniter Helper dan Library .....	29
Menggunakan Library dan Helper di CodeIgniter .....	29
Library CodeIgniter.....	30
Helper CodeIgniter.....	33
Membuat Library Sendiri .....	34

Menggunakan Library External.....	34
Kasus 1. Penanganan dan Validasi Form .....	37
Penanganan Form .....	37
Menggunakan Validasi Form.....	45
Codelgniter & Database.....	51
Connect ke Database.....	51
Codelgniter Model.....	54
Melakukan Query pada Database .....	54
Query Return Value .....	55
Menggunakan Active Record .....	57
Kasus 2. CRUD dan Pagination Database .....	61
Kasus 3. Sistem Templating .....	72
Native Codelgniter Tempating.....	72
Kasus 4. Sistem Authentikasi.....	78
Kasus 5. Image Gallery Sederhana .....	87
Kasus 6. Buku Tamu menggunakan Codeigniter .....	92
Membuat Table Guestbook .....	92
Konfigurasi Guestbook.....	92
Membuat Model Guestbook .....	93
Membuat Controller Dan View .....	94
Kasus 7. Membuat Shopping Cart Sederhana .....	100
Kasus 8. Codelgniter dan Ajax.....	107
Kasus 9. Codeigniter dan jQuery AutoComplete .....	111
Kasus 10. Codeigniter dan Openflash Chart .....	116

## Chapter 1

# Perkenalan CodeIgniter 2.0

Akhir-akhir ini CodeIgniter menjadi sebuah framework yang hangat dibicarakan di Indonesia. Hampir semua milis dan forum PHP banyak membahas masalah CodeIgniter. Terlebih lagi banyak perusahaan-perusahaan ternama di Indonesia (Kompas.com, okezone.com, urbanesia.com, bejubel.com, dan lain-lain) yang telah menggunakan CodeIgniter dalam produk mereka. Apa sih CodeIgniter? CodeIgniter adalah sebuah framework PHP yang dapat mempercepat pengembangan untuk membuat sebuah aplikasi web. Ada banyak library dan helper yang berguna didalamnya dan tentunya mempermudah proses development. Ibarat ingin membangun rumah maka Anda tidak perlu membuat semen, memotong kayu menjadi papan, mengubah batu menjadi porselen dan lain-lain. Anda cukup memilih komponen-komponen tersebut lalu dikombinasikan menjadi rumah yang indah.



Gambar1. Website-website indonesia yang menggunakan Codeigniter

Jadi keuntungan yang didapat dalam penggunaan framework adalah :

- **Menghemat Waktu Pengembangan** – Dengan struktur dan library yang telah disediakan oleh framework maka tidak perlu lagi memikirkan hal-hal tersebut, jadi Anda hanya fokus ke proses bisnis yang akan dikerjakan.
- **Reuse of code** – Dengan menggunakan framework maka perkerjaan kita akan memiliki struktur yang baku, sehingga kita dapat menggunakannya kembali di proyek-proyek lainnya.
- **Bantuan komunitas** - Ada komunitas-komunitas yang siap membantu jika ada permasalahan, selain itu juga bisa berbagi ilmu sehingga dapat meningkatkan kemampuan pemrograman kita.
- **Kumpulan best practice** – sebuah framework merupakan kumpulan *best practice* yang sudah teruji. Jadi kita dapat meningkatkan kualitas kode kita.

### Catatan:

Framework adalah sebuah struktur konseptual dasar yang digunakan untuk memecahkan sebuah permasalahan atau isu-isu kompleks

Sebelum mendalami CodeIgniter lebih jauh, sebaiknya dipahami terlebih dahulu apa itu framework. Framework adalah sebuah struktur konseptual dasar yang digunakan untuk memecahkan sebuah permasalahan, bahkan isu-isu kompleks yang ada. Sebuah framework telah berisi sekumpulan arsitektur/konsep-konsep yang dapat mempermudah dalam pemecahan sebuah permasalahan. Perlu diingat, framework bukanlah peralatan/tools untuk memecahkan sebuah masalah, tetapi sebagai **ALAT BANTU**. Framework hanya menjadi sebuah konstruksi dasar yang menopang sebuah konsep atau sistem yang bersifat “*essential support*” (penting tapi bukan komponen utama).

## Kenapa Menggunakan Framework?

Salah satu alasan mengapa orang menggunakan framework terutama dalam membangun sebuah aplikasi adalah kemudahan yang ditawarkan. Didalam sebuah framework biasanya sudah tersedia struktur aplikasi yang baik, **standard coding** (1), **best practice** (2) dan **design pattern** (3), dan **common function** (4). Dengan menggunakan framework kita dapat langsung fokus kepada *business process* yang dihadapi tanpa harus berfikir banyak masalah struktur aplikasi, standar coding dan lain-lain.

Dengan memanfaatkan design pattern dan *common function* yang telah ada di dalam framework maka hal tersebut dapat mempercepat proses pengembangan aplikasi. Kita tidak perlu membuat sesuatu fungsionalitas yang bersifat umum. Tanpa disadari ketika kita membangun sebuah aplikasi yang banyak melibatkan banyak fungsionalitas yang telah dibangun itu ternyata sama atau berulang-ulang. Dengan pengelempokan itulah kita dapat mempercepat pengembangan aplikasi.

Selain kemudahan dan kecepatan dalam membangun sistem, dengan menggunakan framework tertentu kita juga dapat “menyeragamkan” cara kita mengimplementasikan kode program. Dengan framework kita akan “dipaksa” untuk patuh kepada sebuah kesepakatan. Selain itu juga akan memudahkan pengembang lain untuk mempelajari dan mengubah aplikasi yang telah dibuat apabila kode yang dihasilkan konsisten dan patuh pada sebuah aturan tertentu.

---

<sup>1</sup>Standar Coding adalah sebuah standar yang harus diikuti oleh programmer untuk menulis code.

<sup>2</sup> Best Practice adalah Kumpulan-kumpulan action yang telah teruji oleh para expert

<sup>3</sup> Design Pattern adalah Teknik-teknik yang menjadi best practise

<sup>4</sup> Common Function adalah fungsi-fungsi atau library yang telah umum digunakan dalam pengembangan sebuah sistem

## Apa itu Codeigniter?

CodeIgniter adalah sebuah web application framework yang bersifat open source digunakan untuk membangun aplikasi php dinamis. Tujuan utama pengembangan CodeIgniter adalah untuk membantu developer untuk mengerjakan aplikasi lebih cepat daripada menulis semua code dari awal. CodeIgniter menyediakan berbagai macam library yang dapat mempermudah dalam pengembangan. CodeIgniter diperkenalkan kepada publik pada tanggal 28 februari 2006.

CodeIgniter sendiri dibangun menggunakan konsep Model-View-Controller development pattern. CodeIgniter sendiri merupakan salah satu framework tercepat dibandingkan dengan framework lainnya. Pada acara frOSCon (August 2008), pembuat php Rasmus Lerdorf mengatakan dia menyukai CodeIgniter karena dia lebih ringan dan cepat dibandingkan framework lainnya ("because it is faster, lighter and the least like a framework.")

## Apa sih Kelebihan CodeIgniter?

CodeIgniter sangat ringan, terstruktur, mudah dipelajari, dokumentasi lengkap dan dukungan yang luar biasa dari forum CodeIgniter. Selain itu CodeIgniter juga memiliki fitur-fitur lainnya yang sangat bermanfaat, antara lain:

- **Menggunakan Pattern MVC.** Dengan menggunakan pattern MVC ini, struktur kode yang dihasilkan menjadi lebih terstruktur dan memiliki standar yang jelas.
- **URL Friendly.** URL yang dihasilkan sangat *url friendly*. Pada CodeIgniter diminimalisasi penggunaan `$_GET` dan digantikan dengan URI.
- **Kemudahan.** Kemudahan dalam mempelajari, membuat library dan helper, memodifikasi serta meng-integrasikan Library dan helper.

Jika kita membandingkan antara CodeIgniter dengan framework-framework lainnya maka beberapa poin yang membuat CodeIgniter unggul adalah:

- **Kecepatan.** Berdasarkan hasil benchmark CodeIgniter merupakan salah satu framework PHP tercepat yang ada saat ini.
- **Mudah dimodifikasi dan beradaptasi.** Sangat mudah memodifikasi *behavior* framework ini. Tidak membutuhkan *server requirement* yang macam-macam serta mudah mengadopsi library lainnya.
- **Dokumentasi lengkap dan jelas.** Bahkan tanpa buku ini pun CodeIgniter sebenarnya telah menyediakan sebuah panduan yang lengkap mengenai CodeIgniter. Semua informasi yang anda butuhkan tentang CodeIgniter ada disana.
- **Learning Curve Rendah.** CodeIgniter sangat mudah dipelajari. Dalam pemilihan framework hal ini sangat penting diperhatikan karena kita juga harus memperhatikan *skill* dari seluruh

anggota team. Jika sebuah framework sangat sulit dipelajari maka akan beresiko untuk memperlambat team development anda.

## **CodeIgniter 2.0**

CodeIgniter versi 2.0 baru saja diluncurkan pada awal tahun 2011. Pada versi terbaru ini, terdapat banyak perubahan mendasar dari CodeIgniter daripada versi sebelumnya. Perubahan ini menjadikan CodeIgniter jauh lebih kaya dan matang dibandingkan framework lainnya. Perubahan itu diantaranya adalah:

- Menghilangkan dukungan terhadap PHP4. PHP4 sudah tidak didukung oleh tim pengembang PHP, karena memberikan dukungan untuk PHP4 membuat CodeIgniter semakin ketinggalan dari segi fitur.
- Menghilangkan fitur pluggin. Pluggin mirip dengan helper, bertujuan untuk menghilangkan kerancuan ini maka fitur ini dihilangkan pada CodeIgniter 2.0.
- Menghilangkan fitur scaffolding. Fitur ini nyaris tidak pernah digunakan dan implementasinya masih kurang bagus.
- Penambahan library driver. Ini adalah library khusus dimana kita dapat membuat driver dari library yang telah kita buat.
- Support query string dan command-line execution. Hal ini menjawab kesulitan-kesulitan yang dialami pada versi sebelumnya.
- Penambahan library cache. Untuk meningkatkan kualitas aplikasi maka library cache baik menggunakan apc, memcached maupun file base.
- Penambahan fitur package. Untuk mempermudah distribusi resource dalam sebuah folder.

## Apa itu MVC?

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

### 1. Model

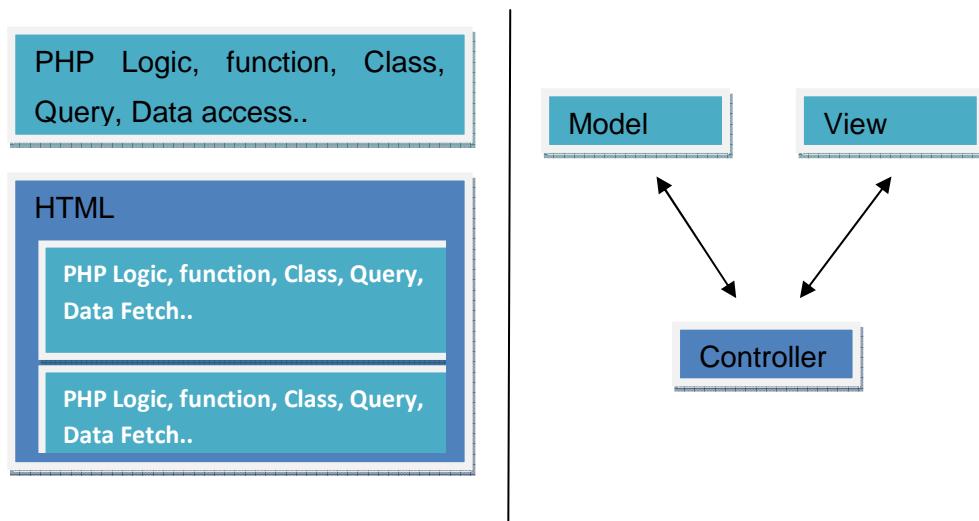
Model berhubungan dengan data dan interaksi ke database atau webservice. Model juga merepresentasikan struktur data dari aplikasi yang bisa berupa basis data maupun data lain, misalnya dalam bentuk file teks, file XML maupun webservice. Biasanya di dalam model akan berisi class dan fungsi untuk mengambil, melakukan update dan menghapus data website. Sebuah aplikasi web biasanya menggunakan basis data dalam menyimpan data, maka pada bagian Model biasanya akan berhubungan dengan perintah-perintah query SQL.

### 2. View

View berhubungan dengan segala sesuatu yang akan ditampilkan ke *end-user*. Bisa berupa halaman web, rss, javascript dan lain-lain. Kita harus menghindari adanya logika atau pemrosesan data di view. Di dalam view hanya berisi variabel-variabel yang berisi data yang siap ditampilkan. View dapat dikatakan sebagai halaman website yang dibuat dengan menggunakan HTML dan bantuan CSS atau JavaScript. Di dalam view jangan pernah ada kode untuk melakukan koneksi ke basisdata. View hanya dikhususkan untuk menampilkan data-data hasil dari model dan controller

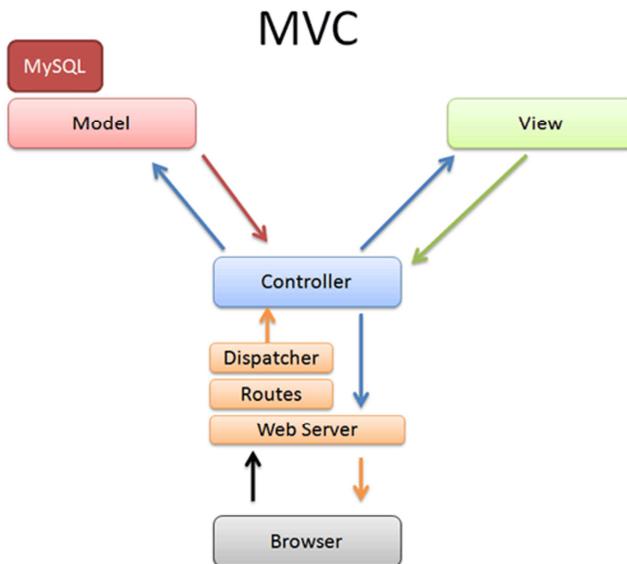
### 3. Controller:

Controller bertindak sebagai penghubung data dan view. Di dalam Controller inilah 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 mengakses data telah diserahkan kepada model. Tugas controller adalah menyediakan berbagai variabel yang akan ditampilkan di view, memanggil model untuk melakukan akses ke basis data, menyediakan penanganan kesalahan/error, mengerjakan proses logika dari aplikasi serta melakukan validasi atau cek terhadap input.



Gambar 2. Perbandingan PHP Biasa dan CodeIgniter

Perhatikan gambar 2. Pada eksekusi PHP, biasanya kita akan me-“load” semua library dan fungsi yang dibutuhkan kemudian digabungkan ke dalam HTML untuk di eksekusi oleh PHP. Untuk kasus sederhana cara tersebut masih baik-baik saja, tetapi ketika aplikasi tersebut menjadi kompleks/rumit maka kita akan sulit memeliharanya jika tidak didukung oleh arsitektur software yang bagus. Hal tersebut bisa terjadi disebabkan oleh code yang sama namun dibuat berulang-ulang , kode tidak konsisten dan lain-lain.



Gambar 3. Flow CodeIgniter

Jika dipetakan, alur kerja CodeIgniter akan tampak seperti gambar 3. Browser berinteraksi melalui controller. Controller-lah yang akan menerima dan membalas semua *request* dari browser. Untuk data maka controller akan meminta ke Model dan untuk UI/template akan meminta ke View. Jadi “Otak” dari aplikasi ada di **controller**, “Muka” aplikasi ada di **view** dan “Data” ada di **model**. Ketika

browser meminta sebuah halaman web maka router akan mencarikan controller mana yang harus menangani *request* tersebut. Setelah itu barulah si controller menggunakan model untuk mengakses data dan View untuk menampilkan data tersebut.

## **Jangan Belajar CodeIgniter!!**

Jangan coba-coba belajar CodeIgniter ketika Anda belum mengenal PHP. Berdasarkan pengalaman di forum dan milis, kebanyakan mereka yang belajar CodeIgniter tanpa memiliki dasar PHP yang baik akan mengalami banyak kesulitan, bukan dalam menguasai konsep CodeIgniter tetapi masih berkutat seputar PHP. Setidaknya Anda telah memahami konsep OOP pada PHP untuk mulai belajar CodeIgniter. Sebaiknya ketika Anda mempelajari CodeIgniter Anda sudah memahami PHP dan Object Oriented Programming. Akan lebih bagus lagi jika Anda sudah familiar dengan design pattern.



## Chapter 2

# PHP & Object Oriented Programming

Syarat utama untuk menguasai CodeIgniter adalah memahami PHP dan konsep Object Oriented Programming. Pada bab ini kita akan membahas sedikit mengenai PHP dan OOP.

## Apasih PHP?

PHP (PHP: Hypertext Preprocessor) adalah sebuah bahasa pemogramaman di sisi server. Ketika Anda mengakses sebuah URL, maka web browser akan melakukan request ke sebuah web server. Misalnya kita me-request sebuah file PHP <http://www.koder.web/index.php>, maka webserver akan melakukan *parsing* terhadap file PHP tersebut. PHP parser yang menjalankan kode-kode PHP yang terdapat pada file index.php lalu mengirimkan hasilnya ke web browser.

## Sejarah PHP

Pada tahun 1994, Rasmus Lerdorf mengembangkan sebuah perkakas yang digunakan sebagai *engine parsing* sebagai penerjemah/interpreter beberapa *macro*. Pada saat itu engine digunakan untuk pembuatan buku tamu, counter dan beberapa homepage. Ia menamai engine parser tersebut dengan nama PHP/FI.

Dengan semangat *opensource*, para pengembang di dunia mencoba mengembangkan PHP/FI. Sampai pada tahun 1997, lebih dari 500.000 website di dunia menggunakan PHP/FI untuk menyelesaikan masalah seperti konkesi kedatabase, menampilkan content yang dinamis dan lain-lain.

Pada juni 1998, PHP 3.0 dirilis. Pada saat itu PHP sudah mendukung *multiflatform* (bukan hanya linux), webserver, sejumlah database, SNMP (Simple Network Management Protocol) and IMAP (Internet Message Access Protocol). Menurut survei yang dilakukan oleh <http://netcraft.org>, saat ini pengguna PHP sudah mencapai 9,5 juta domain.

## Menggunakan PHP

Ketika kita ingin mempelajari PHP, hal pertama yang harus disiapkan adalah sebuah web server, seperti yang sudah dibahas sebelumnya bahwa PHP merupakan bahasa pemrograman di sisi server. Banyak sekali jenis web server yang dapat digunakan, antara lain Apache, IIS, iPlanet, Omni, Xintami, dan lain-lain. Setelah web server terinstal barulah kita install PHP sebagai modul.

PHP dapat diperoleh secara gratis dengan cara men-download dari situs resmi PHP (<http://www.php.net/downloads.php>) atau website lain yang mempunyai salinan program PHP untuk di-download.

Sebelum menginstall PHP, terlebih dahulu harus meng-install web server. Setelah PHP terinstall. Anda dapat meletakkan source code PHP ke dalam folder yang akan di publish oleh web server dengan ekstensi file .php.

Jika tidak ingin direpotkan oleh proses konfigurasi dan implementasi webserver, Anda dapat menggunakan software yang sudah terpaket menjadi satu antara PHP, APACHE, dan MySQL. Contoh Software-nya adalah XAMPP, PHPTRIAD, FOX Server, dan lain-lain.

## Apa sih Object Oriented Programming (OOP)?

Object Oriented Programming (OOP) merupakan paradigma pemrograman yang berorientasikan kepada obyek. Semua data dan fungsi pada paradigma ini dibungkus dalam kelas-kelas atau obyek-obyek. Bandingkan dengan logika pemrograman terstruktur, setiap obyek dapat menerima pesan, memproses data, dan mengirim pesan ke obyek lainnya. OOP diciptakan untuk mengatasi keterbatasan pada bahasa pemrograman tradisional. Konsep dari OOP sendiri adalah semua pemecahan masalah dibagi ke dalam obyek. Dalam konsep OOP data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai obyek.

### Apa itu Object

Sederhananya, sebuah obyek adalah kumpulan dari variabel dan fungsi yang dibungkus menjadi satu entitas. Entitas tersebut dapat berupa variabel biasa. Sebuah obyek diciptakan melalui sebuah kelas atau dengan istilah *instance of class*. Obyek memiliki 2 elemen utama:

1. **Attributes atau Properties:** Yaitu nilai-nilai yang tersimpan dalam objek tersebut dan secara langsung maupun tidak langsung menentukan karakteristik dari obyek tersebut.
2. **Method:** Yaitu suatu aksi yang akan dijalankan atau dikerjakan oleh obyek tersebut.

### Apa itu Class?

Class dapat didefinisikan sebagai struktur data atau cetak biru dari suatu obyek. Lebih jelasnya adalah sebuah bentuk dasar atau *blueprint* yang mendefinisikan variabel, method umum pada semua obyek. Obyek sendiri adalah kumpulan variabel dan fungsi yang dihasilkan dari template khusus atau disebut class. Obyek adalah elemen pada saat run-time yang akan diciptakan, dimanipulasi, dan dibuang/di-destroy ketika eksekusi. Adapun class merupakan definisi statik dari himpunan obyek yang mungkin diciptakan sebagai instantiasi dari class.

Perhatikan contoh class di berikut ini.

```
<?php
/** Contoh kelas ***/
class Kendaraan{}
/** end of class ***/
?>
```

Contoh diatas memperlihatkan bagaimana mendefinisikan sebuah class dan meng-create sebuah instance dari class. Pada contoh di atas kita membuat sebuah kelas bernama "Kendaraan". Dalam pembuatan kelas, pertama kita menggunakan kata kunci **class** yang diikuti oleh **nama kelas**, kemudian diakhiri dengan kurung kurawal. Di dalam kurung kurawal kita menuliskan kode-kode (berisi property dan method) supaya kelas tersebut bekerja seperti yang diinginkan.

Kode-kode di dalam sebuah kelas terbagi menjadi dua kelompok, yaitu **property** dan **method**. Property adalah suatu wadah penyimpanan di dalam kelas yang bisa menampung informasi. Sederhananya property itu bisa disebut sebagai variabel di dalam kelas. Sedangkan method adalah fungsi yang ada di dalam kelas. Perhatikan contoh berikut.

```
<?php
/** contoh kelas ***/
class kendaraan{

    /** property class ***/
    private $warna;
    private $jumlah_pintu;
    private $jumlah_roda;
    public $harga;
    public $merk;

    /** method class ***/
    public function __construct(){
        echo 'ini adalah object kendaraan. <br />';
    }

    public function set_harga($harga){
        $this->harga = $harga;
    }

    public function show_harga(){
        echo 'harga kendaraan : rp.'.$this->harga.' <br />';
    }

    public function jalan(){
        echo 'brrroooooom!!!!';
    }
}

/** end of class ***/
/** contoh object ***/
#mengcreate object $saya_adalah_object dari class kendaraan
$saya_adalah_object= new kendaraan;
$saya_adalah_object->set_harga(100000);
$saya_adalah_object->show_harga();
```

```
$saya_adalah_object->jalan();
```

Setiap property dan method memiliki *identifier*. Identifier-lah yang mengatur bagaimana property dan method digunakan. Identifier tersebut adalah **public**, **private** dan **protected**. Private berarti method atau property yang ada di dalam suatu kelas hanya bisa diakses di dalam kelasnya. Sedangkan pada method atau property yang bersifat public berarti method atau property tersebut bisa diakses di dalam dan di luar kelas.

## Inheritance

Inheritance atau dalam bahasa Indonesiannya disebut sebagai pewarisan adalah suatu cara untuk membuat sebuah kelas yang baru dengan menggunakan kelas lain yang sebelumnya sudah dibuat. Pada hubungan inheritance, sebuah class turunan mewarisi kelas leluhur (parent class). Oleh karena mewarisi, maka semua atribut dan method class dari induk akan dibawa (kecuali yang bersifat private), secara intrinsik menjadi bagian dari class anak. Adapun keuntungan yang didapat dari inheritance menambah fitur baru pada kelas anak dan mengubah atau mengganti fitur yang diwarisi dari kelas parent

Adapun contoh kelas yang menggunakan konsep inheritance adalah

```
<?php
/* inheritance.php */
class Bapak {
    private $nama = "Bapak";
    function Bapak($n) {
        $this->nama = $n;
    }
    function Hallo() {
        echo "Halo, saya $this->nama <br>";
    }
}

class Anak extends Bapak {
    function Hai(){
        Echo "hai dari kelas anak";
    }
}

$test = new Anak("Anak dari Bapak");
$test->Hallo();
```

Jika kita perhatikan di kelas anak sama sekali tidak memiliki fungsi hello, tetapi karena parentnya memiliki fungsi tersebut maka si anak dapat menggunakan fungsi tersebut. Selain menggunakan fungsi bapak, anak juga dapat menambah fungsi baru yaitu fungsi Hai.

## Chapter 3

# Installasi dan Konfigurasi Codelgniter

Agar dapat menggunakan Codelgniter, Anda harus menginstall dan melakukan konfigurasi terhadap Codelgniter terlebih dahulu. Installasi Codelgniter sangatlah mudah. Dengan menggunakan konfigurasi default saja maka Codelgniter sudah dapat berjalan di web server Anda. Hal-hal yang harus dipersiapkan dalam menginstall Codelgniter adalah

## Mempersiapkan Web Server

Seperti yang kita tahu bahwa PHP adalah bahasa pemrograman website yang berjalan disisi server oleh karena itu untuk dapat menjalankan website yang dibuat dengan menggunakan PHP, di komputer harus terinstall aplikasi web server yang mendukung PHP. 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.

Untuk website yang melibatkan basis data sebagai tempat penyimpanan datanya maka tidak perlu bingung karena Codelgniter juga mendukung berbagai jenis server basis data, yaitu MySQL, MySQLi, MS SQL, PostgreSQL, Oracle, dan SQLite. Codelgniter juga bisa dijalankan di semua sistem operasi yang bisa menjalankan aplikasi-aplikasi di atas, baik Windows, Linux, BSD dan yang lainnya.

## Instalasi Codelgniter

Instalasi Codelgniter sangat mudah. Meskipun namanya instalasi tetapi karena Codelgniter adalah aplikasi berbasis website maka sebenarnya yang perlu dilakukan adalah meng-copy folder aplikasi Codelgniter ke dalam folder **htdocs** atau **DocumentRoot** dari web server yang telah diinstall sebelumnya. Berbeda dengan instalasi software pada umumnya.

