PANDUAN
Penggunaan Aplikasi FOSS

CodeIgniter

a powerful PHP framework with a very small footprint

Modul Panduan Framework Codeigniter (Ci)

Oleh Tim Airputih (info@airputih.or.id)

Hak Cipta

DAFTAR ISI

BAB I

TUTORIAL DASAR FRAMEWORK CODEIGNITER (CI)

Pada tahap pembelajaran ini untuk belajar framework harus mengerti dulu atau setidaknya mengenal php dan sistem OOP dalam merancang sebuah website yang terstruktur. Dalam belajar Membangun Framework CI ini yang dibutuhkan adalah.

1. Framework CI (download terlebih dahulu) http://www.codeigniter.com/
2. Siapkan Editornya (Sublime Text / Notepad ++ dll)
3. Database (Xampp/ Wamp)

1.1. Pengenalan Framework PHP

Framework bisa diartikan sebagai kerangka kerja, dalam dunia pemprograman php Framework adalah sekumpulan script yang terstuktur yang dapat membantu seorang develop atau pengembang dalam menangani masalah masalah pemprograman php dengan mudah sehingga mempercepat dalam pembuatan website. Atau bisa dikatakan sebagai sekumpulan fungsi-fungsi php yang terstruktur sehingga programer tidak harus membuat fungsi-fungsi itu dari awal.

Framework beda dengan CMS (Content Management System), apa sih CMS itu ? Apa perbedaan CMS dan Framework. Sebagai orang awam atau bisa dikatakan newbie dalam php akan tanya-tanya. CMS sendiri biasanya digunakan oleh orang develop atau newbie yang tidak mau repot masalah perprograman php. Mereka lebih lebih suka menggunakan CMS karena semua sistemnya sudah terbangun dan tinggal merubah beberapa bagian atau menghilangkan bagian-bagian tertentu untuk bisa menjadi website. Banyak sekali CMS yang ada sekarang ini misalnya Lokomedia, Wordpress, Joomla, Mambo, Drupal dan masih banyak lagi.

Nah sekarang apa sih CMS itu ?? CMS atau dikatakan Sistem Manajemen Konten adalah sebuah perangkat website yang sudah jadi yang memungkinkan seseorang untuk

menambahkan dan juga mengubah atau memanpulasi isi dari website cms tersebut.

Apa Perbedaannya dengan Framework ? Framework sendiri adalah kerangkan kerja website yang terstruktur sedangkan CMS adalah sebuah sistem website yang sudah jadi dan seorang develop tinggal menambahkan atau mengurangi isi kontent tersebut, setidaknya untuk membuat website yang diinginkan hanya merubah sedikit, sedangkan pada framework anda harus membuat dulu backend dan juga frontendnya.

Lebih gampangnya dalam mencermati ini semisal anda adalah seorang pengrajin patung, istilah Framework itu adalah anda membuat patung dari nol, yaitu dengan memahat kayu sampai jadi patung. Kalo CMS adalah anda tinggal membeli patung yang sudah jadi , anda tinggal merakitnya dengan memasang bagian- bagiannya dan memberi warna maka sudah manjadi patung.

Framework dalam pemprograman php ada banyak jenisnya, dalam buku ini saya akan membahas salah satu framework yang sering juga digunakan oleh kalangan programer website yaitu CI (CodeIgniter). Dalam framework ada beberapa struktur website yang harus diketahui. Untuk memulai ini anda harus bisa atau setidaknya mengerti php dasar dulu agar tidak kebingungan masalah pemprograman website dengan framework.

1.2. Pengenalan Framework CodeIgniter

Framework CodeIgniter adalah ini adalah framework yang menggunakan model MVC (model,view dan controller) untuk membangun sebuah website yang dinamis dengan menggunakan php, framework ini adalah turunan dari php jadi kodingnya tidak jauh jauh dari php yang anda kenal sebelumnya dengan model view controller ini seorang developer akan mudah dalam membangun sebuah aplikasi website dan desain tampilan yang terstuktur sehingga dengan model ini dalam maintenance website sangat mudah. CodeIgniter merupakan salah satu framework terbaik saat ini, banyak para develop web menggunakan framework ini dalam pembuatan sistem dan aplikasi mereka.