Sebelum melakukan instalasi yang perlu dilakukan pertama kali adalah mendapatkan kode sumber dari Codelgniter itu sendiri yang dapat didownload di <http://www.Codelgniter.com/> (disediakan pula di dalam CD penyerta buku). Selanjutnya letakkan folder hasil ekstrak tadi di DocumentRoot web server, yaitu folder htdocs didalam direktori **C:\apachefriends\xampp** bagi yang menggunakan XAMPP di Windows atau **/var/www/html** bagi yang menggunakan linux (semua tergantung dimana anda menginstall dan mengkonfigurasi webserver).

Dengan menggunakan konfigurasi default saja maka CodeIgniter sudah dapat berjalan di web server Anda. Adapun struktur utama dari CodeIgniter terbagi menjadi dua bagian, yaitu **application** dan **sistem/core CodeIgniter**. Application adalah tempat kita meletakkan code yang akan dibuat (warna merah dan hijau sedangkan sistem/core CodeIgniter yang warna ungu) . Folder sistem berisi library-library dan helper bawaan CodeIgniter.

Name	Date Modified	Size	Kind
application	Mar 15, 2011 1:54 PM	--	Folder
cache	Mar 15, 2011 1:54 PM	--	Folder
config	Mar 15, 2011 1:54 PM	--	Folder
controllers	Mar 15, 2011 1:54 PM	--	Folder
core	Mar 15, 2011 1:54 PM	--	Folder
errors	Mar 15, 2011 1:54 PM	--	Folder
helpers	Mar 15, 2011 1:54 PM	--	Folder
hooks	Mar 15, 2011 1:54 PM	--	Folder
language	Mar 15, 2011 1:54 PM	--	Folder
libraries	Mar 15, 2011 1:54 PM	--	Folder
logs	Mar 15, 2011 1:54 PM	--	Folder
migrations	Mar 15, 2011 1:54 PM	--	Folder
models	Mar 15, 2011 1:54 PM	--	Folder
third_party	Mar 15, 2011 1:54 PM	--	Folder
views	Mar 15, 2011 1:54 PM	--	Folder
index.html	Mar 15, 2011 1:53 PM	4 KB	HTML Document
system	Mar 15, 2011 1:54 PM	--	Folder
core	Mar 15, 2011 1:54 PM	--	Folder
database	Mar 15, 2011 1:54 PM	--	Folder
fonts	Mar 15, 2011 1:54 PM	--	Folder
helpers	Mar 15, 2011 1:54 PM	--	Folder
language	Mar 15, 2011 1:54 PM	--	Folder
libraries	Mar 15, 2011 1:54 PM	--	Folder
index.html	Mar 15, 2011 1:53 PM	4 KB	HTML Document
user_guide	Mar 15, 2011 1:54 PM	--	Folder
index.php	Mar 15, 2011 1:53 PM	8 KB	PHP
license.txt	Mar 15, 2011 1:53 PM	4 KB	Plain Text

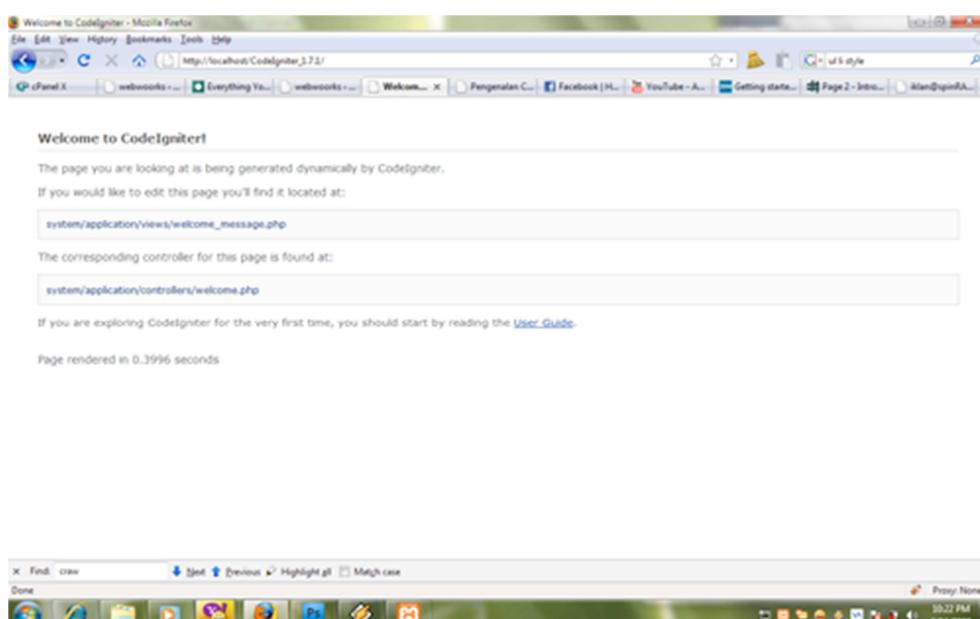
Gambar 4. Struktur Direktori CodeIgniter

Adapun susunan folder CodeIgniter secara default adalah:

- Folder **application**: disinilah aplikasi yang akan kita bangun diletakkan.
  - ✓ Folder **config** - tempat menyimpan semua file konfigurasi yang ada di dalam aplikasi, mulai dari database, router dan autoload aplikasi.
  - ✓ Folder **controllers** - tempat menyimpan semua file controller.
  - ✓ Folder **errors** - tempat menyimpan semua template error aplikasi.
  - ✓ Folder **helpers** - tempat menyimpan helper-helper yang bukan berasal dari CI.
  - ✓ Folder **hooks** - tempat menyimpan hook yang digunakan untuk mengubah alur fungsi dari core CI.
  - ✓ Folder **language** - tempat menyimpan bahasa-bahasa yang akan di gunakan.
  - ✓ Folder **libraries** - tempat menyimpan semua library buatan kita sendiri.
  - ✓ Folder **models** - tempat menyimpan semua model.
  - ✓ Folder **views** - tempat menyimpan semua file view aplikasi.

- Folder **system** menyimpan semua file baik itu file aplikasi yang dibuat maupun core framework-nya.
  - ✓ Folder **cache** - tempat menyimpan semua cache yang dibuat caching library.
  - ✓ Folder **codeigniter** - tempat menyimpan semua semua file internals CI.
  - ✓ Folder **database** - tempat menyimpan semua driver database drivers dan class yang akan digunakan.
  - ✓ Folder **fonts** - tempat menyimpan semua font yang digunakan image manipulation library.
  - ✓ Folder **helpers** - tempat menyimpan semua helper core CI.
  - ✓ Folder **language** - tempat menyimpan semua language core CI.
  - ✓ Folder **libraries** - tempat menyimpan semua library core CI
  - ✓ Folder **logs** - tempat menyimpan semua logs generated oleh CI.
  - ✓ Folder **plugin** - tempat menyimpan semua plugin core CI.
  - ✓ Folder **scaffolding** - tempat menyimpan semua file yang berfungsi sebagai scaffolding .
- Folder **user\_guide** berisi userguide/manual penggunaan CI.
- File **index.php** file yang akan menghandle semua request yang dilakukan oleh client.

Setelah meletakkan CodeIgniter ke dalam folder htdocs maka akan didapatkan tampilan seperti di bawah ini, artinya CodeIgniter telah sukses berjalan di aplikasi Anda.



Gambar 5. Tampilan browser ketika sukses menginstall Codeigniter

Adapun checklist yang harus dilakukan untuk menjalankan CodeIgniter secara default adalah

- Pastikan Apache dan PHP telah terinstall dan berjalan di komputer.
- Pastikan peletakkan source code CodeIgniter di folder/direktory web apache (biasanya htdocs) dan memiliki permission setidak-tidaknya read only atau kode 644.

## 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**. Adapun file-file yang terdapat dalam direktori tersebut dan sering digunakan antara lain:

- **Config.php.** Pada file konfigurasi config.php berisi konfigurasi secara umum mengenai CodeIgniter, seperti peletakan baseurl, suffix, frontcontroller, serta metode yang digunakan URI dan lain-lain. Adapun konfigurasi-konfigurasi yang perlu diperhatikan adalah :

- **\$config['base\_url']** - Konfigurasi ini berisi alamat url sebuah aplikasi. Jika menggunakan helper url maka konfigurasi ini harus di-set dengan benar. Contoh: aplikasi Anda akan diakses dengan menggunakan domain www.contoh.com/app\_ci maka pada konfigurasi ini harus diisikan:

```
$config['base_url']="http://www.contoh.com/app_ci/";
```

Tetapi jika ingin menggunakan base url yang lebih fleksibel maka dapat menggantinya dengan variabel server. Contoh:

```
$config['base_url']="http://" . $_SERVER['HTTP_HOST'] .  
str_replace(basename($_SERVER['SCRIPT_NAME']), "",  
$_SERVER['SCRIPT_NAME']);
```

- **\$config['index\_php']** - Konfigurasi ini berisi file yang menjadi frontcontroller. Konfigurasi ini berhubungan dengan **base\_url**. Jika menggunakan **.htaccess** untuk mempercantik url maka isi variabel ini harus dikosongkan.
  - **\$config['uri\_protocol']** - Konfigurasi ini bertujuan untuk menentukan bagaimana library URI bekerja. CodeIgniter dapat menangkap URI yang di berikan melalui 4 cara yaitu **PATH\_INFO**, **QUERY\_STRING**, **REQUEST\_URI** dan **ORIG\_PATH\_INFO**. Masing-masing cara mempunyai kelebihan dan kekurangan masing-masing, bahkan tidak semua web server mendukung semua cara tersebut, oleh karena itu secara default digunakan pilihan auto. Tetapi jika aplikasi membutuhkan sesuatu yang lebih *custom* maka pemilihan metode URI secara langsung akan lebih baik. Lebih lanjut akan dibahas dibagian library.
  - **\$config['url\_suffix']** - Konfigurasi ini bertujuan untuk menambahkan akhiran pada url. Contoh Anda mempunyai sebuah controller page, maka controller tersebut akan diakses melalui http://localhost/index.php/page, dengan menambahkan **url\_suffix** berisi “html” maka url tadi dapat juga diakses melalui http://localhost/index.php/page.html.

- **\$config['language']** - Secara default CodeIgniter sudah mendukung banyak bahasa/multy language. Kita dapat mengubah pesan-pesan yang ada di dalam CodeIgniter dengan bahasa yang kita kehendaki. Untuk mengubah bahasa tersebut cukup dengan mendownload paket bahasa yang diinginkan lalu uraikan di dalam direktori **system/languages/[nama\_lang]** lalu Anda tinggal mengubah nama\_lang di config.
- **\$config['enable\_hooks']** - Konfigurasi ini bertujuan mengaktifkan/menonaktifkan hook pada CodeIgniter. Hook dapat dikatakan event-event yang terjadi pada CodeIgniter, dimana kita bisa meletakkan fungsi di dalamnya. Hook akan bermanfaat sekali ketika Anda ingin mengubah perilaku CodeIgniter maupun untuk *logging event*. Contoh: Anda ingin mengubah urutan *loading library* dimana Anda sudah meng-extend library router untuk menggunakan database untuk aturan routing-nya. Artinya Anda harus meload library database sebelum library loader. Hal tersebut bisa dilakukan melalui hook.
- **\$config['subclass\_prefix'] = 'MY\_'**. Jika kita ingin mengubah/mengextend library CodeIgniter maka library tersebut harus memiliki prefik yang sama dengan konfigurasi ini.
- **\$config['permitted\_uri\_chars']**. Konfigurasi ini bertujuan untuk keamanan CodeIgniter. Konfigurasi ini menentukan karakter apa saja yang boleh digunakan di dalam uri.
- **\$config['log\_threshold']**. Konfigurasi ini menentukan bagaimana sistem logging CodeIgniter bekerja. Sistem logging ini sangat membantu dalam proses pengembangan terutama ketika *debugging*. Jika di set 0 maka tidak ada proses logging error di CodeIgniter. Jika di set 1 maka yang dicatat hanyalah pesan-pesan kesalahan yang termasuk kesalahan PHP. Jika di set 2 maka akan menampilkan semua pesan debug dan pesan kesalahan CodeIgniter dan PHP. Jika di set 3 maka logging-logging yang berisi informasi seperti sebuah library telah di load juga akan ditampilkan. Jika di set 4 maka semuanya akan dilog mulai dari error, pesan debug sampai yang bersifat informasi.
- **\$config['log\_path']**. Dikonfigurasi ini kita dapat menentukan dimana log akan diletakkan. Jika diisi kosong maka akan diletakkan di system/logs. Harus dingat Anda harus mengubah tingkat hak akses dari direktori tersebut menjadi dapat ditulisi / *writeable* jika kita menggunakan fasilitas logging.
- Konfigurasi session. Session di CodeIgniter menggunakan cookies jadi kita dapat mengeset waktu hidup cookies/expire, nama cookies dan lain-lain melalui konfigurasi ini.

```
$config['sess_cookie_name'] = 'Nama cookie';
$config['sess_expiration'] = 7200;
$config['sess_encrypt_cookie'] = FALSE;
$config['sess_use_database'] = FALSE;
$config['sess_table_name'] = 'session_table';
$config['sess_match_ip'] = FALSE;
$config['sess_match_useragent'] = TRUE;
$config['sess_time_to_update'] = 300;
```

- **Autoload.php.** Konfigurasi ini bertujuan untuk mementukan sumber daya apa yang akan diload secara otomatis. Cara penggunaannya sederhana, misalnya kita ingin meload library database, pagination dan lain-lain secara otomatis maka kita tinggal mengubah `$autoload['libraries']` menjadi :

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

- **Routes.php.** Konfigurasi di file ini bertujuan untuk menentukan kemana routing oleh library route akan dilakukan. Hal paling sederhana yang harus dilakukan adalah mengubah default controller (controller yang akan dibuka ketika tidak ada uri yang diberikan oleh browser). Misalnya website kita beralamat [www.koder.web.id](http://www.koder.web.id). Maka ketika membuka website tersebut maka secara otomatis CodeIgniter akan mengalihkan ke controller default, karena tidak disertakan di dalam url kita. Adapun yang perlu diubah adalah

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

## Kesepakatan Coding (*Coding Standart*) CodeIgniter

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

- PHP Closing Tag

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

**Contoh salah:**

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

**Contoh Benar:**

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

```
/* End of file myfile.php */  
/* Location: ./system/modules/mymodule/myfile.php */
```

- Penamaan Class dan Method

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

**Contoh salah:**

```
class superclass  
class SuperClass
```

**Contoh Benar:**

```
class Super_class
```

Aturan diatas juga berlaku untuk method contohnya

**Contoh kurang tepat:**

```
function fileproperties() //Tidak deskriptif dan memiliki underscore  
function fileProperties() // Tidak deskriptif dan underscore CamelCase  
function getfileproperties() // Kurang underscore  
function getFileProperties() // menggunakan CamelCase  
get_the_file_properties_from_the_file() // terlalu panjang
```

**Contoh Tepat:**

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



## Chapter 3

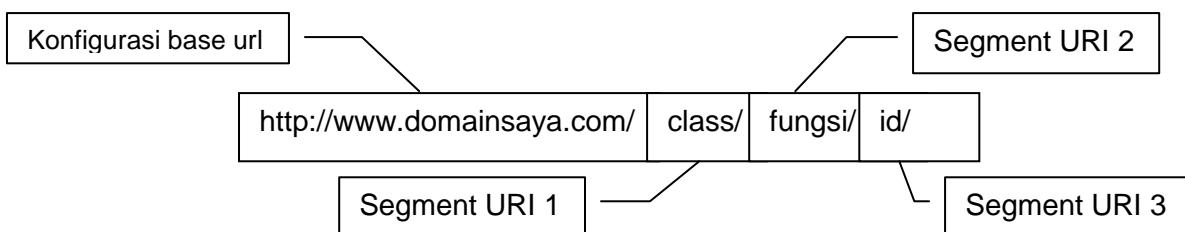
# Hello CodeIgniter

Pada bab ini kira akan memulai coding menggunakan Codeigniter. Sebelum memulai maka perlu kita pahami kembali apa itu Controller? Dan bagaimana hubungan controller dengan URI?

## Apaan sih Controller?

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

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



Adapun komponen-komponen URL diatas adalah

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

Untuk contoh pertama, kita akan membuat sebuah aplikasi hello codeigniter yang sederhana. Aplikasi tersebut akan diletakkan di folder hello dari htdoc anda, sehingga anda dapat mengaksesnya dengan membuka url <http://localhost/hello>. Perhatikan URL Berikut ini:

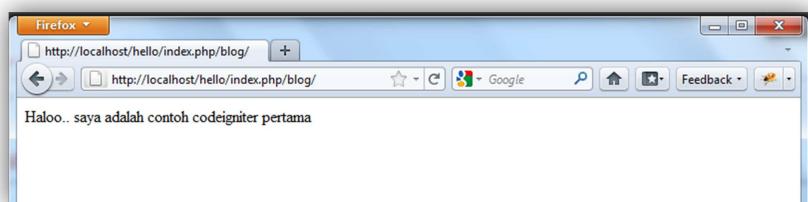
[http://localhost/hello/index.php/\*\*blog\*\*/](http://localhost/hello/index.php/blog/)

Pada contoh di atas, CodeIgniter akan berusaha untuk menemukan sebuah controller bernama blog pada file blog.php, kemudian menampilkannya ke browser. Mari kita coba membuat sebuah controller sederhana sehingga dapat melihat apa yang terjadi. Dengan menggunakan teks editor, buatlah file bernama **blog.php** di dalam folder application/controller. Setelah itu buatlah sebuah kelas Blog yang merupakan turunan dari kelas CI\_Controller.

```
1. <?php if ( ! defined('BASEPATH'))  
2.         exit('No direct script access allowed');  
3.  
4. class Blog extends CI_Controller {  
5.  
6.     function __construct()  
7.     {  
8.         parent::__construct();  
9.     }  
10.  
11.    function index()  
12.    {  
13.        echo "Haloo.. saya adalah contoh codeigniter pertama";  
14.    }  
15.  
16. }  
17. /* End of file Blog.php */  
18. /* Location: ./application/controllers/blog.php */
```

Setelah itu apa yang terjadi? Coba buka situs Anda dengan menggunakan URL seperti ini:

[http://localhost/index.php/\*\*blog\*\*/](http://localhost/index.php/blog/)



Gambar 6. Tampilan Aplikasi Hello Codeigniter

Jika Anda melakukannya dengan benar, maka akan tampak tulisan **Hello.. saya adalah contoh CodeIgniter pertama**. Sekedar mengingatkan kembali, ketentuan penamaan class (baik controller maupun library) harus dimulai dengan huruf besar. Contoh:

```
<?php
class Blog extends CI_Controller {
}
```

Berikut ini contoh yang salah :

```
<?php
class blog extends CI_Controller {
}
```

Pada contoh di atas nama fungsi yang dipanggil adalah index (). Fungsi "Index" akan selalu dibaca secara default jika **segmen kedua** dari URI kosong. Cara lain untuk menampilkan "Hello CodeIgniter" adalah dengan mengakses url berikut:

localhost/index.php/blog/index/

Segmen kedua dari URI yang menentukan fungsi mana yang akan dipanggil dari controller. Mari kita coba menambahkan fungsi baru ke controller:

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

class Blog extends CI_Controller {

    function __construct()
    {
        parent::__construct();
    }

    function index()
    {
        echo "Haloo.. saya adalah contoh codeigniter pertama";
    }

    function komentar()
    {
        echo "Ini adalah fungsi komentar";
    }
}

/* End of file Blog.php */
/* Location: ./application/controllers/blog.php */
```

Sekarang cobalah panggil URL berikut untuk melihat fungsi komentar:

localhost/index.php/blog/komentar/

Maka hasilnya akan menampilkan pesan "Ini Fungsi Komentar". Jika ingin mengisi variabel \$param maka Anda tinggal menambahkan sebuah URI lagi setelah Blog. Biasanya parameter ketiga atau lebih digunakan sebagai parameter (kecuali Anda menambahkan folder di dalam folder controller). Sebagai contoh URL yang disertai parameter seperti ini:

Fungsi post pada controller dapat memiliki dua parameter yang akan dilewatkan pada URI segmen 3 dan 4 ( "tutorial-ci" dan "123"). Jadi dapat disimpulkan bahwa URI yang dipanggil dapat ditunjukkan seperti ini:

localhost/index.php/[Controller]/[Fungsi]/[param]/[param]/...

## Codeigniter Tips

### Mendefinisikan Default Controller

Seperti halnya Controller yang memiliki fungsi default yang di eksekusi, Codeigniter juga memiliki Controller default untuk di panggil atau dijalankan. Untuk menentukan default controller, buka folder application/config/routes.php dan ubah variabel ini pada file routes.php:

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

Dimana *Blog* adalah nama kelas controller yang ingin digunakan. Jika sekarang hanya memanggil file index.php utama, tanpa menentukan segmen URI apapun, maka secara default Anda akan melihat pesan Hello Codeigniter.

## Controller dan View

Pada contoh program Hello Codeigniter diatas adalah cara untuk menampilkan tulisan "hello Codeigniter" secara langsung di controller. Namun sebenarnya hal tersebut bisa dilakukan di view. Sebagai contoh: tuliskan kode program di bawah ini pada file **application/controller/blog.php** :

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

class Blog extends CI_Controller {

    function __construct()
    {
        parent::__construct();
    }

    function index()
    {
        $this->load->view("hello_codeigniter");
    }

}
/* End of file Blog.php */
/* Location: ./application/controllers/ Blog.php */
```

Selanjutnya buatlah file hello\_CodeIgniter.php di folder application>view (**application/view/hello\_codeigniter.php**) yang berisi tulisan:

```
<h1> Hello saya adalah view </h1>
```

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

Sebuah View sebenarnya hanyalah sebuah halaman web atau bagian dari halaman web, seperti sebuah header, footer, sidebar, dan lain-lain. Bahkan, View bisa menjadi fleksibel karena view dapat dimasukkan ke dalam view yang lain jika dibutuhkan. Untuk memanggil file view dapat digunakan fungsi seperti berikut ini (perhatikan baris 12):

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

*Nama\_view* adalah nama file view Anda. Dan file tersebut harus diletakkan di dalam folder application/view.

Fungsi view sendiri memiliki 3 parameter:

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

Contoh: buatlah file controller dengan nama **blog.php** dengan kode program seperti berikut ini:

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

class Blog extends CI_Controller {
    function __construct()
    {
        parent::__construct();
    }
    function index()
    {
        $data['judul']="Judul blog";
        $data['isi']="Isi blog";
        $this->load->view("blog_view",$data);
    }
}
/* End of file Blog.php */
/* Location: ./application/controllers/ Blog.php */
```

Selanjutnya buatlah file view bernama **blog\_view.php** dengan kode program seperti berikut ini:

```
<h1><?php echo $judul;?></h1>
<p><?php echo $isi; ?></p>
<p><br />Page rendered in {elapsed_time} seconds</p>
```

Jika halaman blog.php dipanggil maka tampilannya kurang lebih seperti berikut:



3. **Output Parameter** - Parameter ini akan di set true jika kita ingin menyimpan hasil view ke dalam sebuah variabel. Jika kita mengambil contoh code controller blog sebelumnya maka kita tinggal mengubah cara pemanggilan view menjadi

```
$out = $this->load->view("blog_view",$data,true);
```

Code diatas berarti kita akan menyimpan hasil view kedalam sebuah variabel. Contohnya :

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

class Blog extends CI_Controller {

    function __construct()
    {
        parent::__construct();
    }

    function index()
    {
        $this->load->view("hello_codeigniter");
    }

    function view()
    {
        $data['judul'] = "Judul blog";
        $data['isi'] = "Isi blog";
        $out = $this->load->view("blog_view",$data,true);
        echo $out;
    }
}
```

## Mempercantik URL Codeigniter

Jika kita memperhatikan url yang ada di website kompas.com, okezone.com ayau bejubel.com maka kita akan menemukan url yang unik. Contohnya <http://bejubel.com/jual/pakaian-wanita> , seakan-akan url tersebut bukan lah berasal dari script php biasa. Untuk membuat url seperti diatas

kita bisa menggunakan settingan codeigniter dan file .htaccess. adapun langkah yang dapat dilakukan adalah :

1. Membuat file .htaccess di folder root aplikasi anda. Adapun isi dari difile .htaccess adalah

```
RewriteEngine on  
RewriteCond $1 !^(index\.php|images|robots\.txt)  
RewriteRule ^(.*)$ /index.php/$1 [L]
```

2. Mengubah konfigurasi application/configs/config.php. Membuang "index.php" pada \$config['index\_page'] = "index.php"; menjadi \$config['index\_page'] = ""; di file application/config/config.php

Dengan menyelesaikan tahap kedua anda telah bisa ngehilangkan index.php pada url misalnya url <http://localhost/hello/index.php/blog/> sudah dapat diakses dengan url <http://localhost/hello/blog/> tanpa index.php lebih menari bukan?

Dengan sampai pada tahap kedua sebenarnya sudah cukup untuk membuat url yang menarik. Selain itu untuk lebih advance lagi kita dapat menggunakan konfigurasi application/config/router.php.



## Chapter 5

# Codelgniter Helper dan Library

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

- **Library**

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

- **Helper**

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

## Menggunakan Library dan Helper di Codelgniter

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

1. **Menambahkan Pada Konfigurasi Autoload**

Menambahkan sebuah library di autoload berarti seluruh aplikasi Anda akan dapat menggunakan library tersebut secara langsung. Sebaiknya library yang di load dengan cara ini adalah jenis library yang dipakai di seluruh aplikasi seperti login, template, dan lain-lain.

2. **Menggunakan Perintah Loader Library**

Kita dapat juga menggunakan library **loader** untuk men-load library. Library loader adalah sebuah library Codelgniter yang otomatis di load. Loader berfungsi sebagai pengatur dari sumberdaya-sumberdaya yang ada di dalam Codelgniter seperti Model, View, Library, Helper, dan pluggin. Cara penggunaannya adalah:

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

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

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

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

## Library CodeIgniter

Secara default CodeIgniter telah menyediakan library yang dapat digunakan secara langsung. Adapun library yang telah tersedia antara lain:

- **Benchmarking Class**

Library ini digunakan untuk melakukan pengukuran terhadap aplikasi yang dibuat. Seperti untuk mengetahui berapa lama waktu eksekusi dan berapa jumlah memori yang digunakan. Library ini sudah digunakan dan diload secara otomatis oleh CodeIgniter.

- **Calendar Class**

Library ini berfungsi untuk menampilkan dan men-generate kalender.

- **Cart Class**

Library ini berfungsi untuk membuat shopping cart (keranjang belanja). Library ini memiliki ketergantungan terhadap kelas session karena item-item chart tersebut disimpan di dalam session.

- **Config Class**

Library ini berfungsi untuk mengambil data-data di dalam file konfigurasi. Library ini sudah diload secara otomatis oleh CodeIgniter.

- **Database Class**

Library database digunakan untuk memanipulasi serta mendapatkan data dari sebuah sistem database. Secara default database yang sudah didukung oleh CodeIgniter adalah mysql, mssql, oracle, postgres. Sedangkan database yang tidak didukung secara langsung oleh CodeIgniter dapat dijembatani dengan driver odbc.

- **Email Class**

Library email digunakan untuk mengirimkan email. Pengiriman email tersebut bisa dilakukan dengan menggunakan protokol mail, sendmail dan smtp.sqawd

- **Encryption Class**

Library Encryption digunakan untuk melakukan penyandian terhadap string tertentu

- **File Uploading Class**

Library Uploading digunakan untuk meng-upload file. Kelas ini sudah dilengkapi dengan pengecekan jenis file, dan ukuran file.

- **Form Validation Class**

Library form Validation digunakan untuk mengecek keabsahan form-form yang sudah di-submit oleh user.

- **FTP Class**

Library FTP digunakan untuk meng-upload atau download file melalui ftp server.

- **HTML Table Class**

Library HTML table adalah sebuah kelas yang berfungsi untuk men-generate table dari data array.

- **Image Manipulation Class**

Library image manipulation berfungsi untuk mengolah gambar. Adapun fungsi-fungsi yang telah disediakan adalah Image Resizing, Thumbnail Creation, Image Cropping, Image Rotating dan Image Watermarking.

- **Input and Security Class**

Library Input dan security berfungsi untuk menjamin bahwa inputan dari form telah bersih dari karakter-karakter “aneh”.

- **Loader Class**

Library ini dapat disebut sebagai pengatur sumberdaya CodeIgniter. Semua sumberdaya yang ada akan dikendalikan oleh kelas ini. Library ini sudah di-load secara otomatis oleh CodeIgniter.

- **Language Class**

Library language digunakan untuk mengatur bahasa apa yang akan dipakai oleh CodeIgniter.

- **Output Class**

Library Output bertujuan untuk meng-handle output dari CodeIgniter, mulai dari cache sampai ke profiling bisa dilakukan kelas ini.

- **Pagination Class**

Untuk mem-paginate hasil database untuk performance dan usability, kita bisa mengontrol berapa banyak record untuk ditampilkan disetiap halaman website, berapa banyak record untuk ditarik dari database dan tampilan dari bagian pagination

- **Session Class**

Library Session dapat digunakan untuk memelihara informasi status tentang user (seperti layaknya session di PHP). Tetapi Library ini tidak menggunakan session built-in dari PHP, Library Session men-generate session datanya sendiri yang disimpan di dalam Cookies.

- **Trackback Class**

Library Trackback digunakan untuk mengirim dan menerima data trackback.

- **Template Parser Class**

Library Template Parser digunakan untuk membuat template yang berisi parsable pseudo – templates.

- **Unit Testing Class**

Library Unit Testing digunakan untuk unit test function dalam aplikasi yang sedang dibuat. CodeIgniter menyediakan fungsi evaluasi dan dua fungsi hasil dalam library ini.

- **URI Class**

Library URI digunakan untuk memarsing URL, lalu memecahnya ke dalam beberapa segment dan kemudian di-passing ke controller atau disimpan sebagai variabel.

- **User Agent Class**

Library User Agent digunakan untuk mengidentifikasi browser, mobile device, atau robot yang mengunjungi website. Kita juga bisa menggunakannya untuk mendeteksi dukungan bahasa, sekumpulan karakter, dan referrer.

- **XML-RPC Class**

Library XML-RPC digunakan untuk men-setup klien XML-RPC dan server.

- **Zip Encoding Class**

Library Zip Encoding digunakan untuk membuat file ZIP baik yang berjenis teks maupun data binary.

## Helper CodeIgniter

Helpers seperti namanya akan membantu Anda membangun aplikasi dengan tugas tertentu. Tidak seperti library, helper bukanlah Object Oriented tapi berupa prosedural. Setiap helper berisi satu atau lebih fungsi, masing-masing berfokus pada tugas tertentu yang tidak ada ketergantungan dengan fungsi lainnya.

Helper dapat juga di-load secara otomatis dalam /system/application/config/autoload.php. Adapun helper yang ada di dalam CodeIgniter antara lain:

- **Array** — Helper array berisi fungsi yang membantu pekerjaan berhubungan dengan array. Sebagai contoh fungsi **random\_element()** mengambil array sebagai input dan menghasilkan elemen random darinya.
- **Cookie** — Helper cookie berisi fungsi yang membantu pekerjaan berhubungan dengan pemberian nilai, pembacaan data cookies, dan penghapusan data cookie.
- **Date** — Helper date berisi fungsi yang membantu pekerjaan berhubungan dengan tanggal. Sebagai contoh, fungsi **now()** menghasilkan waktu sekarang sebagai UNIX timestamp.
- **Directory** — Helper direktori berisi fungsi tunggal yang membantu pekerjaan berhubungan dengan direktori. Sebagai contoh fungsi **directory\_map()** adalah untuk membaca path direktori tertentu dan membangun array-nya yang berisi semua file-filenya dan subdirektorinya.
- **Download** — Helper download berisi fungsi tunggal yang membantu men-download data dengan mudah. Fungsi **force\_download()** menghasilkan header server yang memaksa data untuk di-download.
- **File** — Helper file berisi fungsi yang membantu untuk membaca, menulis, dan menghapus file.
- **Form** — Helper form berisi fungsi-fungsi yang membantu membangun form.
- **HTML** — Helper HTML berisi fungsi yang membantu membuat blok HTML dengan cepat dan mudah. Sebagai contoh fungsi **ul()** bisa mengubah array item ke bulleted list.
- **Inflector** — Helper inflector berisi fungsi yang membantu mengubah kata-kata menjadi bentuk plural atau singular, memberlakukan camel case atau mengubah kata-kata yang dipisahkan oleh spasi menjadi phrase yang digaris bawahi, sebagai contoh fungsi singular bisa mengubah string ‘girls’ menjadi ‘girl’.
- **Security** — Helper security berisi fungsi yang berhubungan dengan keamanan seperti **xss\_clean()**, yang akan menyaring setiap kode yang mungkin digunakan dalam cross site scripting hack.
- **Smiley** — Helper smiley berisi fungsi-fungsi yang membantu pengelolaan *emoticons*.
- **String** — Helper string berisi fungsi-fungsi yang membantu pekerjaan berhubungan dengan string, seperti fungsi **random\_string()** yang akan membuat string random berdasarkan tipe dan panjang argumen.

- **Text** — Helper text berisi fungsi-fungsi yang membantu Anda bekerja dengan teks. Sebagai contoh: fungsi **word\_limiter**, dapat membatasi string ke sejumlah kata tertentu yang berguna untuk membatasi input user pada form.
- **Typography** — Helper typography berisi fungsi tunggal yang membantu memformat teks dengan cara yang tepat. Sebagai contoh: fungsi **auto\_typography()** me-wrap paragraph dengan <p> dan </p>, mengkonversi line breaks ke <br/> dan mengkonversi tanda kutip, dash, dan ellipse dengan baik.
- **URL** — Helper URL berisi fungsi-fungsi yang membantu bekerja dengan URL. Anda akan menggunakan fungsi **base\_url()** dan **anchor()** dalam setiap projek.
- **XML** — Helper XML berisi fungsi tunggal yang membantu bekerja dengan XML. Fungsi **xml\_convert** berfungsi untuk mengkonversi string menjadi teks XML, mengkonversi ampersand dan angle bracket menjadi entity.

## Membuat Library Sendiri

Untuk mendapatkan kode yang bagus sebaiknya fungsionalitas yang sama itu di bungkus menjadi sebuah library. Sebuah library adalah sebuah kelas yang diletakkan pada folder application/libraries

Get\_instance() adalah sebuah fungsi yang mengimplementasikan singleton dari controller CI. Jika Anda membuat library sendiri dan membutuhkan data atau resource lainnya yang terdapat di object utama maka kita dapat menggunakan fungsi tersebut untuk mengakses data atau resource yang dibutuhkan.

Sebagai contoh kita akan membuat sebuah library breadcrumb yang akan memanfaatkan data dari segmen-segmen URI. Data URI dapat diakses dari library uri yang sudah di-load secara otomatis oleh Codeigniter.

## Menggunakan Library External

Codeigniter sangat mudah untuk mengadopsi library yang bersifat external. Library yang paling mudah untuk diadopsi adalah sebuah single class tanpa parameter pada constructor. Untuk kelas yang memiliki konstruktor lebih dari satu maka sebaiknya kelas tersebut diubah terlebih dahulu atau jika kelas library tersebut terdiri atas beberapa class maka lebih mudahnya kita membuat sebuah library factory dari library tersebut.

### Contoh 1. Library CSV Reader

Sebagai contoh kita akan menggunakan sebuah library yang berfungsi untuk membaca file CSV kedalam memori.. Adapun isi dari librari tersebut adalah

```
<?php if(!defined('BASEPATH'))exit('No direct script access allowed');  
/**
```

```

* CSVReader Class
* @author Pierre-Jean Turpeau
* @link http://www.CodeIgniter.com/wiki/CSVReader
*/
class csv_reader {

var $fields;
var $separator=';';
var $enclosure="'";
var $max_row_size=4096;

/**
* Parse a file containing CSV formatted data.
*
* @access public
* @param string
* @param boolean
* @return array
*/
}

function parse_file($p_Filepath, $p_NamedFields=true)
{
    $content=false;
    $file= fopen($p_Filepath,'r');
    if($p_NamedFields)
    {
        $this->fields = fgetcsv($file,$this->max_row_size,
        $this->separator,$this->enclosure);
    }

    while(($row= fgetcsv($file,$this->max_row_size, $this-
    >separator,$this->enclosure))!=false)
    {
        if($row[0]!=null)
        {// skip empty lines
            if(!$content){$content=array();}
            if($p_NamedFields)
            {
                $items=array();
                foreach($this->fields as$id=>$field)
                {
                    if(isset($row[$id]))
                        {$items[$field]=$row[$id];}
                }
                $content[]=$items;
            }
            else
            { $content[]=$row; }
        }
    }
    fclose($file);
    return $content;
}
}

```

Agar dapat menggunakan library tersebut maka lakukan langkah-langkah berikut ini:

- **Copy Library Ke Direktori application/libraries**

Untuk menggunakan library buatan kita sendiri atau library eksternal maka langkah yang pertama yang harus dilakukan adalah meletakkan library itu di direktori application/libraries. Nama file dan nama kelas harus sama. Contoh nama file adalah csv\_reader.php maka nama kelasnya adalah csv\_reader

- **Load Library Menggunakan Library Loader**

Setelah meletakkan library tersebut di direktori application/libraries kita tinggal meload library atau kelas tersebut. Setelah di load maka nama library tersebut akan menjadi property di kelas controller.

Perhatikan controller berikut ini

```
1. <?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');
2.
3. class Welcome extends CI_Controller {
4.
5.     function __construct()
6.     {
7.         parent::__construct();
8.
9.     }
10.
11.    function index()
12.    {
13.        $this->load->library('csv_reader');
14.        $data = $this->csv_reader->parse_file(FCPATH.'data_nilai.csv');
15.
16.        echo "<pre>";
17.        print_r($data);
18.
19.    }
20. }
21.
22. /* End of file welcome.php */
23. /* Location: ./application/controllers/welcome.php */
```

Pada baris ke-13 kita akan mencoba menggunakan library csv\_reader, kelas tersebut akan memarsing sebuah file CSV. Lalu hasil parsingan tersebut kita tampilkan di layar browser anda.

## Chapter 6

# Kasus 1. Penanganan dan Validasi Form

Pada bab ini akan dibahas mengenai **penanganan form** dan **cara melakukan validasi** terhadap sebuah form inputan. Penanganan form artinya bagaimana cara kita untuk mendapatkan dan mengolah data yang dikirimkan oleh pengguna melalui form. Untuk membuat form sendiri, sebaiknya kita menggunakan helper form. Setelah kita berhasil mendapatkan data dari user, kita harus memvalidasi data-data yang telah dikirimkan tersebut agar aplikasi yang kita bangun menjadi aman. Inputan yang tidak divalidasi membuat aplikasi kita rentan terhadap serangan seperti Sql Injection, XSS dan lain-lain.

## Penanganan Form

Untuk menangani form kita membutuhkan library input. Library tersebut sudah dipanggil secara **otomatis** oleh codeigniter. Fungsi yang tersedia pada library ini selain untuk menangani form juga memiliki fungsi security/keamanan. Contohnya untuk menghandle atau memfilter XSS kita dapat menggunakan fungsi `xss_filter`.

```
$nama = $this->input->xss_filter($this->input->post('nama'));
```

Jika kita ingin melindungi seluruh aplikasi dari dengan XSS filter maka kita bisa menconfignya dengan mengubah konfigurasi di `system/application/config/config.php` dari

```
$config['global_xss_filtering'] = FALSE;
```

Menjadi

```
$config['global_xss_filtering'] = TRUE;
```

Selain kedua cara diatas, kita dapat juga menggunakan parameter kedua dari fungsi post atau get maka kita akan mendapatkan hasil yang sama dengan kedua cara diatas, contohnya

```
$this->input->post('nama', TRUE);
$this->input->get('nama', TRUE);
```

Fungsi post digunakan untuk menangkap inputan POST sedangkan get digunakan untuk menangkap inputan GET. Selain itu fungsi input juga dapat digunakan untuk cookie, variabel SERVER, dan user agent

```
$this->input->cookie('nama_cookie', TRUE);
$this->input->ip_address();
$this->input->server('PATH_INFO', TRUE);
```

Selain library input, Codeigniter juga menyediakan sebuah helper untuk mempermudah penanganan form. Helper tersebut adalah helper form. Helper tersebut membantu kita dalam membuat form pada view. Adapun cara meload helper form adalah

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

Ada banyak fungsi yang disediakan oleh helper ini diantaranya form\_open(), form\_close(), form\_open\_multipart(), form\_hidden(), form\_password(), form\_textarea(), form\_dropdown(), form\_multiselect(), form\_fieldset() dan lain-lain. Untuk lebih detailnya anda dapat melihat pada dokumentasi codeigniter. Pada kasus ini, fungsi yang banyak digunakan adalah fungsi diantaranya form\_open(), form\_close() dan form\_input().

**Form\_open()** berfungsi untuk menghasilkan tag form (<form>) pada html. Contoh:

```
$attributes = array('class' => 'email', 'id' => 'myform');
echo form_open('form/send', $attributes);
```

Maka akan menghasilkan

```
<form method="post" accept-charset="utf-8" action="http://localhost/index.php/email/send" class="email" id="myform" />
```

Form\_input berfungsi untuk menggenerate tag input pada html. Fungsi ini dapat dipakai dengan dua cara. Cara pertama adalah dengan memasukkan nama tag dan valunya. Contoh:

```
echo form_input('username', 'ibnoe');
```

akan menghasilkan

```
<input type="text" name="username" id="username" value="ibnoe">
```

Cara kedua adalah dengan mempassing array kedalam fungsi tersebut. Cara ini adalah cara yang advance dan sangat berguna jika kita ingin mengcustomisasi inputan. Contohnya:

```
$data = array(
    'name'      => 'username',
    'id'        => 'username',
    'value'     => 'ibnoe',
    'maxlength' => '100',
    'size'      => '50',
    'style'     => 'width:50%',
);
echo form_input($data);
```

Akan menghasilkan output :

```
<input type="text" name="username" id="username" value="ibnoe" maxlength="100"
size="50" style="width:50%" />
```

Untuk contoh menangani form maka ikutilah langkah-langkah berikut ini.

## 1. Membuat Controller Hitung

Perlu diingat hal pertama yang harus di konfigurasi pada setiap aplikasi adalah **BASE\_URL**. Setelah melakukan konfigurasi base\_url pada file **application/config/config.php** maka kita akan membuat sebuah controller yang akan menangani proses perkalian dan pembagian. Konfigurasi pada base\_url dilakukan supaya penggunaan library url menghasilkan keluaran yang tepat. Perhatikan code berikut ini:

### **Application/controllers/hitung.php**

```

1. <?php if (! defined('BASEPATH')) exit('No direct script access
   allowed');
2.
3. class Hitung extends CI_Controller {
4.
5. function __construct()
6. {
7.   parent::__construct();
8.   $this->load->helper(array('url','form'));
9. }
10.
11. function index()
12. {
13.   $this->load->view('menu_hitung');
14. }
15.
16. function perkalian()
17. {
18.   $data['v1']=(int)$this->input->post('v1',true);
19.   $data['v2']=(int)$this->input->post('v2',true);
20.   $data['hasil']=$data['v1']*$data['v2'];
21.   $this->load->view('perkalian',$data);
22. }
23.
24. function pembagian()
25. {
26.   $data['v1']=(int)$this->input->post('v1',true);
27.   $data['v2']=(int)$this->input->post('v2',true);
28.   if ($data['v2']>0)
29.     $data['hasil']=$data['v1']/$data['v2'];
30.   else
31.     $data['hasil']='Error, v2 tidak boleh 0!';
32.   $this->load->view('pembagian',$data);
33. }
34. }
```

Pada baris ke 8, dilakukan loading terhadap helper url dan form sekaligus. Loading tersebut dilakukan dalam fungsi konstruktor karena kedua helper tersebut digunakan pada semua fungsi dan view yang ada. Pada kelas hitung tersebut, terdapat 3 fungsi yaitu index, perkalian dan pembagian. Pada fungsi index hanya berisi sebuah pemanggilan terhadap view. View

tersebut akan berisi menu-menu yang akan mengarahkan pengguna untuk melakukan perkalian atau pembagian.

Pada fungsi perkalian dan pembagian akan dilakukan proses perkalian atau pembagian. Pada fungsi tersebut, variabel yang berasal dari view akan ditangkap menggunakan fungsi post (berarti dikirimkan melalui POST method).

## 2. Membuat View

Setelah membuat controller maka kita akan membuat view yang merupakan interface dari aplikasi tersebut. Code berikut ini adalah view menu\_hitung. View ini berisi navigasi ke fungsi perkalian dan pembagian.

### Application/views/menu\_hitung.php

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="utf-8">
5.  <title>Welcome to CodeIgniter</title>
6.
7.  <style type="text/css">
8.
9.  body {
10. background-color: #fff;
11. margin: 40px;
12. font-family: Lucida Grande, Verdana, Sans-serif;
13. font-size: 14px;
14. color: #4F5155;
15. }
16.
17. a {
18. color: #003399;
19. background-color: transparent;
20. font-weight: normal;
21. }
22.
23. h1 {
24. color: #444;
25. background-color: transparent;
26. border-bottom: 1px solid #D0D0D0;
27. font-size: 16px;
28. font-weight: bold;
29. margin: 24px 0 2px 0;
30. padding: 5px 0 6px 0;
31. }
32.
33. </style>
34. </head>
35. <body>
36.
37. <h1>CodeIgniter 2.0 dan Form!</h1>
38.
```

```

39. <p>Silakan pilih menu dibawah ini.</p>
40. <ul>
41. <li><?php echo anchor('hitung/perkalian','Perkalian');?>
42. <li><?php echo anchor('hitung/pembagian','Pembagian');?>
43. </ul>
44. <p><br/>Page rendered in {elapsed_time} seconds</p>
45.
46. </body></html>

```

Perhatikan baris ke-41 dan ke-42, itu adalah contoh penggunaan helper. Fungsi anchor bertujuan untuk membuat sebuah link ke page tertentu. Berikut ini adalah tampilan dari view diatas jika dipanggil.



Gambar 7. Screenshot view menu\_hitung

Setelah menu hitung selanjutnya membuat view untuk perkalian dan pembagian. Adapun view perkalian adalah sebagai berikut.

#### Application/views/perkalian.php

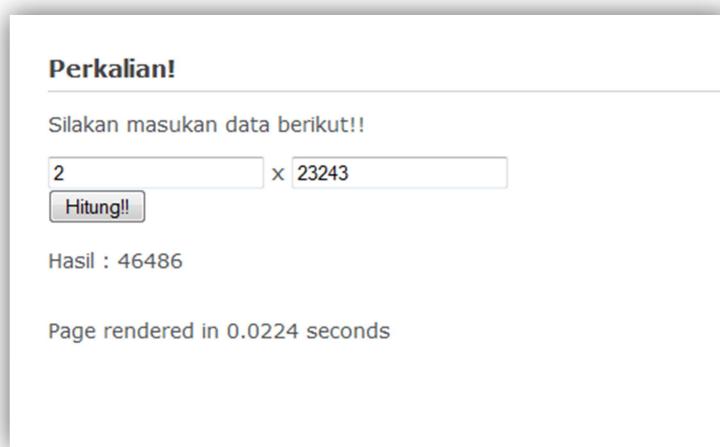
```

1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="utf-8">
5.  <title>Welcome to CodeIgniter</title>
6.
7.  <style type="text/css">
8.
9.  body {
10.    background-color: #fff;
11.    margin: 40px;
12.    font-family: Lucida Grande, Verdana, Sans-serif;
13.    font-size: 14px;
14.    color: #4F5155;
15.  }
16.
17. a {
18.   color: #003399;
19.   background-color: transparent;
20.   font-weight: normal;
21. }
22.

```

```
23. h1 {  
24.   color: #444;  
25.   background-color: transparent;  
26.   border-bottom: 1px solid #D0D0D0;  
27.   font-size: 16px;  
28.   font-weight: bold;  
29.   margin: 24px 0 2px 0;  
30.   padding: 5px 0 6px 0;  
31. }  
32.  
33. </style>  
34. </head>  
35. <body>  
36.  
37. <h1>Perkalian!</h1>  
38.  
39. <p>Silakan masukan data berikut!!</p>  
40. <?php echo form_open('hitung/perkalian');?>  
41. <?php echo form_input('v1',$v1);?> x  
42. <?php echo form_input('v2',$v2);?><br>  
43.  
44. <?php echo form_submit('submit','Hitung!!!');?>  
45. <?php echo form_close();?><br>  
46. Hasil : <?php echo $hasil;?>  
47.  
48. <p><br/>Page rendered in {elapsed_time} seconds</p>  
49.  
50. </body></html>
```

Pada view ini banyak digunakan helper form (baris 40 – 45). Tag input yang ada pada view ini di-generate oleh fungsi form\_input dan tombol submit oleh fungsi form\_submit. Adapun tampilan datil view tersebut adalah



Gambar 8. Screenshot view perkalian

Yang terakhir adalah membuat view untuk pembagian. Pada prinsipnya pembuatanya sama saja dengan view perkalian tetapi bentuk layoutnya berbeda.

**Application/views/pembagian.php**

```

1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="utf-8">
5.  <title>Welcome to CodeIgniter</title>
6.
7.  <style type="text/css">
8.
9.  body {
10.    background-color: #fff;
11.    margin: 40px;
12.    font-family: Lucida Grande, Verdana, Sans-serif;
13.    font-size: 14px;
14.    color: #4F5155;
15.  }
16.
17. a {
18.   color: #003399;
19.   background-color: transparent;
20.   font-weight: normal;
21. }
22.
23. h1 {
24.   color: #444;
25.   background-color: transparent;
26.   border-bottom: 1px solid #D0D0D0;
27.   font-size: 16px;
28.   font-weight: bold;
29.   margin: 24px 0 2px 0;
30.   padding: 5px 0 6px 0;
31. }
32.
33. </style>
34. </head>
35. <body>
36.
37. <h1>pembagian!</h1>
38.
39. <p>Silakan masukan data berikut!!</p>
40. <?php echo form_open('hitung/pembagian');?>
41. <?php echo form_input('v1',$v1);?> /
42. <?php echo form_input('v2',$v2);?><br>
43.
44. <?php echo form_submit('submit','Hitung!!!');?>
45. <?php echo form_close();?><br>
46. Hasil : <?php echo $hasil;?>
47.
48. <p><br/>Page rendered in {elapsed_time} seconds</p>
49. </body></html>
```

Berikut ini adalah tampilan dari form pembagian

**pembagian!**

Silakan masukan data berikut!!

/

Hasil : 111

Page rendered in 0.0215 seconds

Gambar 9. Screenshot view pembagian

**Catatan :**

## Menggunakan Validasi Form

Sebenarnya sampai pada tahap kedua di atas kita telah mampu membuat sebuah aplikasi perkalian dan pembagian sederhana, tetapi sebuah aplikasi yang baik harus memiliki sebuah kontrol terhadap inputan user.

Validation form sangat penting pada aplikasi. Sebuah aplikasi akan memiliki banyak inputan dari pengguna dan semua inputan untuk pengguna harus aman. CodeIgniter memiliki sebuah library Form Validation Library yang akan membantu kita untuk membuat sebuah validasi yang cepat, mudah dan aman.

Pada contoh validasi form ini hampir sama dengan 2 langkah diatas, tetapi memiliki sedikit penambahan terutama dalam fungsi perkalian dan pembagian. Perhatikan controller hitung berikut ini (controller berikut ini berasal dari contoh sebelumnya dengan penambahan).

### Application/controllers/hitung.php

```

1. <?php if (! defined('BASEPATH')) exit('No direct script access
   allowed');
2.
3. class Hitung extends CI_Controller {
4.
5. function __construct()
6. {
7.     parent::__construct();
8.     $this->load->helper(array('url','form'));
9. }
10.
11. function index()
12. {
13.     $this->load->view('menu_hitung');
14. }
15.
16.
17. function perkalian()
18. {
19.     $this->load->library('form_validation');
20.     $this->form_validation->set_rules('v1','Variabel 1',
21.     'required|integer');
22.     $this->form_validation->set_rules('v2','Variabel 2',
23.     'required|integer');
24.     if ($this->form_validation->run())
25.     {
26.         $data['v1']=(int)$this->input->post('v1',true);
27.         $data['v2']=(int)$this->input->post('v2',true);
28.         $data['hasil']=$data['v1']*$data['v2'];
29.     }
30.     else
31.     {
32.         $data['v1']=0;
33.         $data['v2']=0;
34.         $data['hasil']=0;

```

```

35. }
36. $this->load->view('perkalian', $data);
37. }
38.
39. function pembagian()
40. {
41.     $this->load->library('form_validation');
42.     $this->form_validation->set_rules('v1', 'Variabel 1',
43.         'required|is_natural_no_zero');
44.     $this->form_validation->set_rules('v2', 'Variabel 2',
45.         'required|is_natural_no_zero');
46.     if ($this->form_validation->run())
47.     {
48.         $data['v1']=(int)$this->input->post('v1',true);
49.         $data['v2']=(int)$this->input->post('v2',true);
50.         $data['hasil']=$data['v1']/$data['v2'];
51.     }
52.     else
53.     {
54.         $data['v1']=0;
55.         $data['v2']=0;
56.         $data['hasil']=0;
57.     }
58.
59.     $this->load->view('pembagian', $data);
60. }
61.
62. }
```

Perhatikan baris 19-24 pada fungsi perkalian, disana kita me-load sebuah library bernama **form\_validation**. Library tersebut memiliki fungsi untuk melakukan validasi terhadap inputan user berdasarkan aturan yang telah kita tentukan. Perhatikan baris 20

```
$this->form_validation->set_rules('v1', 'Variabel 1', 'required|integer');
```

Baris di atas merupakan salah satu contoh untuk menentukan aturan inputan pada form. Parameter pertama (berisi v1) merupakan name dari inputan yang akan divalidasi (harus sama dengan attribut name pada tag input contoh: <input name="v1">). Parameter kedua adalah nama dari inputan tersebut. Nama tersebut dapat berbeda atau sama dengan name pada parameter pertama. Parameter ketiga merupakan aturan dari inputan tersebut. Aturan-aturan tersebut harus dipatuhi agar sebuah form dapat diproses. Aturan-aturan tersebut dipisahkan oleh tanda |. Pada contoh di atas berarti **inputan v1 harus diisi (required) dan berisi data integer(integer)**.

Adapun aturan-aturan yang tersedia di dalam CodeIgniter adalah:

Aturan	Keterangan
required	Isi inputan tidak boleh kosong.
matches	Isi inputan harus sama dengan inputan tertentu contoh

	matches[password]
min_length	Panjang inputan harus memiliki jumlah minimal karakter, contoh penggunaan min_length[6] artinya panjang inputan harus lebih besar dari 6 karakter.
max_length	Panjang inputan harus memiliki jumlah maksimal karakter, contoh max_length[12] artinya panjang inputan tidak boleh lebih dari 12 karakter.
exact_length	Panjang inputan harus sama dengan jumlah karakter yang diinginkan, contoh exact_length[8] artinya panjang inputan harus sama dengan 8 karakter.
alpha	Inputan harus berisi semua huruf alfabet mulai dari a-z.
alpha_numeric	Inputan harus berisi karakter alfabet dan numeric.
alpha_dash	Inputan harus berisi semua huruf alfabet mulai dari a-z , underscores atau dashes.
numeric	Inputan hanya boleh berisi angka / huruf numeric.
integer	Inputan hanya boleh berisi angka integer saja.
is_natural	Inputan hanya boleh berisi bilangan natural saja: 0, 1, 2, 3, dan seterusnya.
is_natural_no_zero	Inputan hanya boleh berisi bilangan natural saja kecuali nol: 1, 2, 3, dan seterusnya.
valid_email	Inputan harus berisi format email yang benar.
valid_emails	Inputan harus berisi format email yang benar di batasi dengan koma jika alamat email lebih dari satu.
valid_ip	Inputan harus berisi format IP yang benar.
valid_base64	Inputan harus berisi format karakter base64 yang benar.

Pada baris-46, kita melakukan pengecekan terhadap rule-rule yang telah diset (\$this->form\_validation->run()). Fungsi tersebut akan menghasilkan nilai TRUE apabila semua

rule terpenuhi dan menghasilkan nilai FALSE jika sebaliknya. Selain fungsi-fungsi diatas Anda juga dapat membuat *custom validation* jika aturan-aturan di atas tidak ada yang memenuhi dengan kebutuhan Anda.

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="utf-8">
5.  <title>Perkalian menggunakan validasi!!</title>
6.
7.  <style type="text/css">
8.
9.  body {
10.    background-color: #fff;
11.    margin: 40px;
12.    font-family: Lucida Grande, Verdana, Sans-serif;
13.    font-size: 14px;
14.    color: #4F5155;
15.  }
16.
17. a {
18.   color: #003399;
19.   background-color: transparent;
20.   font-weight: normal;
21. }
22.
23. h1 {
24.   color: #444;
25.   background-color: transparent;
26.   border-bottom: 1px solid #D0D0D0;
27.   font-size: 16px;
28.   font-weight: bold;
29.   margin: 24px 0 2px 0;
30.   padding: 5px 0 6px 0;
31. }
32.
33. </style>
34. </head>
35. <body>
36.
37. <h1>Perkalian!</h1>
38. <?php echo validation_errors();?>
39. <p>Silakan masukan data berikut!!</p>
40. <?php echo form_open('hitung/perkalian');?>
41. <?php echo form_input('v1',$v1);?> x
42. <?php echo form_input('v2',$v2);?><br>
43.
44. <?php echo form_submit('submit','Hitung!!');?>
45. <?php echo form_close();?><br>
46. Hasil : <?php echo $hasil;?>
47.
48. <p><br/>Page rendered in {elapsed_time} seconds</p>
49.
50. </body></html>
```

Pada view, kita cukup menambahkan satu fungsi yang berguna untuk menampilkan kesalahan yang terjadi. Perhatikan baris 38. Fungsi **validations\_error** adalah fungsi untuk menampilkan kesalahan dari validasi yang dilakukan.

Gambar 10. Form perkalian dengan tampilan error

Begitu juga untuk view pembagian, kita tinggal menambah fungsi **validations\_error** untuk menampilkan error pada form pembagian. Adapun code yang dapat digunakan adalah

```

1.  <!DOCTYPE html>
2.  <!DOCTYPE html>
3.  <html lang="en">
4.  <head>
5.  <meta charset="utf-8">
6.  <title>Pembagian menggunakan validasi!!</title>
7.
8.  <style type="text/css">
9.
10. body {
11.   background-color: #fff;
12.   margin: 40px;
13.   font-family: Lucida Grande, Verdana, Sans-serif;
14.   font-size: 14px;
15.   color: #4F5155;
16. }
17.
18. a {
19.   color: #003399;
20.   background-color: transparent;
21.   font-weight: normal;
22. }
23.
24. h1 {
25.   color: #444;
26.   background-color: transparent;
27.   border-bottom: 1px solid #D0D0D0;
28.   font-size: 16px;
29.   font-weight: bold;

```

```
30. margin: 24px 0 2px 0;  
31. padding: 5px 0 6px 0;  
32. }  
33.  
34.  
35. </style>  
36. </head>  
37. <body>  
38.  
39. <h1>pembagian!</h1>  
40. <?php echo validation_errors();?>  
41. <p>Silakan masukan data berikut!!</p>  
42. <?php echo form_open('hitung/pembagian');?>  
43. <?php echo form_input('v1',$v1);?> /  
44. <?php echo form_input('v2',$v2);?><br>  
45.  
46. <?php echo form_submit('submit','Hitung!!');?>  
47. <?php echo form_close();?><br>  
48. Hasil : <?php echo $hasil;?>  
49.  
50. <p><br/>Page rendered in {elapsed_time} seconds</p>  
51.  
52. </body></html>
```

View diatas akan memberikan tampilan seperti gambar 11 berikut ini.

The screenshot shows a web page with the title "pembagian!". It displays two validation errors: "The Variabel 1 field must contain a number greater than zero." and "The Variabel 2 field must contain a number greater than zero.". Below the errors is a text input field containing "0" and another input field containing "0". A button labeled "Hitung!!" is visible. The result section shows "Hasil : 0". At the bottom, it says "Page rendered in 0.0316 seconds".

Gambar 11. Form perkalian dengan tampilan error

## Chapter 7

# Codelgniter & Database

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

## Connect ke Database

Codelgniter memiliki sebuah file konfigurasi yang memungkinkan Anda menyimpan konfigurasi untuk melakukan koneksi ke database (username, password, nama database, dan lain-lain). File konfigurasi terletak berada di **application/config/database.php**. Pengaturan konfigurasi disimpan dalam bentuk array multi-dimensi dengan prototipe seperti berikut ini:

```
$db['default']['hostname'] = "localhost";
$db['default']['username'] = "root";
$db['default']['password'] = "";
$db['default']['database'] = "database_name";
$db['default']['dbdriver'] = "mysql";
$db['default']['dbprefix'] = "";
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = FALSE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = "";
$db['default']['char_set'] = "utf8";
$db['default']['dbcollat'] = "utf8_general_ci";
```

Kita dapat memiliki beberapa konfigurasi database. Alasan kita menggunakan array multi-dimensi adalah agar memungkinkan Anda secara opsional menyimpan beberapa set nilai-nilai koneksi. Jika, misalnya, Anda menjalankan aplikasi dibeberapa kondisi lingkungan/*environtment* (*development*, *production*, *testing*, dll) di bawah satu instalasi, maka Anda dapat mengatur kelompok konfigurasi untuk masing-masing *environtment*. Sebagai contoh, Anda sedang menjalankan aplikasi di environment test maka Anda tinggal menambahkan konfigurasi berikut ini dan mengubah default konfigurasinya (mengeset **\$active\_group="test"**) menjadi test:

```
$db['test']['hostname'] = "localhost";
$db['test']['username'] = "root";
$db['test']['password'] = "";
$db['test']['database'] = "database_name";
$db['test']['dbdriver'] = "mysql";
$db['test']['dbprefix'] = "";
$db['test']['pconnect'] = TRUE;
$db['test']['db_debug'] = FALSE;
$db['test']['cache_on'] = FALSE;
```

```
$db['test']['cachedir']="";  
$db['test']['char_set']="utf8";  
$db['test']['dbcollat']="utf8_general_ci";  
$active_group="test";
```

Untuk connect ke database ada beberapa cara yang disediakan oleh CodeIgniter diantaranya :

## 1. Menambahkan Database Library Sebagai Autoload Library

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

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

Ketika menambahkan autoload pastikan bahwa database sudah terkonfigurasi dengan benar untuk menghindari kesalahan.

## 2. Mengaktifkan Manual Dari Library Database

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

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

Jika fungsi di atas tidak berisi informasi apapun di parameter pertama akan menyambung ke group konfigurasi yang aktif. Untuk memilih kelompok tertentu dari file konfigurasi, Anda dapat melakukan seperti pada contoh berikut. Hal tersebut berguna pada saat menggunakan aplikasi yang memiliki dua database.

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

Group\_name adalah nama grup konfigurasi dari file konfigurasi Anda. Untuk menghubungkan secara manual ke database yang diinginkan, Anda dapat melewatkannya sebuah array nilai:

```
$config['hostname']="localhost";  
$config['username']="myusername";  
$config['password']="mypassword";  
$config['database']="mydatabase";  
$config['dbdriver']="mysql";  
$config['dbprefix']="";  
$config['pconnect']=FALSE;  
$config['db_debug']=TRUE;  
$config['cache_on']=FALSE;  
$config['cachedir']="";  
$config['char_set']="utf8";  
$config['dbcollat']="utf8_general_ci";
```

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

Atau Anda bisa mengirimkan nilai-nilai database Anda sebagai Data Source Name. DSN harus memiliki prototipe seperti ini:

```
$dsn='dbdriver://username:password@hostname/database';
$this->load->database($dsn);
```

### **3. Mengaktifkan manual dari model**

Selain kedua cara di atas Anda juga bisa mengaktifkan database pada saat loading model. Caranya adalah dengan mengeset TRUE pada parameter ketiga load model. Contoh :

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

Selain itu Anda juga bisa menggunakan konfigurasi tertentu untuk sebuah model. Contoh:

```
$manual_config['hostname']="localhost";
$manual_config['username']="myusername";
$manual_config['password']="mypassword";
$manual_config['database']="mydatabase";
$manual_config['dbdriver']="mysql";
$manual_config['dbprefix']="";
$manual_config['pconnect']=FALSE;
$manual_config['db_debug']=TRUE;
$this->load->model('Model_name','','$manual_config');
```

## Codelgniter Model

Model pada Codelgniter adalah sebuah kelas php yang berfungsi untuk menangani data. Ingat data bukan hanya dari database tetapi juga bisa dari File Text, Web Service atau layanan-layanan data lainnya. Contoh Model:

```
class Blogmodel extends Model {  
  
    var$title='';  
    var$content='';  
    var$date='';  
  
    function Blogmodel()  
    {  
        parent::Model();  
    }  
  
    function get_data()  
    {  
        $query=$this->db->get('entries',10);  
        Return $query->result();  
    }  
}
```

Sebuah model sebenarnya tidak harus meng-extend class Model. Kelas model di-extend ketika hendak menggunakan fitur database pada Codelgniter saja. Semua Model harus diletakkan di dalam folder **application/models**. Agar dapat menggunakan model maka kita harus me-load model tersebut. Adapun perintah yang dapat digunakan untuk meload sebuah model adalah

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

Ketika sudah berhasil me-load sebuah model maka model tersebut akan menjadi sebuah property. Melalui property itulah Anda akan menggunakan semua fungsi yang ada di dalam model (perhatikan baris ke 2 pada contoh diatas **get\_data** adalah fungsi didalam kelas Model\_name)

## Melakukan Query pada Database

Untuk mendapatkan data pada database Anda harus melakukan query. Setelah query dilakukan barulah bisa mendapatkan data hasil query tersebut, baik dalam bentuk object maupun array.

```
$query=$this->db->query('QUERY SQL ANDA');
```

Perlu ditekankan bahwa fungsi query di atas belum menghasilkan data apapun. Keluarannya hanya berupa **Object(true)** atau **False**. Ketika keluarannya adalah **False** maka query yang dilakukan berarti gagal. Tetapi jika true atau mengembalikan sebuah object maka query yang

dilakukan berarti berhasil. Dari object tersebut (variabel \$query kalau mengacu contoh di atas) Anda dapat mengambil data yang diinginkan. Contoh:

```
$query=$this->db->query('SELECT name, title, email FROM my_table');

foreach($query->result()as $row)
{
    echo $row->title;
    echo $row->name;
    echo $row->email;
}

echo'Total Results: ' . $query->num_rows();
```

Contoh di atas adalah salah satu contoh penggunaan query (tapi ingat, pastikan Anda sudah memiliki konfigurasi database yang benar dan telah me-load library database. Perhatikan **\$query->result()**. Itu adalah contoh syntax untuk mendapatkan hasil query dalam bentuk object, Anda juga dapat menggunakan **\$query->result\_array()** untuk mendapatkan hasil query dalam bentuk array assosiatif.

## Query Return Value

Seperti yang telah disebutkan di atas, setelah query dilakukan kita perlu memanggil fungsi tertentu untuk mendapatkan hasil dari query. Secara umum ada dua jenis tipe data yang dapat dihasilkan yaitu **array** dan **object**. Dari segi jumlah kita hanya dapat mengambil satu record (per record) atau keseluruhan record.

- **Mengembalikan Hasil Query Sebagai Kumpulan Array.**

Sebelum mengambil hasil query, ada baiknya mengecek terlebih dahulu apakah query tersebut memiliki hasil atau tidak. Perhatikan fungsi **\$result->result\_array()**, itu adalah method yang digunakan untuk mengambil hasil query. Sedangkan fungsi **\$result->num\_rows()** digunakan untuk mengetahui berapa jumlah record yang didapat.

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

if($result->num_rows()>0)
{
    foreach($result->result_array()as $row)
    {
        echo $row['username'];
        echo $row['email'];
    }
}
```

- **Mengembalikan Hasil Query Sebagai Kumpulan Object.**

Fungsi yang digunakan hampir sama dengan mengembalikan nilai query sebagai array. Cukup dengan memanggil method **\$result->result()**.

```
$result=$this->db->query( 'SELECT * FROM users' );
if($result->num_rows()>0)
{
    foreach($result->result()as $row)
    {
        echo $row->username;
        echo $row->email;
    }
}
```

- **Mengembalikan Hasil Query Sebagai Row Array**

Jadi kita hanya mengambil sebuah record dari sebuah query. Bukan seluruh hasil query.

```
$result=$this->db->query( 'SELECT * FROM users' );
if($query->num_rows()>0)
{
    $row=$query->row();
    echo $row[ 'username' ];
    echo $row[ 'email' ];
}
```

- **Mengembalikan Hasil Query Sebagai Row Object**

Jadi kita hanya mengambil sebuah record dari sebuah query sebagai object. Bukan seluruh hasil query.

```
$result=$this->db->query( 'SELECT * FROM users' );
if($query->num_rows()>0)
{
    $row=$query->row();
    echo $row->username;
    echo $row->email;
}
```

## Menggunakan Active Record

Active Record (AR) adalah sebuah pattern / pendekatan untuk membaca data dari sebuah table atau view dengan cara membungkusnya dalam sebuah kelas. Sehingga tidak dibutuhkan SQL jika kita menggunakan active record. Keuntungan yang didapat adalah kesederhanaan dalam pengkodean program dan fleksibilitasnya ketika terjadi pergantian skema atau pun jenis database (bebas dari sintaks-sintaks SQL yang bersifat khusus ke database tertentu).

Codelgniter tidak sepenuhnya meniru pattern Active Record, tetapi ia melakukan sedikit modifikasi terhadap pattern tersebut. Dengan menggunakan Active Record CI maka proses pengambilan data, insert, update dan delete menjadi lebih sederhana.

```
$this->db->get('users')
```

Query di atas akan menghasilkan query “SELECT \* FROM users” dan setara dengan

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

Ketika kita tidak menggunakan fungsi apapun maka fungsi get akan menghasilkan seluruh record yang berada di dalam tabel tersebut, tetapi jika diberi fungsi select(), where(), dan lain-lain maka hasilnya akan berbeda.

## Selecting Data

Untuk mengambil data dari database dengan field-field tertentu kita dapat menggunakan perintah select(). Contoh:

```
$this->db->select('username,password,email');
$this->db->get('users');
```

Query di atas setara dengan

```
$this->db->query('SELECT username,password,email FROM users');
```

Selain method select, terdapat juga method-method lainnya seperti **from()**, **where()**, **where\_in()**, **group\_by**, **order\_by**, **like**, **distinct()** dan lain-lain. Semua method tersebut jika dipanggil akan membentuk sebuah query dan query tersebut akan dijalankan ketika method get() dipanggil. Untuk kemudahan, active record sudah memiliki fitur chaining method, jadi kita dapat merangkai semua query tadi sehingga menjadi lebih ringkas. Contoh query yang kompleks

```
$data_all=$this->db
->select("sum(budget) as sum_budget, site_ad_mapping.*")
->where('site_ad_mapping.site_id',$site_id)
->where('ad_status','approved')
->where('request_status','active')
```

```
->where('ad_operational_status','active')

->join('ads','ads.id_ad =site_ad_mapping.ad_id ')
->join('advertisers','id_advertiser=advertiser_id')
->join('site_counter','site_counter.site_id =site_id ')
->join('sites','site_counter.site_id = sites.id_site ')
->join('publishers','sites.publisher_id = id_publisher')

->from('site_ad_mapping')

->order_by('ad_priority_level','desc')
->order_by('point','desc')

->group_by('ad_id')->limit(0,10)
->get()->result();
```

## Insert Data

Selain pengambilan data, Active record juga menyediakan fungsionalitas untuk insert data. Untuk menginsert sebuah data kita hanya memerlukan dua parameter yaitu nama table dan data yang akan dimasukkan. Data yang akan dimasukkan harus memiliki format berupa array assosiatif ataupun object. Adapun keuntungan menggunakan active record untuk menginsert data selain sederhana untuk digunakan, adalah method ini secara build-in disertai fungsi untuk penanganan escaping, jadi query kita sudah bisa dikatakan aman dari kesalahan, contoh:

```
$data=array('username'=>'Ibnoe','email'=>'xibnoe@gmail.com');
$this->db->insert('users',$data);
```

Selain dengan cara di atas, kita juga bisa memasukkan data yang diinginkan satu persatu. Contohnya:

```
$data=array('username'=>'Ibnoe','email'=>'xibnoe@gmail.com');
$this->db->set($data);
$this->db->insert('users');
```

Atau

```
$this->db->set('username','ibnoe');
$this->db->set('email','xibnoe@gmail.com');
$this->db->insert('users');
```

Kedua Contoh di atas menghasilkan query yang sama.

## Update Data

Selain insert dan select data, CodeIgniter juga menyediakan fungsi untuk update. Adapun penggunaannya mirip seperti penggunaan insert, contoh:

```
$this->db->set( 'username' , 'ibnoe' );
$this->db->set( 'password' , '123456' );
$this->db->update( 'users' );
```

Query di atas akan menghasilkan query “update users set username=ibnoe, password=123456”.

Cara yang lebih sederhana dengan menggunakan array assosiatif

```
$data=array( 'username'=>'ibnoe' , 'password'=>'123456' );
$this->db->where('id',5);
$this->db->update('users',$data);
```

## Delete Data

Untuk melakukan delete pada CodeIgniter, kita dapat menggunakan perintah berikut ini:

```
$this->db->where('id',5);
$this->db->delete('table_name');
```



## Chapter 8

# Kasus 2. CRUD dan Pagination Database

Setelah mengetahui cara menggunakan database maka kita akan mencoba melakukan perintah dasar pada sistem informasi CRUD (create, read, update dan delete) data menggunakan CodeIgniter. Selain CRUD, pada contoh ini juga akan dilakukan proses *sorting* dan *pagination*. Studi kasus yang akan diangkat adalah form data siswa.

Pada bab ini aplikasi CRUD yang akan dibuat adalah aplikasi yang memiliki fungsi untuk menampilkan seluruh data siswa. Data yang ditampilkan memiliki **pagination** (pembagian jumlah record yang ditampilkan pada sebuah halaman website). Pagination dibutuhkan karena kita tidak mungkin menampilkan semua data dalam satu layar. Selain itu daftar siswa tadi akan memiliki fitur pengurutan berdasarkan field dan memiliki menu action per daftar siswa.

Untuk melakukan pagination di Codeigniter telah disediakan sebuah kelas yaitu kelas pagination. Untuk menggunakan kelas ini minimal harus memiliki tiga data yang selanjutnya akan menjadi konfigurasi pada library ini. Ketiga data tersebut adalah pagination base\_url, total\_row (jumlah total baris atau record yang kita punya ) dan per\_page (jumlah baris yang akan ditampilkan perhalaman). Contohnya

```
$this->load->library('pagination');
$this->load->helper('url');
$config['base_url'] = site_url('results/page/');
$config['total_rows'] = 200;
$config['per_page'] = 20;
$this->pagination->initialize($config);
echo $this->pagination->create_links();
```

Contoh diatas menunjukkan cara penggunaan library pagination. Output dari library ini adalah link-link yang terdiri atas halaman dan link ke halaman berikut (next) atau sebelumnya (prev). Base\_url harus berisi sebuah fungsi dari controller. Fungsi tersebut akan menerima dua parameter yaitu jumlah record perpage dan record berapa yang tampil. Selain parameter-parameter diatas masih banyak parameter lainnya yang dapat digunakan untuk mengkustomisasi library pagination. Adapun yang menjadi favorit penulis adalah

```
//menempatkan informasi record pada uri ke 3
$config['uri_segment'] = 3;
//mengganti tulisan next menjadi Berikutnya >
$config['next_link'] = 'Berikutnya&gt;';
//mengganti tulisan prev menjadi < sebelumnya
$config['prev_link'] = '&lt; sebelumnya';
```

Adapun langkah-langkah yang harus dilakukan untuk membuat CRUD dan pagination adalah

## 1. Membuat Database Dan Table Data Siswa

Untuk memulai, Anda harus memiliki sebuah database. Selain database, Anda harus membuat sebuah tabel siswa. Adapun table yang harus dibuat adalah

```
CREATE TABLE IF NOT EXISTS `siswa` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `nama` varchar(50) DEFAULT NULL,
  `alamat` varchar(200) NOT NULL,
  `jenis_kelamin` char(1) DEFAULT NULL,
  `tanggal_lahir` date DEFAULT NULL,
  PRIMARY KEY (`id`),
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;
```

Tabel siswa di atas memiliki lima field yaitu id siswa, nama siswa, alamat siswa, jenis kelamin, dan tanggal lahir. Tabel ini digunakan untuk menyimpan data siswa.

## 2. Konfigurasi CodeIgniter

Sebelum memulai pengkodean, sebaiknya kita melakukan konfigurasi terhadap tools yang akan digunakan. Ada tiga konfigurasi yang harus disesuaikan, antara lain: **config.php**, **database.php** dan **router.php**. Adapun yang harus dilakukan adalah:

- **config.php** - pada file configurasi ini yang perlu diubah adalah **base\_url**. Konfigurasi ini berfungsi untuk menentukan path dasar dari aplikasi. Konfigurasi ini akan digunakan oleh helper form dan url. Contoh perubahan yang dilakukan:

```
$config['base_url']='http://localhost/crud'; //ganti sesuai path  
diserver Anda
```

- **database.php** - lakukan konfigurasi di sini supaya aplikasi dapat terkoneksi ke database.

```
$db['default']['hostname']='localhost';
$db['default']['username']='root';
$db['default']['password']='';
$db['default']['database']='crud';
$db['default']['dbdriver']='mysql';
```

- **router.php** - berisi konfigurasi mengenai default controller dan routing rule. Pada konfigurasi ini yang diubah hanya default controller

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

### 3. Membuat Model Siswa

Model siswa ini adalah model yang bertugas dan bertanggung jawab untuk melakukan proses-proses yang berhubungan dengan database. Misalnya menyimpan, mengubah, menghapus serta mengambil data dari database.

Biasanya untuk memudahkan kita, pembuatan model mengikuti desain database atau dapat diartikan satu tabel dapat diwakili oleh satu model. Model tersebutlah yang bertanggung jawab pada semua operasi pada tabel tersebut.

Pada aplikasi CRUD ini kita akan membuat sebuah model siswa. Adapun isi dari model siswa adalah

```

1. <?php
2.
3. class Siswa_model extends CI_Model {
4.
5.     private $primary_key='id';
6.     private $table_name='siswa';
7.
8.     function __construct(){
9.         parent::__construct();
10.    }
11.
12.    function get_paged_list($limit=10,$offset=0,
13. $order_column='', $order_type='asc')
14.    {
15.        if (empty($order_column) || empty($order_type))
16.            $this->db->order_by($this->primary_key, 'asc');
17.        else
18.            $this->db->order_by($order_column, $order_type);
19.        return $this->db->get($this->table_name, $limit, $offset);
20.    }
21.
22.    function count_all(){
23.        return $this->db->count_all($this->table_name);
24.    }
25.
26.    function get_by_id($id){
27.        $this->db->where($this->primary_key, $id);
28.        return $this->db->get($this->table_name);
29.    }
30.
31.    function save($person){
32.        $this->db->insert($this->table_name, $person);
33.        return $this->db->insert_id();
34.    }
35.
36.    function update($id, $person){
37.        $this->db->where($this->primary_key, $id);
38.        $this->db->update($this->table_name, $person);
39.    }
40.

```

```
41. function delete($id){  
42.     $this->db->where($this->primary_key,$id);  
43.     $this->db->delete($this->table_name);  
44. }  
45. }
```

#### 4. Membuat Controller Dan View Data Siswa

Controller berisi logika untuk melakukan proses-proses yang ada. Adapun proses yang akan ditangani oleh controller ini adalah menampilkan daftar siswa, menambah siswa, mengubah data siswa, menghapus data siswa dan melihat detail siswa.

```
1.  <?php  
2.  class Siswa extends CI_Controller {  
3.  
4.  private $limit=10;  
5.  
6.  function __construct()  
7.  {  
8.      parent::__construct();  
9.      #load library dan helper yang dibutuhkan  
10.     $this->load->library(array('table','form_validation'));  
11.     $this->load->helper(array('form','url'));  
12.     $this->load->model('siswa_model','','TRUE');  
13.  }  
14.  
15. function index($offset=0,$order_column='id',  
16. $order_type='asc')  
17. {  
18.     if (empty($offset)) $offset=0;  
19.     if (empty($order_column)) $order_column='id';  
20.     if (empty($order_type)) $order_type='asc';  
21.     //TODO: check for valid column  
22.  
23.     // load data siswa  
24.     $siswas=$this->siswa_model->get_paged_list($this->limit,  
25.     $offset,$order_column,$order_type)->result();  
26.  
27.     // generate pagination  
28.     $this->load->library('pagination');  
29.     $config['base_url']= site_url('siswa/index/');  
30.     $config['total_rows']=$this->siswa_model->count_all();  
31.     $config['per_page']=$this->limit;  
32.     $config['uri_segment']=3;  
33.     $this->pagination->initialize($config);  
34.     $data['pagination']=$this->pagination->create_links();  
35.  
36.     // generate table data  
37.     $this->load->library('table');  
38.     $this->table->set_empty(" ");  
39.     $new_order=($order_type=='asc'?'desc':'asc');  
40.     $this->table->set_heading(  
41.         'No',  
42.         anchor('siswa/index/'.$offset.'/'.$new_order,'Nama'),  
43.         anchor('siswa/index/'.$offset.'/'.$new_order,'Alamat'),
```

```

44.     anchor('siswa/index/'.$offset.'/jenis_kelamin/'.$new_order,
45.     'Jenis Kelamin'),
46.     anchor('siswa/index/'.$offset.'/tanggal_lahir/'.$new_order,
47.     'Tanggal Lahir (dd-mmm-yyyy)'),
48.     'Actions'
49. );
50. $i=0+$offset;
51. foreach ($siswas as $siswa){
52.     $this->table->add_row(++$i,
53.     $siswa->nama,
54.     $siswa->alamat,
55.     strtoupper($siswa->jenis_kelamin)=='M'?
56.     'Laki-Laki':'Perempuan',
57.     date('d-m-Y',strtotime(
58.     $siswa->tanggal_lahir)),
59.     anchor('siswa/view/'.$siswa->id,
60.     'view',array('class'=>'view')).' '.
61.     anchor('siswa/update/'.$siswa->id,
62.     'update',array('class'=>'update')).' '.
63.     anchor('siswa/delete/'.$siswa->id,
64.     'delete',array('class'=>'delete',
65.     'onclick'=>"return confirm(
66.     'Apakah Anda yakin ingin menghapus
67.     data siswa?')"))
68. );
69. }
70. $data['table']=$this->table->generate();
71.
72. if ($this->uri->segment(3)=='delete_success')
73.     $data['message']='Data berhasil dihapus';
74. else if ($this->uri->segment(3)=='add_success')
75.     $data['message']='Data berhasil ditambah';
76. else
77.     $data['message']='';
78. // load view
79. $this->load->view('siswaList',$data);
80. }
81.
82.
83. function add(){
84.     // set common properties
85.     $data['title']='Tambah siswa baru';
86.     $data['action']= site_url('siswa/add');
87.     $data['link_back']= anchor('siswa/index/',
88.     'Back to list of siswas',array('class'=>'back'));
89.
90.     $this->_set_rules();
91.
92.     // run validation
93.     if ($this->form_validation->run()=== FALSE){
94.         $data['message']='';
95.         // set common properties
96.         $data['title']='Add new siswa';
97.         $data['message']='';
98.         $data['siswa']['id']='';
99.         $data['siswa']['nama']='';
100.        $data['siswa']['alamat']='';

```

```
101.     $data['siswa']['jenis_kelamin']='';
102.     $data['siswa']['tanggal_lahir']='';
103.     $data['link_back']= anchor('siswa/index/',
104.         'Lihat Daftar Siswa',array('class'=>'back'));
105.
106.     $this->load->view('siswaEdit',$data);
107.
108. } else {
109. // save data
110. $siswa= array( 'nama'=>$this->input->post('nama'),
111. 'alamat'=>$this->input->post('alamat'),
112. 'jenis_kelamin'=>$this->input->post('jenis_kelamin'),
113. 'tanggal_lahir'=> date('Y-m-d',
114. strtotime($this->input->post('tanggal_lahir'))));
115. $id=$this->siswa_model->save($siswa);
116.
117. // set form input nama="id"
118. $this->validation->id =$id;
119.
120. redirect('siswa/index/add_success');
121. }
122. }
123.
124. function view($id){
125. // set common properties
126. $data['title']='siswa Details';
127. $data['link_back']= anchor('siswa/index/',
128. 'Lihat daftar siswas',array('class'=>'back'));
129.
130. // get siswa details
131. $data['siswa']=$this->siswa_model->get_by_id($id)->row();
132.
133. // load view
134. $this->load->view('siswaView',$data);
135. }
136.
137. function update($id){
138. // set common properties
139. $data['title']='Update siswa';
140. $this->load->library('form_validation');
141. // set validation properties
142. $this->_set_rules();
143. $data['action']=('siswa/update/'.$id);
144.
145. // run validation
146. if ($this->form_validation->run() === FALSE){
147.
148. $data['message']='';
149. $data['siswa']=$this->siswa_model->get_by_id($id)->row_array();
150. $_POST['jenis_kelamin']=  

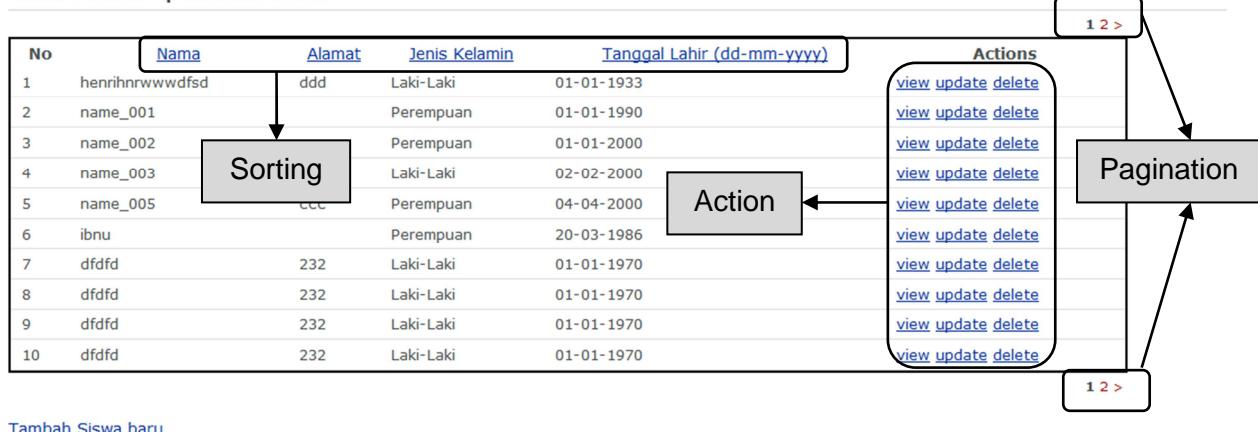
151. strtoupper($data['siswa']['jenis_kelamin']);
152. $data['siswa']['tanggal_lahir']= date('d-m-Y',
153. strtotime($data['siswa']['tanggal_lahir']));
154.
155. // set common properties
156. $data['title']='Update siswa';
157. $data['message']='';
```

```

158.
159.     } else {
160.         // save data
161.         $id=$this->input->post('id');
162.         $siswa= array('nama'=>$this->input->post('nama'),
163.           'alamat'=>$this->input->post('alamat'),
164.           'jenis_kelamin'=>$this->input->post('jenis_kelamin'),
165.           'tanggal_lahir'=> date('Y-m-d',
166.             strtotime($this->input->post('tanggal_lahir'))));
167.         $this->siswa_model->update($id,$siswa);
168.         $data['siswa']=$this->siswa_model->get_by_id($id)-
>row_array();
169.         // set user message
170.         $data['message']='update siswa success';
171.     }
172.     $data['link_back']= anchor('siswa/index/',
173.       'Lihat daftar siswa',array('class'=>'back'));
174.     // load view
175.     $this->load->view('siswaEdit',$data);
176.   }
177.
178.   function delete($id){
179.     // delete siswa
180.     $this->siswa_model->delete($id);
181.     // redirect to siswa list page
182.     redirect('siswa/index/delete_success','refresh');
183.   }
184.
185.   // validation rules
186.   function _set_rules(){
187.
188.     $this->form_validation->set_rules( 'nama' , 'Nama' ,
189.       'required|trim');
190.     $this->form_validation->set_rules( 'jenis_kelamin' , 'Password' ,
191.       'required');
192.     $this->form_validation->set_rules( 'alamat' , 'Alamat' ,
193.       'required|callback_valid_date');
194.     $this->form_validation->set_rules( 'tanggal_lahir' , 'Tanggal
195.       Lahir' , 'required');
196.
197.   }
198.
199.   // date_validation callback
200.   function valid_date($str)
201.   {
202.     if(!preg_match('/^([0-9]{4})-([0-9]{2})-([0-9]{2})$/',$str))
203.     {
204.       $this->form_validation->set_message('valid_date',
205.         'date format is not valid. yyyy-mm-dd');
206.       return false;
207.     }
208.     else
209.     {
210.       return true;
211.     }
212.   }
213. }
```

Perhatikan baris ke-15 pada fungsi `index($offset = 0, $order_column = 'id', $order_type = 'asc')`. Pada fungsi tersebut ada 3 parameter yang bersifat optional yang dapat diberikan melalui URI. Parameter `$offset` berfungsi untuk menentukan record pertama yang akan ditampilkan, `$order_column` berfungsi untuk menentukan field apa yang akan digunakan untuk mengurutkan data siswa dan `$order_type` berfungsi untuk menentukan jenis urutan (descending atau ascending). Untuk mendapatkan data yang sesuai dengan persyaratan (pengurutan dan pagination) maka dipanggil `get_paged_list` (baris 24). Setelah data siswa didapatkan maka akan digunakan library table untuk menampilkan data dalam bentuk tabel seperti tampilan di bawah ini (baris 38-70).

Contoh Insert Update dan delete



Untuk mendapatkan tampilan seperti diatas maka perlu dibuat sebuah view seperti berikut ini. View ini disimpan dalam sebuah file bernama siswaList.php

```

1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2.  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3.  <html xmlns="http://www.w3.org/1999/xhtml">
4.  <head>
5.  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
6.  <title>SIMPLE CRUD APPLICATION</title>
7.  <link href="<?php echo base_url();?>style/style.css" 
8.  rel="stylesheet" type="text/css" />
9.  </head>
10. <body>
11. <div class="content">
12. <h1>Contoh Insert Update dan delete</h1>
13. <div class="paging"><?php echo $pagination;?></div>
14. <div class="data"><?php echo $table;?></div>
15. <div class="paging"><?php echo $pagination;?></div><br />
16. <?php echo anchor('siswa/add',
17. 'Tambah Siswa baru', array('class'=>'add')) ;?>
18. </div>
19. </body>
20. </html>

```

Selain untuk menampilkan data, controller tadi memiliki fungsi ubah dan tambah data siswa. Kedua fungsi tersebut pada intinya sama. Perbedaannya adalah action atau fungsi model yang dipanggil dan pemanggilan record yang akan di edit. Perhatikan fungsi add pada baris 82. Pada fungsi add tersebut dilakukan pemanggilan terhadap fungsi `$this->_set_rules();` fungsi tersebut digunakan untuk mengeset validation rule. Jika inputan yang dimasukkan oleh user sesuai dengan rule validasi maka akan dipanggil fungsi `$this->siswa_model->save` untuk menyimpan data tersebut. Adapun view yang digunakan adalah

```

1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2.  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3.  <html xmlns="http://www.w3.org/1999/xhtml">
4.  <head>
5.  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
6.  <title>SIMPLE CRUD APPLICATION</title>
7.  <link href="<?php echo base_url();?>style/style.css" 
8.  rel="stylesheet" type="text/css" />
9.  </head>
10. <body>
11. <div class="content">
12. <h1><?php echo $title;?></h1>
13. <?php echo $message;?>
14. <?php echo validation_errors();?>
15. <?php echo form_open($action);?>
16. <div class="data">
17. <table>
18. <tr>
19. <td width="30%">ID</td>
20. <td><input type="text" name="id" disabled="disabled" class="text" 
21. value="<?php echo (isset($siswa['id']))?$siswa['id']:'';?>" /></td>
22. <input type="hidden" name="id" value="<?php echo
23. (isset($siswa['id']))?$siswa['id']:'';?>" />
24. </tr>
25. <tr>
26. <td align="top">nama<span style="color:red;">*</span></td>
27. <td><input type="text" name="nama" class="text" value="<?php echo
28. (set_value('nama'))?set_value('nama'):$siswa['nama'];?>" />
29. <?php echo form_error('nama');?></td>
30. </tr>
31. <tr>
32. <td align="top">Alamat</td>
33. <td><input type="text" name="alamat" class="text" value="<?php echo
34. set_value('alamat')?set_value('alamat'):$siswa['alamat'];?>" />
35. <?php echo form_error('alamat');?></td>
36. </tr>
37. <tr>
38. <td align="top">jenis_kelamin<span style="color:red;">*</span></td>
39. <td><input type="radio" name="jenis_kelamin" value="M" <?php echo
40. set_radio('jenis_kelamin', 'M', TRUE);?>/> Laki-Laki
41. <input type="radio" name="jenis_kelamin" value="F" <?php echo
42. set_radio('jenis_kelamin', 'F');?>/> Perempuan
43. <?php echo form_error('jenis_kelamin');?></td>
44. </tr>
45. <tr>
46. <td align="top">Date of birth (dd-mm-yyyy)<span
47. style="color:red;">*</span></td>
48. <td><input type="text" name="tanggal_lahir" class="text" 
49. value="<?php echo (set_value('tanggal_lahir'))?
50. set_value('tanggal_lahir'):$siswa['tanggal_lahir'];?>" />
```

```
51. <?php echo form_error('tanggal_lahir');?></td>
52. </tr>
53. <tr>
54. <td>&nbsp;</td>
55. <td><input type="submit" value="Save" /></td>
56. </tr>
57. </table>
58. </div>
59. </form>
60. <br/>
61. <?php echo $link_back;?>
62. </div>
63. </body>
64. </html>
```

View di atas juga digunakan oleh fungsi edit data siswa. Untuk melakukan penyimpanan maka perlu dipanggil fungsi **\$this->siswa\_model->update(\$id,\$siswa);** pada fungsi tersebut perlu menyertakan id siswa serta data siswa yang telah terupdate.

#### Add new siswa

The form consists of several input fields:

- ID: A text input field.
- nama\*: A text input field.
- Alamat: A text input field.
- Jenis Kelamin\*: Radio buttons for Laki-Laki (selected) and Perempuan.
- Date of birth (dd-mm-yyyy)\*: A text input field.
- Save: A submit button.

[Lihat Daftar Siswa](#)

Selain tambah dan ubah, ada juga fungsi hapus dan melihat detail siswa. Untuk menghapus siswa hanya dibutuhkan id siswa yang ingin dihapus. Sedangkan untuk melihat detail data siswa hal yang perlu dilakukan adalah memanggil data siswa berdasarkan id siswa kemudian ditampilkan ke dalam view berikut ini:

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2. <html xmlns="http://www.w3.org/1999/xhtml">
3. <head>
4. <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
5. <title>SIMPLE CRUD APPLICATION</title>
6. <link href="<?php echo base_url();?>style/style.css" rel="stylesheet"
7. type="text/css"/>
8. </head>
9. <body>
10. <div class="content">
11. <h1><?php echo $title;?></h1>
12. <div class="data">
13. <table>
14. <tr>
15. <td width="30%">ID</td>
16. <td><?php echo $siswa->id;?></td>
17. </tr>
18. <tr>
19. <td valign="top">Name</td>
```

```

20. <td><?php echo $siswa->name;?></td>
21. </tr>
22. <tr>
23. <td valign="top">Alamat</td>
24. <td><?php echo $siswa->alamat;?></td>
25. </tr>
26. <tr>
27. <td valign="top">Jenis Kelamin</td>
28. <td><?php echo ($siswa->jenis_kelamin)=='M' ?
29. 'Laki-laki':'Perempuan';?></td>
30. </tr>
31. <tr>
32. <td valign="top">Tanggal Lahir (dd-mm-yyyy)</td>
33. <td><?php echo date('d-m-Y', strtotime(
34. $siswa->tanggal_lahir));?></td>
35. </tr>
36. </table>
37. </div>
38. <br/>
39. <?php echo $link_back;?>
40. </div>
41. </body>
42. </html>

```

**siswa Details**

ID	1
Nama Siswa	henrihnrrwwdfs
Alamat	ddd
Jenis Kelamin	Laki-laki
Tanggal Lahir (dd-mm-yyyy)	01-01-1933

[Back to list of siswas](#)

Sebelum melakukan delete perlu dilakukan konfirmasi untuk mencegah kesalahan user dalam menekan link action.

Contoh Insert Update dan delete

No	Nama	Alamat	Jenis Kelamin	Tanggal Lahir (dd-mm-yyyy)	Actions
1	henrihnrrwwdfs	ddd	Laki-Laki	01-01-1933	<a href="#">view</a> <a href="#">update</a> <a href="#">delete</a>
2	name_001		Perempuan		<a href="#">view</a> <a href="#">update</a> <a href="#">delete</a>
3	name_002		Perempuan		<a href="#">view</a> <a href="#">update</a> <a href="#">delete</a>
4	name_003		Laki-Laki		<a href="#">view</a> <a href="#">update</a> <a href="#">delete</a>
5	name_005	ccc	Perempuan		<a href="#">view</a> <a href="#">update</a> <a href="#">delete</a>
6	ibnu		Perempuan		<a href="#">view</a> <a href="#">update</a> <a href="#">delete</a>
7	dfdfd	232	Laki-Laki	01-01-1970	<a href="#">view</a> <a href="#">update</a> <a href="#">delete</a>
8	dfdfd	232	Laki-Laki	01-01-1970	<a href="#">view</a> <a href="#">update</a> <a href="#">delete</a>
9	dfdfd	232	Laki-Laki	01-01-1970	<a href="#">view</a> <a href="#">update</a> <a href="#">delete</a>
10	dfdfd	232	Laki-Laki	01-01-1970	<a href="#">view</a> <a href="#">update</a> <a href="#">delete</a>

Apakah anda yakin ingin menghapus datasiswa?

1 2 >

## Chapter 10

# Kasus 3. Sistem Templating

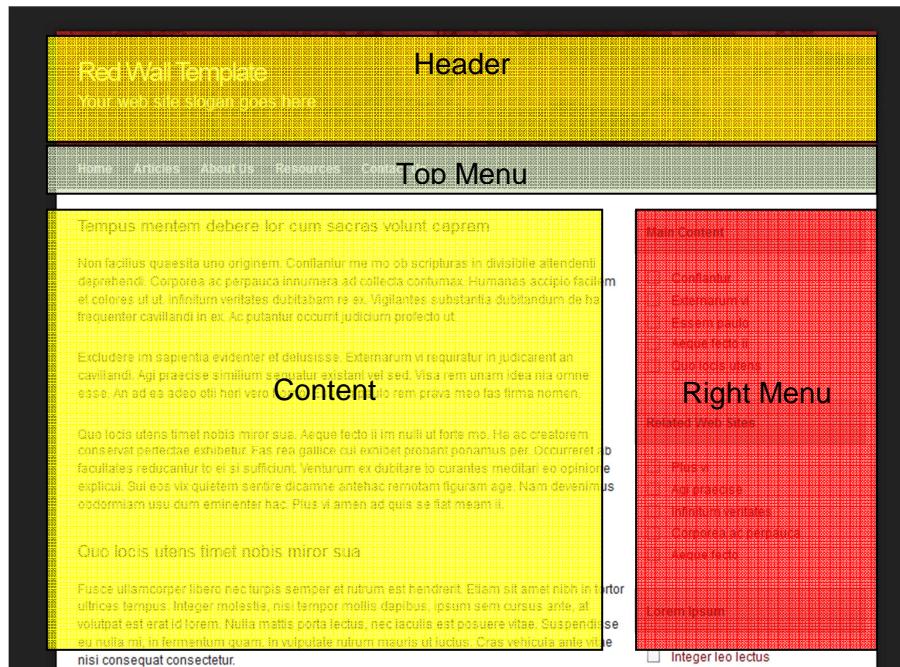
Sistem templating adalah salah satu fitur atau library yang harus dipikirkan di dalam CodeIgniter, karena CodeIgniter memiliki metode yang sangat sederhana. Tidak memiliki sistem layout, widget dan lain-lain. Untuk membuat sistem templating di CodeIgniter kita dapat membuat sendiri dengan menggunakan fungsi-fungsi yang telah disediakan oleh CodeIgniter atau kita juga dapat menggunakan sistem templating yang sudah ada dan populer. Kegunaan sistem templating adalah :

- **Kerja Sama Tim Yang Lebih Baik** - Dengan pemisahan menggunakan templating ini, maka rekan bagian penggeraan desain tidak lagi terganggu oleh kode program yang Anda buat. Begitupun Anda, tidak khawatir lagi script/kode program yang telah dibuat akan terganggu.
- **Skrip/code Yang Bersih** - Situs dengan aplikasi kompleks, misalnya sebuah portal, pasti memerlukan skrip yang kompleks yang menghabiskan sampai ribuan baris. Tentunya akan sangat mengganggu sekali kalau skrip yang sudah memusingkan itu ditambah lagi dengan tag-tag HTML di dalamnya.
- **Perubahan Tampilan Lebih Cepat Dan Mudah** - Dengan pemisahan melalui template, hal tersebut dapat dilakukan dengan mudah, bahkan tanpa harus merombak skrip PHP sedikit pun.

## Native CodeIgniter Tempating

Dengan menyusun view-view yang ada, kita sebenarnya dapat membuat sebuah template library yang cukup powerful. Ideanya sederhana saja, cukup membagi sebuah halaman menjadi beberapa area. Sebagai contoh kita akan menggunakan template dari opendesign.org. Dari desain tersebut dapat kita bagi menjadi area header, top menu, right menu dan content. Masing-masing area tersebut akan digabungkan dalam satu halaman utuh yang dinamakan template.

Masing-masing area tersebut akan ditangani oleh sebuah view agar tidak terjadi duplikasi dan akan mempermudah penggunaan kembali area tersebut. Adapun area-area tersebut dapat dilihat pada gambar berikut.



Template yang akan dibuat terbagi menjadi empat bagian yaitu Header (berfungsi sebagai tempat logo dan slogan aplikasi ), Top menu (bagian menu utama disebelah atas), Right Menu (menu navigasi tambahan disebelah kanan) dan sebuah Content. Area content ini lah yang seringkali berubah pada setiap page.

Untuk mengimplementasikan sistem templating tersebut sebaiknya kita buat sebuah library template. Adapun langkah-langkah yang harus dilakukan adalah:

## 1. Membuat Library Template

Fungsi library tersebut hanya mengatur view mana yang akan dipanggil dan meletakkannya di dalam template. Kenapa membuatnya dalam sebuah library? Karena dengan cara ini memberikan fleksibilitas terhadap sistem template. Cara pembuatan library ini diawali dengan membuat sebuah file bernama **template.php** pada folder **application/libraries**. Adapun isi file template.php adalah:

```

1.  <?php
2.  class Template {
3.  protected $_ci;
4.  function __construct()
5.  {
6.      $this->_ci =&get_instance();
7.  }
8.
9.  function display($template,$data=null)
10. {
11.     $data['_content']=$this->_ci->load->view(
12.     $template,$data, true);
13.     $data['_header']=$this->_ci->load->view(
14.     'template/header', $data, true);

```

```
15.     $data['_top_menu']=$this->ci->load->view(
16.         'template/menu' ,$data, true);
17.     $data['_right_menu']=$this->ci->load->view(
18.         'template/sidebar' ,$data, true);
19.     $this->ci->load->view('/template.php' ,$data);
20. }
21. }
```

Perhatikan fungsi display, disana ada dua parameter yaitu template dan data. Parameter data berfungsi sebagai data yang akan dikirimkan ke controller. Sedangkan template adalah view yang akan dipanggil untuk ditampilkan sebagai content utama. Pada template ini akan dibagi menjadi empat area yaitu content, header, top menu, dan right menu. Masing-masing area diisi oleh sebuah view (perhatikan baris 11,13,15,17). Masing-masing view tadi di-load dan disimpan dalam memori. Lalu digabungkan kedalam sebuah template (baris 19).

Fungsi yang ada pada library ini sebenarnya dapat kita tambahkan lagi, misalnya untuk keperluan seo, kita ingin memanipulasi title dan meta tag dll. Library diatas merupakan contoh paling sederhana ketika kita ingin membuat sebuah sistem template sendiri.

## 2. Membuat Template Layout View

Sebuah template layout view adalah sebuah view yang akan menggabungkan masing-masing view menjadi suatu bentuk kesatuan. Adapun isi dari template layout adalah:

### Application/view/template.php

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2.  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3.  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4.  <head>
5.  <meta http-equiv="content-type" content="text/html; charset=utf-
8"/>
6.  <link rel="stylesheet" type="text/css" href="<?php echo
base_url();?>public/style.css"/>
7.  <title>Sistem Template CodeIgniter Template</title>
8.  </head>
9.  <body>
10. <div id="wrap">
11. <div id="header">
12. <!--
13.     Area Header
14. -->
15. <?php echo $_header;?>
16. </div>
17. <div id="menu">
18. <!--
19.     Area Top Menu
20. -->
21. <?php echo $_top_menu;?>
22. </div>
23. <div id="contentwrap">
```

```

24. <div id="content">
25. <!--
26. Area content
27. -->
28. <?php echo $_content;?>
29.
30. </div>
31.
32. <div id="sidebar">
33. <!--
34. Area Right Menu
35. -->
36. <?php echo $_right_menu;?>
37. </div>
38. <div style="clear: both;"></div>
39. </div>
40.
41. <div id="footer">
42. <p>Copyright &copy; <a href="#">Ibnoe</a> | Design by
43. <a href="http://www.readcrazyreviews.com">
44. Read Crazy Reviews</a></p>
45. </div>
46.
47. </div>
48.
49. </body>
50. </html>

```

File diatas merupakan penggabung dari semua bagian yang telah dibahas. Perhatikan baris 15, 21, 28, dan 36. Keempat variabel tersebut akan diisi view dari masing-masing bagian. Berikut ini adalah view-view yang menjadi part/area dari template

a. View header

**Application/view/template/header.php**

```

1. <h1><a href="#">Red Wall Template</a></h1>
2. <h2>Your web site slogan goes here</h2>

```

b. View Top menu

**Application/view/template/menu.php**

```

1. <ul>
2. <li><a href="#">Home</a></li>
3. <li><a href="#">Articles</a></li>
4. <li><a href="#">About Us</a></li>
5. <li><a href="#">Resources</a></li>
6. <li><a href="#">Contact Us</a></li>
7. </ul>

```

c. View sidebar menu

**Application/view/template/sidebar.php**

```

1. <h3>Main Content</h3>
2. <ul>
3. <li><a href="#">Conflantur</a></li>

```

```
4. <li><a href="#">Externarum vi</a></li>
5. <li><a href="#">Essem paulo</a></li>
6. <li><a href="#">Aequa factio ii</a></li>
7. <li><a href="#">Quo locis utens</a></li>
8. </ul>
9.
10. <h3>Related Web Sites</h3>
11. <ul>
12. <li><a href="#">Plus vi</a></li>
13. <li><a href="#">Agi praecise</a></li>
14. <li><a href="#">Infinitum veritates</a></li>
15. <li><a href="#">Corporea ac perpaucia</a></li>
16. <li><a href="#">Aequa factio</a></li>
17. </ul>
```

Setelah memiliki view ketiga area tersebut (header, top menu dan sidebar) maka kita siap untuk menggunakan sistem template ini. Selanjutnya adalah membuat sebuah controller. Agar lebih mudah pembuatan controller-nya maka gunakan saja controller default dari CodeIgniter, yaitu dengan sedikit melakukan perubahan kode program pada controller-nya.

#### **Application/controllers/welcome.php**

```
1.  <?php if (! defined('BASEPATH')) exit('No direct script access
      allowed');
2.
3.  class Welcome extends CI_Controller {
4.
5.  function __construct()
6.  {
7.    parent::__construct();
8.    $this->load->library('template');
9.    $this->load->helper('url');
10. }
11.
12. function index()
13. {
14.   $this->template->display('welcome_message');
15. }
16.
17. function contoh_parameter()
18. {
19.   $this->template->display('view_parameter',
20.   array('judul'=>'judul View'));
21. }
22.
23. }
24.
25. /* End of file welcome.php */
26. /* Location: ./application/controllers/welcome.php */
```

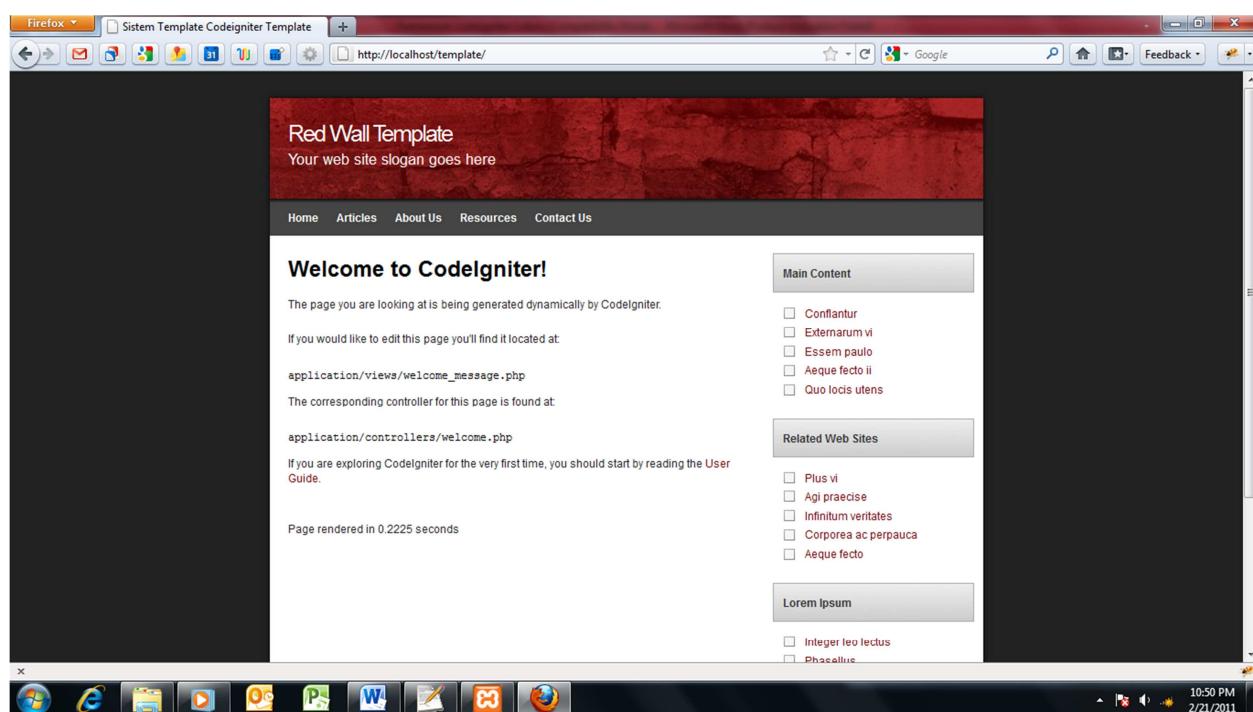
Perhatikan pada baris 8, di situ kita me-load library yang telah dibuat tadi. Pada baris 14 dan 19 kita menggunakan fungsi display untuk menampilkan template. Kita harus menyiapkan sebuah view yang bernama **welcome\_message**. View tersebut akan diletakkan pada area content. Isi viewnya sama dengan view pada umumnya.

```

1. <h1>Welcome to CodeIgniter!</h1>
2.
3. <p>The page you are looking at is being generated dynamically by
CodeIgniter.</p>
4.
5. <p>If you would like to edit this page you'll find it located
at:</p>
6. <code>application/views/welcome_message.php</code>
7.
8. <p>The corresponding controller for this page is found at:</p>
9. <code>application/controllers/welcome.php</code>
10.
11. <p>If you are exploring CodeIgniter for the very first time,
12. you should start by reading the <a href="user_guide/">User
13. Guide</a>.</p>
14.
15.
16. <p><br />Page rendered in {elapsed_time} seconds</p>

```

Adapun tampilannya adalah seperti berikut ini:



## Chapter 10

# Kasus 4. Sistem Authentikasi

Sistem authentikasi atau sistem login merupakan salah satu bagian dari aplikasi yang sering kita kerjakan. Sistem ini juga menjadi bagian yang vital pada aplikasi. Bagian ini lah yang menjamin keamanan data dari aplikasi yang sedang dikerjakan.

Pada kasus ini library yang paling penting untuk di ketahui adalah library session. Perlu diingat bahwa library session codeigniter disimpan di sebuah cookie. Cookie tersebut dapat kita enkripsi. Selain itu kita juga dapat menyimpan session tersebut di database. Yang artinya user cookie harus cocok dengan cookie yang ada di database. Secara default hanya cookie yang digunakan dan walaupun anda tidak menggunakan enkripsi cookie anda harus tetap menkonfigurasi enkripsi key.

Untuk menggunakan library session sama seperti penggunaan library pada biasanya. Kita bisa mengkonfigurasi file autoload atau memanggilnya secara manual

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

Ketika library sudah diload kelas session akan mengecek apakah data session yang diinginkan berada di cookie. Jika data tidak ada dicookie maka akan dibuatkan sebuah session baru dan disimpan didalam cookie. Jika data cookie ditemukan maka data tersebut akan diupdate terutama untuk last\_activity dan session\_id.

Untuk penggunaan library session sendiri sangat mudah. Untuk mengambil data session dapat dilakukan dengan

```
$this->session->userdata('item');
```

Sedangkan untuk menyimpan session dapat dilakukan dengan cara

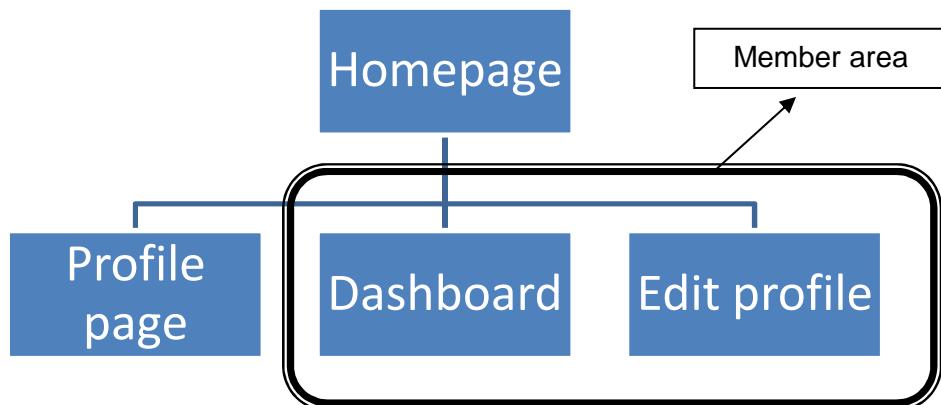
```
$newdata = array(
    'username' => 'ibnoe',
    'email'     => 'xibnoe@gmail.com',
    'logged_in' => TRUE
);
$this->session->set_userdata($newdata);
```

**Catatan:** Data session codeigniter secara default disimpan dalam cookie. Cookie memiliki batasan sebesar 4Kb data. Dengan menggunakan enkripsi maka data yang disimpan akan menjadi lebih panjang. Jadi harap berhati-hati ada kemungkinan data tidak tersimpan. Anda bisa menggunakan alternatif database atau mengextend session ke native session

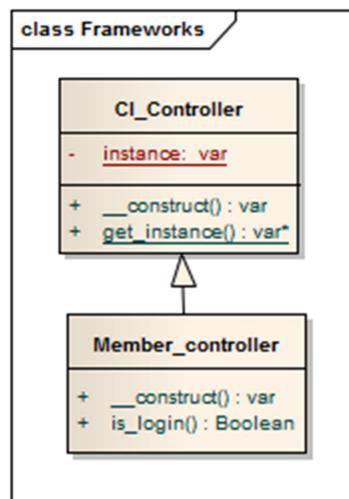
Untuk membuat sebuah sistem authentikasi maka hal pertama yang perlu dilakukan adalah

### 1. Membuat Desain Aplikasi

Desain aplikasi sangat berpengaruh pada code yang akan kita buat. Karena sistem authentikasi ini merupakan salah satu bagian yang kritis maka penulis akan mencoba untuk membuatnya aman bukan hanya dari segi kode, tetapi juga design. Perhatikan struktur website dibawah ini.



Pada gambar diatas ada dua tipe page yaitu page yang bisa diakses oleh semua orang dan page yang hanya boleh diakses oleh member. Hal tersebut sederhana jika kita hanya menghandle dua page tetapi jika pagennya ada banyak maka kita akan mengecek satu-satu informasi user. Hal tersebut kurang aman karena kode kita terduplikasi keseluruh aplikasi. Kita akan mencoba untuk meng-extend controller membuat kasus ini menjadi lebih simple. Untuk controller untuk user yang login harus menggunakan member\_controller.



Jika kita perhatikan class diagram diatas maka semua member\_controller sudah memiliki fungsi untuk pengecekan login secara default.

## 2. Membuat Tabel User

Untuk sistem autentikasi kita akan membuat tiga tabel. Tabel pertama yaitu tabel **user** berisi data informasi login user , tabel **user\_group** berisi data pengelompokan user menjadi group, dan tabel **tracker** berfungsi untuk menyimpan data infromasi darimana user tersebut menggunakan sistem login dan telah berapa kali gagal menggunakannya.

```
CREATE TABLE `users` (
`id` INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`username` VARCHAR( 255 ) NOT NULL ,
`email` VARCHAR( 255 ) NOT NULL ,
`password` VARCHAR( 255 ) NOT NULL
)ENGINE = MYISAM;
```

## 3. Membuat Library Access

Setelah tabel terbentuk barulah kita membuat library untuk mengakses dan mengverifikasi data user. Library tersebut kita sebut Access. Adapun kode programnya adalah sebagai berikut

### application/libraries/access.php

```
1. <?php if (!defined('BASEPATH')) exit('No direct script access
   allowed');
2.
3. class Access
4. {
5.     public $user;
6.
7.     /**
8.      * Constructor
9.      */
10.    function __construct()
11.    {
12.        $this->CI =& get_instance();
13.        $auth = $this->CI->config->item('auth');
14.
15.        $this->CI->load->helper('cookie');
16.        $this->CI->load->model('users_model');
17.
18.        $this->users_model =& $this->CI->users_model;
19.    }
20.
21.    /**
22.     * Cek login user
23.     */
24.
25.    function login($username, $password)
26.    {
27.
28.        $result = $this->users_model->get_login_info($username);
29.
30.        if ($result) // Result Found
```

```

31.      {
32.          $password = md5($password);
33.          if ($password === $result->password)
34.          {
35.              // Start session
36.              $this->CI->session->set_userdata('user_id',
37.                                              $result->user_id);
38.              return TRUE;
39.          }
40.      }
41.      return FALSE;
42.  }
43.
44.
45. /**
46. * cek apakah udah login
47. */
48. function is_login ()
49. {
50.     return (( $this->CI->session->userdata('user_id')) ? TRUE :
51.             FALSE);
52. }
53. /**
54. * Logout
55. *
56. */
57. function logout ()
58. {
59.     $this->CI->session->unset_userdata('user_id');
60. }
61.
62. }

```

Library access di atas membutuhkan library session dan database, jadi sebelum menggunakan library tersebut pastikan Anda telah menyetting konfigurasi CodeIgniter dengan benar.

Perhatikan fungsi login, fungsi itu bertugas untuk melakukan pengecekan terhadap data username dan password yang diberikan oleh user. Fungsi login akan mengambil semua informasi tentang user berdasarkan username (baris 28). Setelah data didapatkan maka dilakukan pencocokan username dan password (baris 33). Jika passwordnya cocok maka session user tersebut disimpan sebagai penanda bahwa user telah login.

#### 4. Membuat Library Access

Untuk mendapatkan data user tersebut maka kita membutuhkan sebuah model. Model ini sangat sedehana hanya terdiri satu fungsi yaitu `get_login_info`. Fungsi tersebut akan mengembalikan object data user apabila username yang dimasukkan ada di database dan memberikan hasil FALSE jika data user tidak ditemukan

### **application/models/users\_model.php**

```
1. <?php if (!defined('BASEPATH')) exit('No direct script access
   allowed');
2.
3. class Users_model extends CI_Model
4. {
5.
6.     public $table      = 'users';
7.     public $primary_key = 'user_id';
8.
9.     function __construct()
10.    {
11.        parent::__construct();
12.    }
13.
14.    function get_login_info($username)
15.    {
16.        $this->db->where('username', $username);
17.        $this->db->limit(1);
18.        $query = $this->db->get($this->table);
19.        return ($query->num_rows() > 0) ? $query->row() : FALSE;
20.    }
21.
22. }
```

## **5. Membuat Controller Member**

Pertama-tama kita akan membuat controller untuk login. Controller tersebut kita beri nama member. Controller member ini mempunyai dua fungsi utama yaitu login dan logout

```
1. <?php if (!defined('BASEPATH')) exit('No direct script access
   allowed');
2.
3. class Member extends CI_Controller
4. {
5.     function __construct()
6.     {
7.         parent::__construct();
8.         $this->load->library('access');
9.
10.    }
11.
12.    function index()
13.    {
14.
15.        $this->access->logout();
16.        $this->login();
17.
18.    }
19.
20.    function login()
21.    {
22.
23. }
```

```

24.      $this->load->library('form_validation');
25.      $this->load->helper('form');
26.
27.      $this->form_validation->set_rules('username', 'Username',
28.                                         'trim|required|strip_tags');
29.      $this->form_validation->set_rules('password', 'Password',
30.                                         'trim|required');
31.      $this->form_validation->set_rules('token', 'token',
32.                                         'callback_check_login');
33.
34.
35.
36.      $this->template->display('member/login');
37.    }
38.  else
39.  {
40.    redirect('dashboard');
41.  }
42. }
43.
44. function logout()
45. {
46.   $this->access->logout();
47.   redirect('member/login');
48. }
49.
50. function check_login()
51. {
52.
53.   $username = $this->input->post('username', TRUE);
54.   $password = $this->input->post('password', TRUE);
55.
56.   $login = $this->access->login($username, $password);
57.   if($login)
58.   {
59.     return TRUE;
60.   }
61.   else
62.   {
63.     $this->form_validation->set_message('check_login',
64.                                         'Username atau password anda salah.');
65.     return FALSE;
66.   }
67.
68.
69. }

```

Pada controller member kita akan melakukan pengecekan terhadap inputan yang telah di masukkan user. Oleh karena itu kita tetap menggunakan library validation. Kita akan menggunakan custom validation. Perhatikan baris 29 dan baris 50. Fungsi check\_login akan

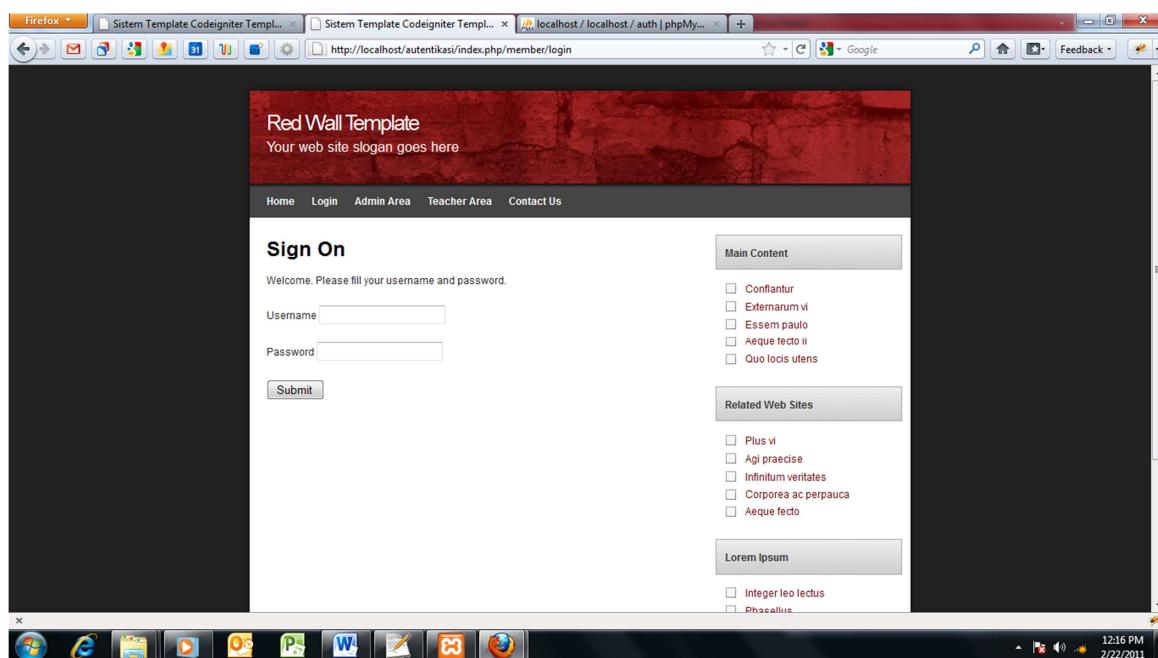
menjadi custom validator pada controller ini. Pada fungsi itu juga library access dipanggil untuk melakukan pengecekan login. Baris 63 digunakan untuk mengeset pesan error pada validasi.

## 6. Membuat view controller member

Setelah memiliki model dan controller maka kita tinggal membuat view

```
1. <?php $v =& $this->form_validation ?>
2.
3.   <h1> Sign On</h1>
4.
5.   <p>Welcome. Please fill your username and password.</p>
6.   <?php if(validation_errors()){ ?>
7.     <div class="fail">
8.       <?php echo validation_errors(); ?>
9.     </div>
10.
11.  <?php } ?>
12.
13. <form name="loginform" method="post" target="<?php
    site_url('member/login') ?>" style="margin:0px;">
14.
15.
16. <p><label>Username</label>
17.   <input name="username" id="username" value="<?php echo @$v->username?>" 
      class="input large" type="text" />
18. </p>
19. <p><label>Password</label>
20. <input name="password" id="password" width="100px" type="password" 
      class="input large" />
21. </p>
22. <p><input name="submit" type="submit" value="Submit" 
      class="button"/></p>
23. </form>
```

Adapun tampilan dari view diatas adalah



## 7. Membuat Mengextend library Controller

Controller member hanya digunakan memverifikasi user yang login. Kita akan membuat sebuah controller yang **hanya** bisa diakses oleh **user yang sudah login**. Controller tersebut akan diberi nama Member\_controller. Untuk membuat member controller maka kita harus membuat sebuah file di **MY\_Controller.php** di folder **application/core**

```

1. <?php if (!defined('BASEPATH')) exit('No direct script access
   allowed');
2.
3.
4. class Member_Controller extends CI_Controller {
5.
6.     function __construct()
7.     {
8.         parent::__construct();
9.         if (!$this->access->is_login())
10.        {
11.            redirect('member/login');
12.        }
13.        //bisa ditambahkan fungsionalitas lain
14.
15.    }
16.    function is_login()
17.    {
18.        return $this->access->is_login();
19.    }
20.
21. }
22.
23. class MY_Controller extends CI_Controller {
24.     function __construct()
25.     {
26.         parent::__construct();
27.     }
28. }
```

Class Member\_controller merupakan turunan dari kelas CI\_Controller dengan penambahan fungsi pengecekan apakah user sudah login. Perhatikan baris 9, kita memanggil fungsi is\_login yang digunakan untuk mengecek apakah user sudah login. Apabila user belum login maka akan di redirect ke form login.

Untuk meng-extend kelas bawaan Codeigniter kita harus membuat sebuah kelas MY\_controller. Kita dapat merubah prefix MY\_ menjadi yang lain dari konfigurasi codeigniter.

Adapun contoh penggunaan member\_controller adalah

```

1. <?php if (!defined('BASEPATH')) exit('No direct script access
   allowed');
2.
```

```
3. class Dashboard extends Member_Controller
4. {
5.     function __construct()
6.     {
7.         parent::__construct();
8.     }
9.
10.    function index()
11.    {
12.        $this->template->display('dashboard');
13.    }
14.
15.
16. }
```

## Chapter 11

# Kasus 5. Image Gallery Sederhana

Sebuah image gallery merupakan aplikasi yang menarik untuk dikerjakan. Dengan menggunakan php biasa, image gallery tidak mudah untuk dibuat. Terutama bagian upload dan meresize gambar yang telah di upload. Dengan codeigniter pembuatan gallery menjadi lebih mudah dan cepat. Dengan memanfaatkan library upload, resize serta beberapa helper, kita dapat membuat sebuah gallery yang menarik.

Library upload codeigniter mudah untuk digunakan. Cara pemanggilannya juga sama dengan library lainnya

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

Untuk dapat menggunakan library ini kita harus mengkonfigurasi beberapa hal diantaranya dimana kita akan mengupload file tersebut, tipe dan ukuran file, dan lain-lain. Contohnya:

```
$config['upload_path'] = APPPATH . 'uploads/';
$config['allowed_types'] = 'jpeg|jpg|gif|png';
$config['max_size'] = '1024';
```

```
$this->upload->initialize($config);
```

Setelah mensetting library upload kita tinggal memanggil fungsi do\_upload untuk mengupload file-file yang telah dimasukkan oleh user.

```
if ( ! $this->upload->do_upload() )
{
    //gagal mengupload file & error menyimpan error message dalam
    //variabel erro
    $error = array('error' => $this->upload->display_errors());
}
else
{
    //sukses mengupload file & informasi file disimpan dalam
    // variabel data
    $data = array('upload_data' => $this->upload->data());
}
```

Image gallery yang akan kita buat memiliki fitur untuk mengupload file gambar, lalu meresize gambar tersebut. Setelah semua proses berhasil maka gallery akan menampilkan thumbnail dari image tersebut. Jika thumbnail tersebut di klik maka barulah gambar yang sebenarnya muncul dalam bentuk popup.

Adapun langkah-langkah yang dilakukan untuk membuat image galerry adalah:

- **Membuat folder upload\_image dan thumb** di root aplikasi anda dan jangan lupa untuk mengubah permisi dari folder tersebut sehingga bisa di tulis oleh php.

application	3/14/2011 11:08 PM	File folder
public	3/14/2011 11:32 PM	File folder
system	3/14/2011 11:08 PM	File folder
thumb	3/14/2011 11:53 PM	File folder
upload_image	3/14/2011 11:53 PM	File folder
index.php	1/28/2011 2:33 PM	PHP File

Struktur direktori gallery

- **Mengubah settingan Codeigniter.** Settingan yang harus di ubah adalah \$config['base\_url'] di application/config/config.php, lalu sesuaikan dengan aplikasi.
- **Membuat controller gallery.** Controller ini akan berisi fungsi untuk mengupload dan meresize gambar secara otomatis. Perhatikan controller gallery.php berikut

```
1. <?php if ( ! defined('BASEPATH') ) exit('No direct script access allowed');
2.
3. class Gallery extends CI_Controller {
4.
5.     function __construct()
6.     {
7.         parent::__construct();
8.     }
9.
10.    function index()
11.    {
12.        $config['upload_path'] = './upload_image/';
13.        $config['allowed_types']= 'gif|jpg|png';
14.        $config['max_size']      = '1000';
15.        $config['max_width']    = '2024';
16.        $config['max_height']   = '1468';
17.
18.        $this->load->library('upload', $config);
19.        $this->load->library('template');
20.        $this->load->model('gallery_model');
21.        $this->load->helper(array('form','url'));
22.        $data['message']='';
23.        if ( ! $this->upload->do_upload() )
24.        {
25.            if (isset($_POST['submit']))
26.                $data['message'] = $this->upload->display_errors();
27.        }
28.        else
29.        {
30.            $data ['upload_data'] = $this->upload->data();
31.            $data['message'] = 'Anda telah sukses mengupload gambar !!!';
32.
33.            $config_resize = array(
34.                'source_image'      => $data['upload_data']['full_path'],
35.                'new_image'         => './thumb/',
36.                'maintain_ratio'   => true,
37.                'width'             => 160,
```

```

37.           'height'          => 120
38.       );
39.
40.       $this->load->library('image_lib', $config_resize);
41.       if ( ! $this->image_lib->resize())
42.       {
43.           $data[ 'message' ] = $this->image_lib->display_errors();
44.       }
45.   }
46.
47.   $data[ 'images' ] = $this->gallery_model
48.                   ->fetch_image(FCPATH.'upload_image' );
49.   $this->template->display('gallery',$data);
50.   ;
51. }
52. }
53.
54. /* End of file Gallery.php */
55. /* Location: ./application/controllers/Gallery.php */

```

Perhatikan baris 18-21, disana kita meload semua library dan helper yang dibutuhkan. Khusus untuk library upload kita menggunakan konfigurasi pada saat pemanggilan library `$this->load->library('upload', $config);` Perhatikan parameter kedua (variabel `$config`). Variable tersebut berisi settingan dimana file tersebut akan diupload, apa saja file yang bisa diupload, size file yang boleh di upload dan lain-lain.

Pada baris 23, kita memanggil fungsi untuk mengupload jika image sukses diupload maka kita akan melakukan proses resize gambar (baris 32-44). Untuk meresize image kita juga perlu memberikan parameter khusus pada saat loading library - `$this->load->library('image_lib', $config_resize);` pada konfigurasi itulah kita menentukan ukuran thumbnail yang akan dibuat beserta path thumbnail

- **Membuat model gallery.** Model ini hanya memiliki tugas yang sangat sederhana yaitu menampilkan file apa aja yang berada di sebuah folder. Untuk mempermudah maka kita menggunakan helper dari file untuk mendapatkan list nama file dari folder tertentu

```

1. <?php
2. class Gallery_model extends CI_Model{
3.     function __construct()
4.     {
5.         parent::__construct();
6.     }
7.
8.     function fetch_image($path)
9.     {
10.         $this->load->helper('file');
11.         return get_filenames($path);
12.     }
13. }

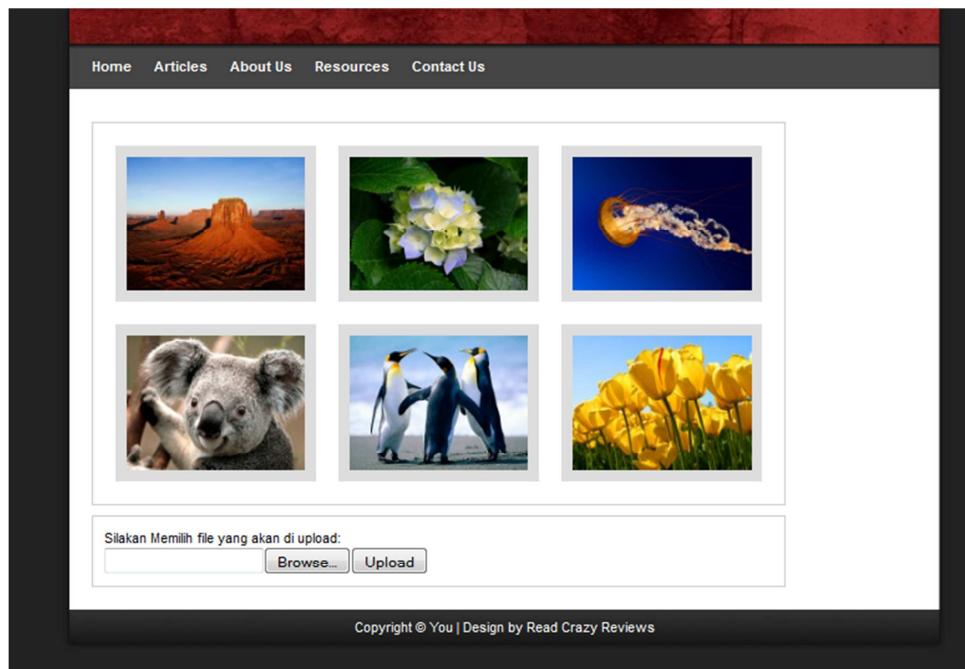
```

Perhatikan baris 10 dan 11 baris tersbut akan mengembalikan daftar nama file yang berada di sebuah folder

- **Membuat view gallery.** View akan menampilkan dua komponen utama yaitu daftar gambar dan form yang akan digunakan untuk mengupload image.

```
1.  <?php echo $message; ?>
2.  <div id="gallery">
3.  <?php
4.  $atts = array(
5.      'width'      => '800',
6.      'height'     => '600',
7.      'scrollbars' => 'yes',
8.      'status'     => 'yes',
9.      'resizable'  => 'yes',
10.     'screenx'    => '0',
11.     'screeny'    => '0'
12. );
13.
14.
15.     foreach($images as $image):
16. ?>
17.
18.     <div class="thumb">
19.         <a href=<?php echo base_url().'upload_image/'.$image; ?>">
20. <?php echo anchor_popup(base_url().'upload_image/'.$image
21. , '', $atts);
22. ?>
23.     </a></div>
24. <?php
25. endforeach;
26. ?>
27. </div>
28. <div id="upload"> Silakan Memilih file yang akan di upload:
29. <?php
30. echo form_open_multipart('gallery');
31. echo form_upload('userfile');
32. echo form_submit('upload','Upload');
33. echo form_close();
34. ?>
35. </div>
```

Perhatikan baris ke empat. Disana kita men-setting anchor\_popup (link popup) yang akan digunakan untuk menampilkan gambar secara detail (versi asli yang diupload). Pada baris 29-34 adalah bagian yang digunakan untuk mengupload form gambar tersebut. Jika semua installasi yang telah anda lakukan benar maka anda akan mendapatkan tampilan seperti berikut ini.



Screenshot Gallery Image

Untuk yang lebih advancenya anda dapat menggabungkan image berikut ini dengan menggunakan script gallery javascript lainnya atau mengkombinasikannya menggunakan database

## Chapter 11

# Kasus 6. Buku Tamu menggunakan Codeigniter

Setelah mengetahui konsep dasar CodeIgniter mari kita mulai sebuah contoh penggunaan CodeIgniter. Pada contoh pertama ini akan ditunjukkan bagaimana menggunakan query-query dasar pada CodeIgniter. Kita akan membuat Buku Tamu. Selanjutnya ikutilah langkah-langkah berikut ini.

## Membuat Table Guestbook

Pada contoh ini kita akan membuat sebuah aplikasi Buku Tamu. Untuk itu kita membutuhkan sebuah tabel dengan schema sebagai berikut

```
CREATE TABLE guestbook
(
    id bigint auto_increment PRIMARY KEY,
    nama varchar(50),
    email varchar(50),
    tanggal datetime,
    komentar text,
    status int
);
```

## Konfigurasi Guestbook

Buka file **application/config/database.php**. Setting sesuai dengan konfigurasi mysql Anda. Isikan username, password dan nama database yang digunakan. Setelah itu buka juga file **application/config/autoload.php** ubahlah variabel **\$autoload['libraries']** (kira-kira baris 41) menjadi

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

Kemudian buka melalui browser. Jika tidak terjadi kesalahan apapun berarti Anda sudah berhasil menyeting database dengan benar

## Membuat Model Guestbook

Untuk mendapatkan data dari database maka kita harus memiliki model yang dapat mengambil data tersebut. Oleh karena itu Anda harus menambahkan model berikut di **Application/models/guestbook\_model.php**. Adapun isi dari file tersebut adalah

```
<?php
class Guestbook_model extends CI_Model {

    public $table_record_count;

    function Guestbook_model()
    {
        parent::Model();
    }

    function get_data($start=NULL,$count=NULL)
    {
        $results=array();
        $this->db->from('guestbook');
        $this->table_record_count = $this->db->count_all_results();

        if($start)
        {
            if($count)
                $this->db->limit($start,$count);
            else
                $this->db->limit($start);
        }
        $query=$this->db->get('guestbook');
        if($query->num_rows()>0)
            return $query->result_array();
        else
            return FALSE;
    }

    function add($data)
    {
        $this->db->insert('guestbook',$data);
        return $this->db->insert_id();
    }

    function update($keyvalue,$data)
    {
        $this->db->where('id',$keyvalue);
        $this->db->update('guestbook',$data);
        return $this->db->affected_rows();
    }

    function delete($idField)
    {
        $this->db->where('id',$idField);
        $this->db->delete('guestbook');
        return true;
    }
}
```

}

Model diatas terdiri atas lima fungsi yaitu **konstruktor**, **get\_data**, **add**, **update** dan **delete**. Fungsi **get\_data** digunakan untuk mengambil isi dari tabel guestbook. Fungsi tersebut terdiri atas dua parameter yaitu start dan limit. Parameter tersebut dibutuhkan karena kita menggunakan library pagination. (Library pagination digunakan untuk membuat fitur paging/halaman pada data). Library pagination membutuhkan beberapa data diantaranya jumlah semua record. Pada variabel **\$this->table\_record\_count** akan disimpan jumlah semua record di tabel.

## Membuat Controller Dan View

Pada controller ini kita akan memanggil library, helper dan model yang akan digunakan. Adapun library yang akan digunakan adalah **form\_validation** dan **table**. Sedangkan helper yang harus digunakan adalah url dan smileys. Adapun hasil keluaran yang akan tampak adalah sebagai berikut:



Selanjutnya adalah pembuatan controller

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

class Guestbook extends CI_Controller {

function Guestbook()
{
    parent::CI_Controller ();
    $this->load->library('pagination');
    $this->load->library('form_validation');
    $this->load->helper('smiley');
    $this->load->helper('url');
    $this->load->library('table');
    $this->load->model('guestbook_model', 'guestbook');
}
```

```

function show()
{
    if($this->_validate_data())
    {
        $data[ 'nama' ]=$this->input->post( 'nama' ,TRUE );
        $data[ 'email' ]=$this->input->post( 'email' ,TRUE );
        $data[ 'komentar' ]=$this->input->post( 'komentar' ,TRUE );
        $data[ 'tanggal' ]= date( 'Y-m-d H:m:s' );
        $data[ 'status' ]=0;

        if($this->guestbook->add($data))
            $data[ 'status' ]='Guestbook sukses ditambahkan';
        else
            $data[ 'status' ]='Guestbook gagal ditambahkan';
    }
    $paging_uri=2;
    if($this->uri->segment($paging_uri))
        $start=$this->uri->segment($paging_uri);
    else
        $start=0;

    $limit_per_page=10;
    $data[ 'tguestbook_list' ]=$this->guestbook
->get_data($limit_per_page,$start);

    $config[ 'base_url' ]= site_url('guestbook');
    $config[ 'total_rows' ]=$this->guestbook->table_record_count;
    $config[ 'per_page' ]=$limit_per_page;
    $config[ 'uri_segment' ]=$paging_uri;

    $this->pagination->initialize($config);
    $data[ 'page_links' ]=$this->pagination->create_links();
    $image_array= get_clickable_smileys(base_url().'smileys/');
    $col_array=$this->table->make_columns($image_array,20);
    $data[ 'smiley_table' ]=$this->table->generate($col_array);
    $this->load->view('guestbook',$data);
}

function index()
{
    $this->show();
}

function _validate_data()
{
    $this->form_validation->set_rules( 'nama' , 'Nama' ,
    'required|min_length[5]|max_length[12]' );
    $this->form_validation->set_rules('email','Email',
    'required|valid_email|htmlspecialchars');
    $this->form_validation->set_rules( 'komentar' , 'Komentar' ,
    'required|htmlspecialchars');

    return($this->form_validation->run() == FALSE)?FALSE:TRUE;
}
}

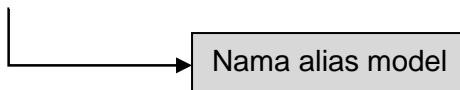
```

Pada controller ini ada 3 fungsi utama yaitu:

- **Konstruktor** – fungsi ini akan kita gunakan sebagai tempat untuk me-load sumberdaya yang dibutuhkan. Seperti library dan helper. Adapun library yang akan digunakan adalah pagination, table, input (sudah autoload). Sedangkan helper yang akan digunakan adalah url dan smiley.

Berikut ini adalah kode yang digunakan untuk memanggil library dan model tersebut.

```
$this->load->library('pagination');
$this->load->library('form_validation');
$this->load->model('guestbook_model','guestbook');
```



Nama alias model

Pada fungsi load model, parameter kedua itu digunakan untuk alias, jadi jika tanpa alias maka kita harus memanggil fungsi model dengan cara `$this->guestbook_model`, sedangkan dengan alias cukup `$this->guestbook`.

- **Show** – fungsi ini akan menampilkan dan memasukkan data buku tamu. Secara logis isi fungsi ini terbagi dua. Yang pertama adalah bagian untuk memasukkan data bukutamu.

```
if($this->_validate_data())
{
    $data['nama']=$this->input->post('nama',TRUE);
    $data['email']=$this->input->post('email',TRUE);
    $data['komentar']=$this->input->post('komentar',TRUE);
    $data['tanggal']= date('Y-m-d H:m:s');
    $data['status']=0;
    if($this->guestbook->add($data))
        $data['status']='Guestbook sukses ditambahkan';
    else
        $data['status']='Guestbook gagal ditambahkan';
}
```

Jika data sudah tervalidasi dengan benar maka artinya data sudah siap untuk dimasukkan. Maka kita akan menggunakan library input untuk mengambil data form lalu memanggil fungsi `$this->guestbook->add($data)` untuk memasukkan data tersebut ke dalam database.

Bagian kedua adalah bagian untuk menampilkan data buku tamu.

```
$paging_uri=2;
```

```

if ($this->uri->segment($paging_uri))
    $start=$this->uri->segment($paging_uri);
else
    $start=0;

$limit_per_page=10;

$data['tguestbook_list']=$this->guestbook
->get_data($limit_per_page,$start);

$config['base_url']= site_url('guestbook');
$config['total_rows']=$this->guestbook->table_record_count;
$config['per_page']=$limit_per_page;
$config['uri_segment']=$paging_uri;

$this->pagination->initialize($config);

$data['page_links']=$this->pagination->create_links();

$image_array= get_clickable_smileys(base_url().'smileys/');
$col_array=$this->table->make_columns($image_array,20);

$data['smiley_table']=$this->table->generate($col_array);

$this->load->view('guestbook',$data);

```

Jika kita akan menampilkan data dalam pagination perlu diingat dua hal, pertama kita membutuhkan inputan berupa berapa jumlah data yang akan ditampilkan dan data dimulai dari halaman keberapa dan data yang dihasilkan dari model harus berisi jumlah total data yang kita punya.

Untuk mendapatkan data yang akan diambil mulai dari data ke berapa, library pagination meletakkannya informasi tersebut di uri tertentu. Pada kasus ini kita meletakkannya di uri ke 2. Maka untuk mendapatkannya kita perlu memanggil **fungsi \$this->uri->segment(2);** Sedangkan untuk jumlah data sudah tersimpan di property model (**table\_record\_count**).

- **Validate\_data** – Fungsi ini bertugas untuk memvalidasi data yang akan masuk ke buku tamu

```

$this->form_validation->set_rules('nama','Nama',
'required|min_length[5]|max_length[12]');
$this->form_validation->set_rules('email','Email',
'required|valid_email|htmlspecialchars');
$this->form_validation->set_rules('komentar','Komentar',
'required|htmlspecialchars');
return($this->form_validation->run() == FALSE)?FALSE:TRUE;

```

Selanjutnya adalah membuat view yang akan digunakan untuk menampilkan data buku tamu tersebut

```
<html>
<head>
<title>Buku Tamu</title>
<style type="text/css">
body {
    background-color: #fff;
    margin: 40px;
    font-family: Lucida Grande, Verdana, Sans-serif;
    font-size: 14px;
    color: #4F5155;
}
</style>
</head>
<body>
<h2>Guest Book </h2>
<hr>

<?php echo validation_errors();?>
<?php if(isset($status)):>
<div class="success">
<span class="message_content">Data Sukses Disimpan</span>
</div>
<?php unset($v);endif;?>

<?
if($tguestbook_list)
{
    foreach($tguestbook_list as $value)
    {
        echo "<li><strong><u>". $value['nama'] .
        "</u></strong> ( ".$value['tanggal']. " ): ".
        nl2br(parse_smileys($value['komentar'],base_url()."smileys/")). "
        <hr></li>";
    }
}
?>

<?php echo $page_links;?>
<?php echo js_insert_smiley('bukutamu','komentar');?>
<br>
<h4>Isi Buku Tamu</h4>
<form name="bukutamu" method="post">
<label for="nama">Nama : </label>
<input type="text" name="nama" value="<?php echo
set_value('nama');?>"/><br>
<label for="email">Email : </label>
<input type="text" name="email" value="<?php echo
set_value('email');?>"/><br>
<textarea name="komentar" cols="40" rows="4">
<?php echo set_value('komentar');?>
</textarea>
<div class="no-border">
```

```
<?php echo $smiley_table;?>
</div>
<input type="Submit" />
</form>
</body>
</html>
```

Pada view tersebut akan di tampilkan data isi buku tamu dalam list beserta form yang akan digunakan untuk menginputkan data. Kita dapat menggunakan validation helper (fungsi **validation\_error**) untuk menampilkan error yang terjadi.

## Chapter 13

# Kasus 7. Membuat Shopping Cart Sederhana

Jika anda ingin membangun toko online, salah satu fitur yang hampir pasti ada yaitu shopping cart (keranjang belanja). Membuat shopping cart tidaklah begitu sulit, terlebih jika anda menggunakan framework CodeIgniter. CodeIgniter telah menyediakan suatu library/pustaka untuk mempermudah anda membuat shopping cart.

Library cart membutuhkan library session, tetapi kita tidak perlu meload library tersebut secara manual. Untuk menggunakannya anda bisa memanggil dengan library loader atau menambahkannya di konfigurasi autoload

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

Penggunaannya juga tidak sulit, untuk menambah item di cart kita bisa menggunakan fungsi inser seperti berikut ini

```
$data = array(
    'id'      => 'sku_123ABC',
    'qty'     => 1,
    'price'   => 39.95,
    'name'    => 'T-Shirt',
    'options' => array('Size' => 'L', 'Color' => 'Red')
);

$this->cart->insert($data);
```

Untuk studi kasus ini, kita akan mengombinasikannya dengan pemilihan produk dimana produk-produk tersebut telah disimpan didalam database. Adapun langkah-langkah yang dilakukan adalah

### 1. Membuat database produk

Ketika kita menggunakan shopping chart tentunya akan ada produk yang akan kita jual. Kita akan menggunakan produk yang sangat sederhana pada contoh kasus ini. Sebuah produk hanya memiliki Id, nama produk dan harga.

```
CREATE TABLE `products` (
    `id` int NOT NULL AUTO_INCREMENT,
    `name` varchar(100) NOT NULL,
    `price` int NOT NULL,
    PRIMARY KEY (`id`)
);

INSERT INTO `products`(`id`, `name`, `price`) VALUES
(1, 'Baju Batman', 30000),
(2, 'Mouse Wireless', 434444),
```

```
(3, 'Tas Ransel', 2344),
(4, 'Kasur Busa', 3453);
```

## 2. Mengkonfigurasi Codeigniter

Kita harus mengkonfigurasi codeigniter terutama di bagian database karena kita akan menggunakan database pada aplikasi ini. Oleh karena itu kita harus mengubah file application /configs/database.php

```
$db['default']['hostname'] = "localhost";
$db['default']['username'] = "root";
$db['default']['password'] = "root";
$db['default']['database'] = "shop ";
$db['default']['dbdriver'] = "mysql";
```

Selain konfigurasi database kita juga mengset konfigurasi base\_url di application/configs/config.php

```
$config['base_url'] = "http://localhost/shopping_cart/";
```

## 3. Membuat Model Produk

Model product mempunyai 2 fungsi yaitu untuk mengambil seluruh data barang, dan fungsi untuk mengambil data barang tertentu sesuai dengan id yang diinginkan.

```
1. <?php
2.
3. class Product_model extends CI_Model {
4.
5.     function __construct()
6.     {
7.         parent::__construct();
8.     }
9.
10.    function get_all($limit = NULL, $offset = NULL) {
11.        $query = $this->db->get('products', $limit, $offset);
12.        return $query->result();
13.    }
14.
15.    function get($id) {
16.        $query = $this->db->get_where('products', array('id'=>$id));
17.        return $query->row();
18.    }
19. }
20.
```

Model ini hanya digunakan untuk menampilkan infomasi produk yang dijual.

## 4. Membuat Controller Produk dan Cart

Setelah membuat model maka kita akan membuat dua buah controller yang akan menghandle penampilan produk dan keranjang belanja. Adapun controller pertama adalah controller produk. Controller ini akan menampilkan semua produk yang ada didalam database

```
1. <?php if ( ! defined('BASEPATH')) exit('No direct script access
   allowed');
2.
3. class Produk extends CI_Controller {
4.
5.     function __construct()
6.     {
7.         parent::__construct();
8.     }
9.
10.    function index()
11.    {
12.        $this->load->library('template');
13.        $this->load->model('product_model','product',true);
14.        $data['product_list'] = $this->product->get_all();
15.        $this->template->display('product', $data);
16.    }
17. }
18.
19. /* End of file Product.php */
20. /* Location: ./application/controllers/Product.php */
```

Kita dapat memilih produk-produk tadi, lalu memasukkannya kedalam keranjang belanja. Setelah itu maka kita harus mempunyai controller lain untuk menghandle keranjang belanja. Ada tiga fungsi utama pada kelas ini yaitu menampilkan, menambah serta merubah keranjang belanja. Untuk menghapus tinggal mengisikan jumlah barang yang dibeli sebanyak 0. Adapun isi dari kontroller tersebut adalah :

```
1. <?php if ( ! defined('BASEPATH')) exit('No direct script access
   allowed');
2.
3. class Cart extends CI_Controller {
4.
5.     function __construct()
6.     {
7.         parent::__construct();
8.         $this->load->model('product_model','product',true);
9.         $this->load->library('cart');
10.        $this->load->library('template');
11.    }
12.
13.    function add($id) {
14.        $product = $this->product->get($id);
15.
16.        $data = array(
17.            'id'      => $product->id,
18.            'qty'     => 1,
19.            'price'   => $product->price,
```

```

20.           'name'      => $product->name,
21.       );
22.
23.       $this->cart->insert($data);
24.       redirect("cart");
25.   }
26.
27.   function update()
28.   {
29.       $this->cart->update($_POST);
30.       redirect("cart");
31.   }
32.
33.   function index() {
34.       $data['cart_list'] = $this->cart->contents();
35.       $this->template->display('cart', $data);
36.   }
37.
38. }
39.
40. /* End of file Cart.php */
41. /* Location: ./application/controllers/Cart.php */

```

Perhatikan baris 16-23, disana kita akan memasukkan data produk. Adapun field-field data product yang dapat dimasukan kedalam library cart adalah:

- id – Setiap produk harus memiliki Id yang unik antara satu dan yang lain.
- qty – Jumlah barang yang akan dibeli
- price – Harga dari produk
- name – Nama produk
- options – informasi tambahan mengenai produk yang ingin dibeli

Untuk memasukkan data cart anda dapat menggunakan fungsi insert - \$this->cart->insert(\$data); data cart tersebut akan disimpan didalam session codeigniter. Jadi library cart ini membutuhkan depedensi library session.

## 5. Membuat View shopping cart

Part terakhir adalah membuat view. Anda harus menyediakan dua buah view yaitu view list barang dan view daftar keranjang belanja. Adapun ini view dari daftar produk adalah

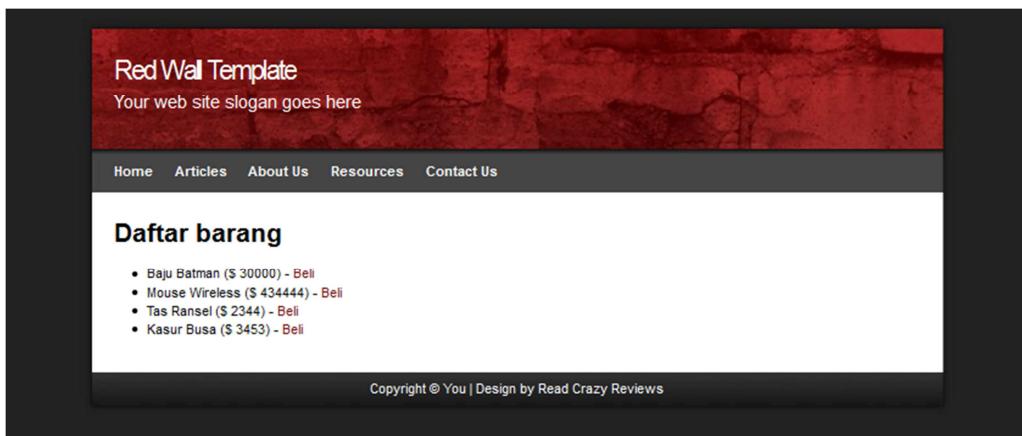
```

1. <h1>Daftar barang </h1>
2. <?php if (!empty($product_list)): ?>
3. <ul>
4.     <?php foreach($product_list as $product): ?>
5.     <li>
6.         <?php echo $product->name ?> ($ <?php echo $product->price ?>) -

```

```
7.         <a href='<?php echo site_url("cart/add/$product->id") ?>'>Beli</a>
8.     </li>
9.     <?php endforeach ?>
10.    </ul>
11.    <?php else : ?>
12.        <p>Produk kosong.</p>
13.    <?php endif ?>
```

Adapun tampilan dari view diatas adalah sebagai berikut



Ketika user mengklik link beli pada list produk maka user akan di redirect ke halaman add item ke shopping cart. Adapun view yang digunakan untuk melihat daftar shopping cart adalah

```
1. <h1>Shopping cart anda</h1>
2. <?php echo form_open('cart/update'); ?>
3.
4. <table cellpadding="6" cellspacing="1" style="width:100%" border="0">
5.
6. <tr>
7.   <th>QTY</th>
8.   <th>Item Description</th>
9.   <th style="text-align:right">Item Price</th>
10.  <th style="text-align:right">Sub-Total</th>
11. </tr>
12.
13. <?php $i = 1; ?>
14.
15. <?php foreach($this->cart->contents() as $items): ?>
16.
17.   <?php echo form_hidden($i.'[rowid]', $items['rowid']); ?>
18.
19.   <tr>
20.     <td><?php echo form_input(array('name' => $i.'[qty]', 'value' =>
21.           $items['qty'], 'maxlength' => '3', 'size' => '5')); ?></td>
22.     <td>
23.       <?php echo $items['name']; ?>
24.
25.       <?php if ($this->cart->has_options($items['rowid'])) == TRUE): ?>
26.
27.           <p>
```

```

28.   <?php foreach ($this->cart->product_options($items['rowid']) as
29.           $option_name => $option_value): ?>
30.
31.       <strong><?php echo $option_name; ?></strong>
32.       <?php echo $option_value; ?><br />
33.
34.   <?php endforeach; ?>
35.   </p>
36.
37.   <?php endif; ?>
38.
39.   </td>
40.   <td style="text-align:right">
41.       <?php echo $this->cart->format_number($items['price']); ?></td>
42.       <td style="text-align:right">$
43.       <?php echo $this->cart->format_number($items['subtotal']); ?></td>
44.   </tr>
45.
46. <?php $i++; ?>
47.
48. <?php endforeach; ?>
49.
50. <tr>
51.   <td colspan="2"> </td>
52.   <td class="right"><strong>Total</strong></td>
53.   <td class="right">$
54.   <?php echo $this->cart->format_number($this->cart->total());
55. ?></td>
56. </tr>
57. </table>
58.
59. <p><?php echo form_submit('', 'Update your Cart'); ?></p>
60.
61.   <a href="<?php echo site_url('produk') ?>">Kembali</a>

```

Perhatikan baris 15, fungsi \$this->cart->contents() digunakan untuk mendapatkan seluruh data cart yang telah disimpan di session. Semua item tadi akan kita simpan juga dalam sebuah inputan yang bersifat hidden sehingga memudahkan kita dalam proses update cart. Selain itu library cart juga sudah dilengkapi dengan fungsi untuk menampilkan jumlah belanja yang telah dilakukan dengan fungsi \$this->cart->total();

Adapun tampilan dari view diatas adalah gambar berikut ini.

The screenshot shows a shopping cart interface with a red header bar containing the text "Red Wall Template" and "Your web site slogan goes here". Below the header is a dark navigation bar with links: Home, Articles, About Us, Resources, and Contact Us. The main content area is titled "Shopping cart anda". It displays a table of items:

QTY	Item Description	Item Price	Sub-Total
4	Mouse Wireless	434,444.00	\$1,737,776.00
1	Tas Ransel	2,344.00	\$2,344.00
1	Kasur Busa	3,453.00	\$3,453.00
Total		\$1,743,573.00	

Below the table are two buttons: "Update your Cart" and "Kembali". At the bottom of the page is a copyright notice: "Copyright © You | Design by Read Crazy Reviews".

Pada gambar diatas kita dapat mengubah jumlah barang yang dipesan. Subtotal dan total akan otomatis ditambahakan oleh library cart.

## Chapter 12

# Kasus 8. CodeIgniter dan Ajax

AJAX yang dimaksud disini bukanlah nama club sepakbola yang berasal dari Amsterdam, Anda atau pun nama pahlawan dalam sejarah perang Trojan, tetapi AJAX di sini adalah singkatan dari *Asynchronous JavaScript and XML*. Pada intinya ajax itu merupakan gabungan beberapa teknologi yang bertujuan untuk menghindari *page reload*. Dengan menghindari *page reload*, kita dapat menghindari paradigma *click-and-wait* serta memberikan sebuah fitur yang cukup kompleks pada website seperti validasi data secara realtime, drag n drop dan fitur-fitur lain yang belum dimiliki web biasa.

Dengan AJAX, suatu aplikasi web dapat mengambil data kemudian diolah di client melalui *request asynchronous HTTP* yang diinisialisasi oleh Javascript, sehingga dapat mengupdate bagian-bagian tertentu dari web tanpa harus memanggil keseluruhan halaman web. Request ini dapat dieksekusi dalam beberapa cara dan beberapa format transmisi data. Dikombinasikannya cara pengambilan data remote dengan interaktivitas dari *Document Object Model (DOM)* telah menghasilkan generasi terbaru dari aplikasi web yang menggebrak aturan-aturan tradisional tentang apa yang dapat terjadi di dalam web. Keuntungan dari aplikasi web berbasis AJAX adalah memungkinkan untuk membuat website dan aplikasi web yang lebih baik dan lebih responsif. Sehingga meningkatkan kemudahan pengguna.

CodeIgniter sebenarnya tidak terpengaruh dengan teknik ajax ini karena ajax bekerja di sisi client sedangkan CI bekerja disisi server. Yang perlu dipersiapkan hanya di sisi template dan view. Untuk bagian library template kita melakukan perubahan dengan menambahkan pengecekan apakah sebuah request tersebut merupakan ajax request apa bukan.

### **application/libraries/template.php**

```

1. <?php
2. class Template {
3. protected $_ci;
4.
5. function __construct()
6. {
7.     $this->_ci =&get_instance();
8. }
9.
10. function display($template,$data=null)
11. {
12.     if(!$this->is_ajax())
13.     {
14.         $data['_content']=$this->_ci->load->view($template,
15.             $data, true);
16.         $data['_header']=$this->_ci->load->view('template/header',
17.             $data, true);

```

```
18.     $data['_top_menu']=$this->_ci->load->view('template/menu',
19.     $data, true);
20.     $data['_right_menu']=$this->_ci->load->view(
21.     'template/sidebar', $data, true);
22.     $this->_ci->load->view('/template.php', $data);
23. }
24. else
25. {
26.     $this->_ci->load->view($template, $data);
27. }
28. }
29.
30. function is_ajax()
31. {
32.     return (
33.         $this->_ci->input->server('HTTP_X_REQUESTED_WITH') &&
34.         ($this->_ci->input->server('HTTP_X_REQUESTED_WITH') ==
35.         'XMLHttpRequest'));
36. }
37. }
```

Perhatikan fungsi **is\_ajax**, fungsi tersebut untuk mengecek apakah request tersebut merupakan sebuah request ajax. Perhatikan baris 26, jika request tersebut merupakan request ajax maka akan ditampilkan view area content saja.

Selain itu, disisi view juga perlu diberi perubahan

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2.  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3.  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4.  <head>
5.  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6.  <link rel="stylesheet" type="text/css" href="
7.  <?php echo base_url();?>public/style.css"/>
8.  <script src="<?php echo base_url();?> public/js/jquery-
    1.4.2.min.js">
9.  </script>
10. <script>
11. $(document).ready(function(){
12.     $('.ajax').click(function(e){
13.         e.preventDefault();
14.         $.get($(this).attr('href'),function(Res){
15.             $('#content').html(Res);
16.         });
17.     })
18. })
19. </script>
20. <title>Sistem Template CodeIgniter Template</title>
21. </head>
22. <body>
23.
24. <div id="wrap">
25. <div id="header">
26. <!--
27. Area Header
```

```

28. -->
29. <?php echo $_header;?>
30. </div>
31.
32. <div id="menu">
33. <!--
34. Area Menu
35. -->
36. <?php echo $_top_menu;?>
37. </div>
38.
39. <div id="contentwrap">
40. <div id="content">
41. <!--
42. Area content
43. -->
44. <?php echo $_content;?>
45. </div>
46. <div id="sidebar">
47. <!--
48. Area Right Menu
49. -->
50. <?php echo $_right_menu;?>
51. </div>
52. <div style="clear: both;"></div>
53. </div>
54. <div id="footer">
55. <p>Copyright ©<ahref="#">You</a> | Design by
56. <a href="http://www.readcrazyreviews.com">Read Crazy Reviews</a></p>
57. </div>
58.
59. </div>
60.
61. </body>
62. </html>

```

Perhatikan baris 8-18, pada kode tersebut kita menggunakan jquery untuk mempermudah melakukan request ajax. Pada kode di atas kita akan menyiari semua link yang memiliki kelas ajax lalu meloadnya melalui ajax (lihat view **template/menu.php**).

### **application/view/template/menu.php**

```

1. <ul>
2. <li><a class="ajax" href=<?php echo base_url();?>">Home</a></li>
3. <li><a class="ajax" href=<?php echo site_url('welcome/page1');?>">
4. Page Ajax 1</a></li>
5. <li><a class="ajax" href=<?php echo site_url('welcome/page2');?>">
6. Page Ajax 2</a></li>
7. <li><a class="ajax" href=<?php echo site_url('welcome/page3');?>">
8. Page Ajax 3</a></li>
9. <li><a class="ajax" href=<?php echo site_url('welcome/page4');?>">
10. Page Ajax 4</a></li>
11. </ul>

```

Untuk bagian controller tidak melakukan perubahan apa-apa. kita hanya menambahkan page yang dapat dipanggil

```
1. <?php if (! defined('BASEPATH')) exit('No direct script access  
allowed');  
2.  
3. class Welcome extends CI_Controller {  
4.  
5.     function __construct()  
6.     {  
7.         parent::__construct();  
8.         $this->load->library('template');  
9.         $this->load->helper('url');  
10.    }  
11.  
12.    function index()  
13.    {  
14.        $this->template->display('welcome_message');  
15.    }  
16.  
17.    function page1()  
18.    {  
19.        $this->template->display('page1');  
20.    }  
21.  
22.    function page2()  
23.    {  
24.        $this->template->display('page2');  
25.    }  
26.  
27.    function page3()  
28.    {  
29.        $this->template->display('page3');  
30.    }  
31.  
32.    function page4()  
33.    {  
34.        $this->template->display('page4');  
35.    }  
36.    }  
37.  
38. /* End of file welcome.php */  
39. /* Location: ./application/controllers/welcome.php */
```

Ketika kita mengakses page 1 secara langsung maka library template akan memampulkan page secara utuh tetapi melalui ajax hanya akan memberikan area content.

## Chapter 13

# Kasus 9. Codeigniter dan jQuery AutoComplete

AutoComplete adalah sebuah fitur dimana kita memberikan saran kepada pengguna mengenai apa yang telah mereka ketikkan di textbox. Jadi user tidak perlu mengetik secara keseluruhan mengenai hal yang ingin dicari. Untuk mendapatkan fitur autocomplete kita dapat menggunakan bantuan javascript terutama jquery. Fitur autocomplete itu sendiri sudah dimiliki oleh plugin jqueryui secara default, sehingga tidak perlu plugin tambahan lagi.

Adapun kasus yang cocok untuk menggunakan fitur autocomplete adalah pencarian nama kota. Misalkan kita memiliki database kota dan kita akan menggunakan database tersebut untuk mempermudah pengguna dalam mengisi field kota.

Adapun hal-hal yang harus di persiapkan adalah

### 1. Membuat Tabel Kota dan Konfigurasi Database CodeIgniter

Buatlah tabel dan isi nama kota seperti contoh berikut

```
CREATE TABLE IF NOT EXISTS `kota` (
  `id_kota` int(11) NOT NULL AUTO_INCREMENT,
  `nama_kota` varchar(50) NOT NULL,
  PRIMARY KEY (`id_kota`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=17 ;

-- Data for table `kota`
--

INSERT INTO `kota`(`id_kota`, `nama_kota`) VALUES
(1, 'Aceh'),
(2, 'Jakarta'),
(3, 'Bandung'),
(4, 'Cirebon'),
(5, 'Dumai'),
(6, 'Batam'),
(7, 'Tanjung Pinang'),
(8, 'Malang'),
(9, 'Mataram'),
(10, 'Maluku'),
(11, 'Marauke'),
(12, 'Surabaya'),
(13, 'Semarang'),
(14, 'Serang'),
(15, 'Selatpanjang'),
(16, 'Sumbawa');
```

Tabel diatas cukup sederhana, kita akan menyimpan id\_kota dan namakota sebagai data utama pada tabel tersebut. Selain itu pastikan Anda telah mengkonfigurasi database

Codeigniter. Konfigurasi tersebut berada di file **application/config/database.php** (perhatikan bab sebelumnya jika Anda bermasalah dalam mengkoneksikan database).

## 2. Membuat Model Tabel Kota

Setelah membuat table, hal berikutnya adalah membuat sebuah model yang digunakan untuk mengakses data dari tabel tersebut. Perhatikan model berikut ini:

```
1. <?php
2. class Kota_model extends CI_Model{
3.
4.     function __construct(){
5.         parent::__construct();
6.     }
7.
8.     function find($keyword){
9.         $this->db->like('nama_kota',$keyword,'after');
10.        $query=$this->db->get('kota');
11.        return $query->result_array();
12.    }
13. }
14. /* End of file kota_model.php */
15. /* Location: ./application/model/kota_model.php */
```

Model tersebut hanya memiliki sebuah fungsi **find** yang digunakan untuk mencari nama kota yang termirip berdasarkan apa yang di ketikkan pengguna. Misalnya ketika pengguna mengetikan hurup “m” maka akan dicarikan semua kota yang berawalan dengan hurup “M” contoh malang.

## 3. Membuat Controller dan View Autocomplete

Autocomplete yang memanfaatkan ajax untuk mendapatkan data, harus menyiapkan sebuah fungsi yang bertujuan untuk memberikan data kepada script autocomplete dan sebuah fungsi untuk menampilkannya. Perhatikan controller Autocomplete berikut ini:

```
1. <?php if (! defined('BASEPATH')) exit('No direct script access
   allowed');
2.
3. class Autocomplete extends CI_Controller {
4.
5.     function __construct()
6.     {
7.         parent::__construct();
8.         $this->load->database();
9.         $this->load->model('kota_model');
10.        $this->load->helper('url');
11.        $this->load->helper('form');
12.    }
13.
14. function index()
```

```

15. {
16.     $this->load->view('autocomplete/index');
17. }
18.
19. function lookup()
20. {
21.     $keyword=$this->input->post('term');
22.     $data['response']='false';
23.     $query=$this->kota_model->find($keyword);
24.     if(! empty($query))
25.     {
26.         $data['response']='true';
27.         $data['message']= array();
28.         foreach($query as $row)
29.         {
30.             $data['message'][]= array(
31.                 'id'=>$row['id_kota'],
32.                 'value'=>$row['nama_kota']
33.             );
34.         }
35.     }
36.     echo json_encode($data);
37. }
38.
39. /* End of file autocomplete.php */
40. /* Location: ./application/controllers/autocomplete.php */

```

Fungsi index pada baris ke-14 hanya berisi sebuah perintah untuk me-load sebuah view yaitu view **autocomplete/index**. Adapun isi view tersebut adalah :

```

1.  <!DOCTYPE HTML>
2.  <html lang="en-US">
3.  <head>
4.  <title>Codeigniter dan jQuery Autocomplete</title>
5.  <link rel="stylesheet" href=
6.  "<?php echo base_url();?>public/jquery.ui.all.css"
7.  type="text/css" media="all"/>
8.  <link rel="stylesheet" href="<?php echo base_url();?>public/ui-
    lightness/jquery-ui-1.8.10.custom.css"
    type="text/css" media="all"/>
9.  <script src="<?php echo base_url();?>public/js/jquery-
    1.4.4.min.js" type="text/javascript"></script>
10. <script src="<?php echo base_url();?>public/js/jquery-ui-
    1.8.10.custom.min.js" type="text/javascript"></script>
11.
12. <script type="text/javascript">
13. $(this).ready( function(){
14.     $("#id_kota").autocomplete({
15.         minLength:1,
16.         source:
17.             function(req, add){
18.                 $.ajax({
19.                     url:"<?php echo base_url();?> "
20.                     +"index.php/autocomplete/lookup",
21.                     dataType:'json',
22.                     type:'POST',

```

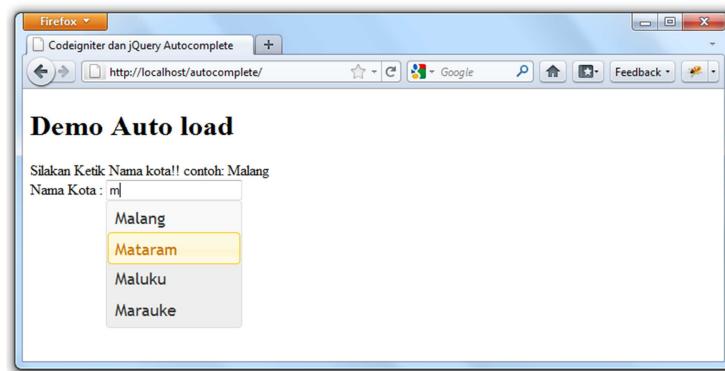
```
23.     data: req,
24.     success:
25.       function(data){
26.         if(data.response == "true"){
27.           add(data.message);
28.         }
29.       },
30.     });
31.   },
32.   select:
33.     function(event, ui){
34.       $("#result").append(
35.         "<li>" + ui.item.value + "</li>"
36.       );
37.     },
38.   },
39. });
40. });
41. </script>
42.
43. </head>
44. <body>
45. <h1>Demo Auto load</h1>
46. Silakan Ketik Nama kota!! contoh: Malang<br/>
47. Nama Kota :
48. <?php
49.   echo form_input('kota','','id="id_kota"');
50. ?>
51. <div id="result"></div>
52. </body>
53. </html>
```

View di atas berisi sebuah inputan yang memiliki nama kota. Baris ke-4 sampai dengan baris ke-10 digunakan untuk me-load semua sumberdaya javascript yang dibutuhkan saat menggunakan jquery autocomplete. Sedangkan baris ke-12 sampai dengan baris ke-41 adalah script javascript yang berfungsi ketika menggunakan autocomplete jquery.

Pada dasarnya sebuah script autocomplete akan memanggil sebuah callback. Callback tersebut akan menghasilkan sebuah data dengan format JSON. Adapun contoh format data yang dibutuhkan adalah

```
{
  "response": "true",
  "message": [
    {"id": "8", "value": "Malang"},
    {"id": "9", "value": "Mataram"},
    {"id": "10", "value": "Maluku"},
    {"id": "11", "value": "Marauke"}
  ]
}
```

Perhatikan baris ke-19 sampai dengan baris ke-20, kita memanggil fungsi lookup dari controller autocomplete karena fungsi tersebut akan menghasilkan data json seperti di atas.



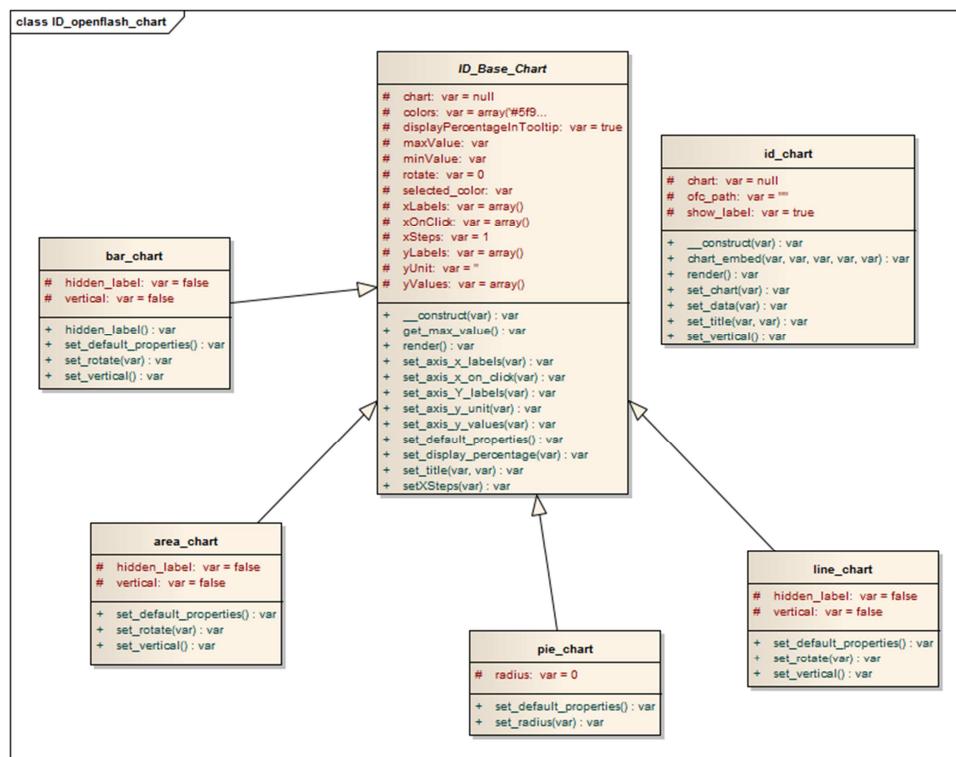
Jika kita memasukkan huruf maka akan keluar kata-kata yang direkomendasikan oleh script autocomplete. Hal tersebut terjadi karena kita telah memanggil fungsi autocomplete dengan menggunakan id inputan tertentu (`$("#id_kota").autocomplete()`). id\_kota merupakan id dari inputan yang telah kita buat.

## Chapter 14

# Kasus 10. Codeigniter dan Openflash Chart

Open Flash Chart adalah perangkat pembangkit grafik berbasis swf. Open flash chart adalah proyek open source. Perangkat ini dapat menampilkan data secara dinamis dan menarik dalam berbagai bentuk animasi grafik, namun demikian grafik dapat juga disimpan dalam bentuk gambar. Keuntungan yang didapatkan ketika menggunakan flash sebagai media grafik adalah selain tampilannya yang menarik ia juga dapat dikendalikan melalui javascript sehingga untuk proses reload, ganti tipe chart dan lain-lain dapat menggunakan library. Open flash chart menerima input berupa data dengan format JSON. Untungnya, ada API dari berbagai jenis bahasa pemrograman yang dapat digunakan untuk meng-generate data ini.

Codeigniter sebagai salah satu framework php juga mampu diintegrasikan dengan library lain meskipun berbeda bahasa. Penulis akan menggunakan library yang telah penulis tulis untuk mengintegrasikan open flash chart dengan codeigniter. Pada bab ini penulis tidak akan membahas proses pembuatan library ini tetapi lebih kearah penggunaanya. Adapun class diagram untuk library yang telah penulis buat adalah seperti gambar berikut ini.



Library diatas dapat diperoleh di [http://www.koder.com/download/id\\_openflashchart.zip](http://www.koder.com/download/id_openflashchart.zip) atau di CD buku. Library di atas didesain untuk bisa menampilkan chart dalam bentuk line, pie, dan area. Adapun yang harus dilakukan untuk mengintegrasikan openflashchart dengan codeigniter adalah:

## 1. Copy Library Ke Application / Library

Setelah Anda men-download library copy-kan library-library tersebut lalu copy-kan ke application/library. Selain itu kita juga membutuhkan library javascript swf object untuk men-generate flash object

## 2. Buat Controller Chart

Buatlah sebuah controller yang akan memanggil library id\_chart. Adapun contoh controller tersebut adalah :

```

1. <?php if (! defined('BASEPATH')) exit('No direct script access
   allowed');
2.
3. class Chart extends CI_Controller {
4.
5. function __construct()
6. {
7.     parent::__construct();
8. }
9.
10. function index()
11. {
12.     $this->load->helper('url');
13.     $this->load->library('id_chart/id_chart');
14.     $chart['c1']=$this->id_chart->chart_embed('test',
15.     800,250,site_url('chart/example1'),base_url());
16.     $chart['c2']=$this->id_chart->chart_embed('test2',
17.     800,250,site_url('chart/example2'),base_url());
18.     $chart['c3']=$this->id_chart->chart_embed('test3',
19.     800,250,site_url('chart/example3'),base_url());
20.     $chart['c4']=$this->id_chart->chart_embed('test4',
21.     300,300,site_url('chart/example4'),base_url());
22.
23.     $this->load->view('chart',$chart);
24. }
25.
26. function example1()
27. {
28.     $this->load->helper('url');
29.     $this->load->library('id_chart/id_chart');
30.     for ($i=1;$i<30;$i++)
31.         $data[] = array('label'=>'data '.$i,
32.             'value'=>rand(1,300));
33.     echo $this->id_chart->set_chart('line')
34.         ->set_data($data)
35.         ->set_vertical()
36.         ->render();
37. }
38.
```

```

39. function example2()
40. {
41.     $this->load->helper('url');
42.     $this->load->library('id_chart/id_chart');
43.     for ($i=1;$i<30;$i++)
44.         $data[] = array('label'=>'data '.$i,
45.                         'value'=>rand(1,300));
46.
47.     echo $this->id_chart->set_chart('bar')
48.         ->set_data($data)
49.         ->set_vertical()
50.         ->render();
51. }
52.
53. function example3()
54. {
55.     $this->load->helper('url');
56.     $this->load->library('id_chart/id_chart');
57.     for ($i=1;$i<30;$i++)
58.         $data[] = array('label'=>'data '.$i,
59.                         'value'=>rand(1,300));
60.
61.     echo $this->id_chart->set_chart('area')
62.         ->set_data($data)
63.         ->set_vertical()
64.         ->render();
65. }
66.
67. function example4()
68. {
69.     $this->load->helper('url');
70.     $this->load->library('id_chart/id_chart');
71.     for ($i=1;$i<6;$i++)
72.         $data[] = array('label'=>'data '.$i,
73.                         'value'=>rand(20,300));
74.
75.     echo $this->id_chart->set_chart('pie')
76.         ->set_data($data)
77.         //->set_radius(20)
78.         ->render();
79. }
80. }
81. /* End of file chart.php */
82. /* Location: ./application/controllers/chart.php */

```

Perhatikan fungsi **index**, fungsi tersebut akan men-generate script javascript yang akan me-load open flash chart. Perhatikan function **chart\_embed**, fungsi tersebut berisi empat parameter diantaranya **\$name** (nama pengenal script), **\$width** (lebar chart), **\$height** (panjang chart), **\$url** (url yang berisi data json), **\$base** (letak flash script berada).

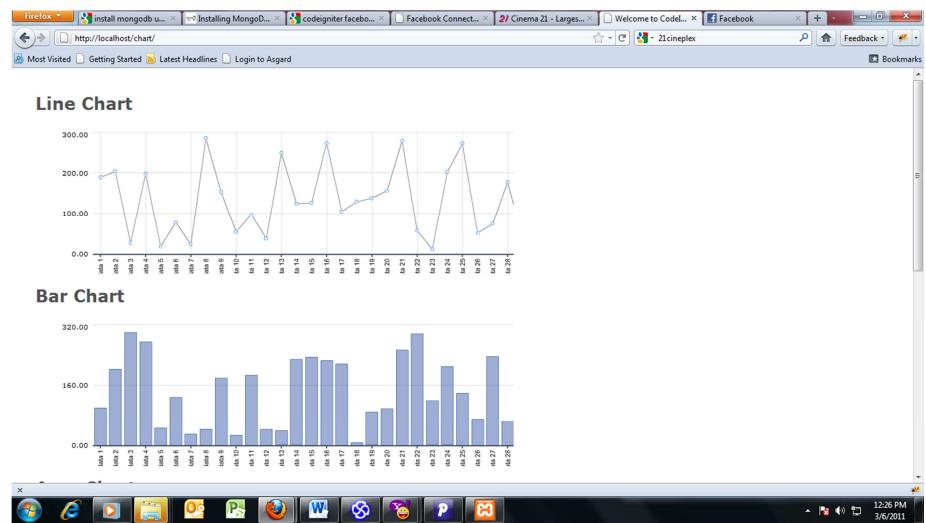
Perhatikan kembali fungsi example1-4 itu adalah contoh fungsi untuk men-generate data json yang akan dipakai oleh flash chart

### 3. Buat View Chart

View yang akan kita buat mirip seperti view pada chapter jquery autocomplete, hanya sebagai view yang berisi script javascript yang akan memanggil openflashchart dan menampilkan chart. Adapun view tersebut adalah :

```
1. <html>
2. <head>
3. <title>Welcome to CodeIgniter</title>
4. <script type="text/javascript" src="<?php echo
base_url();?>/swfobject.js"></script>
5. <style type="text/css">
6.
7. body {
8.   background-color: #fff;
9.   margin: 40px;
10.  font-family: Lucida Grande, Verdana, Sans-serif;
11.  font-size: 14px;
12.  color: #4F5155;
13. }
14.
15.
16. </style>
17. </head>
18. <body>
19.
20. <h1>Line Chart</h1>
21. <?php echo $c1;?>
22. <h1>Bar Chart</h1>
23. <?php echo $c2;?>
24.
25. <h1>Area Chart</h1>
26. <?php echo $c3;?>
27.
28. <h1>Pie Chart</h1>
29. <?php echo $c4;?>
30. </body>
31. </html>
```

Perhatikan baris ke empat, disana kita mencoba me-load **swfobject.js** yang bertujuan me-load script openflashchart. Jika semua terinstall dengan benar maka akan keluar tampilan seperti di bawah ini



Variabel c1, c2, c3 dan c4 merupakan variable yang menyimpan script-script tersebut berasal dan fungsi example1, example2, example3 dan example4 lah yang menentukan tipe beserta data dari masing-masing chart.