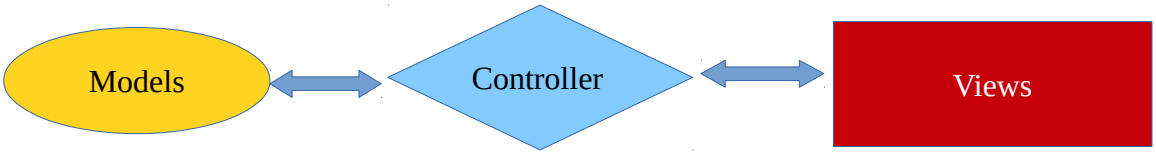
Selain kemudahan dan penggunaannya, codeIgniter juga termasuk framework yang stabil dan cepat dalam penggunaannya, karena menggunakan struktur MVC dalam penggunaan framework ini tak jarang sudah banyak website dan aplikasi menggunakan ini bahkan sudah banyak komunitas codeIgniter dimana – mana.

1.3. Pengertian MVC

Konsep MVC merupakan konsep yang harus atau wajib diketahui terlebih dahulu sebelum mengenal framework CodeIgniter. MVC sendiri merupakan sebuah patent/ teknik pemprograman yang memisahkan antara alur, data dan antarmuka suatu sistem atau bisa dikatakan secara sederhana bahwa MVC sebuah patent dalam framework yang memisahkan antara desain, data dan proses

Untuk lebih jelasnya konsep MVC sebagai berikut :

- **Model**, biasanya berhubungan langsung dengan database untuk memanipulasi data (insert, update, delete, search), menangani validasi dari bagian controller, namun tidak dapat berhubungan langsung dengan bagian view.
- **View**, merupakan bagian yang menangani presentation logic. Pada suatu aplikasi web bagian ini biasanya berupa file template HTML, yang diatur oleh controller. View berfungsi untuk menerima dan merepresentasikan data kepada user. Bagian ini tidak memiliki akses langsung terhadap bagian model.
- **Controller**, merupakan bagian yang mengatur hubungan antara bagian model dan bagian view, controller berfungsi untuk menerima request dan data dari user kemudian menentukan apa yang akan diproses oleh aplikasi.



Jadi Controller bertugas sebagai penjembatani antara model yang terkoneksi dengan database dengan dibuatkan class dan fungtions di controllers dan dihubungkan ke views untuk ditampilkan aplikasinya.

Keuntungan yang didapatkan dari penggunaan framework codeIgniter ini adalah sebagai berikut :

- **Mudah Dimengerti** : Framework ini tidak jauh – jauh beda dengan php biasanya, anda akan bisa mengerti maksud dari code – code yang ada dalam CI tersebut, karena framework ini turunan dari php biasanya jadi mudah dipelajari.
- **Menghemat Waktu Pembuatan** : Dengan stuktur MVC dan library yang telah disediakan oleh framework CI ini anda tidak usah memikirkan strukturnya, jadi anda hanya fokus pada proses pembuatannya aja.

- **Penggunaan Code Berulangan** : Dengan Framework CodeIgniter, pekerjaan anda akan memiliki struktur yang baku, sehingga anda bisa menggunakan stuktur ini kembali pada pekerjaan anda yang lain.
- **Perfomanya Cepat dan Stabil** : Dalam melakukan eksekusi, CodeIgniter lebih cepat dan stabil dari pada framework yang lain.
- **Konfigurasi yang Minim** : Untuk menyesuaikan dengan database dan keleluasaan routing tetap diizinkan melakukan konfigurasi dengan mengubah beberapa file konfigurasi seperti routes.php, config.php, namun untuk menggunakan CI dengan settingan yang standar, anda tidak perlu merubah banyak setingan di file yang ada pada folder config.php.
- **Banyak Sekali Komunitas** : CodeIgniter ini banyak sekali pengembangnya dan komunitasnya . Jadi anda tidak perlu kawatir akan kesulitan dalam pembuatannya karena dari kalangan komunitas CodeIgniter akan membantu anda untuk memperdalam pemrograman anda.

- **URL Friendly** : dengan menggunakan routes.php anda dapat dengan mudah membuat url menarik dengan meminimalisasi menggunaan $_GET menjadi URL.

Catatan Penting !

Permulaan dalam belajar Framework CodeIgniter ini anda diharusnya mengerti terlebih dahulu konsep php sederhana karena bila anda tidak mengenali terlebih dahulu konsep PHP tersebut, anda akan banyak mengalami kesulitan dalam melakukan pemprograman dengan menggunakan konsep MVC yang ada pada framework CI tersebut. Bukan hanya masalah konsep yang ada pada framework CodeIgniter anda akan kesulitan dan akan terus berkutat seputar PHP.

Framework CodeIgniter ini sudah menggunakan konsep PHP dan OOP (Object Oriented Programming) dan akan lebih bagus lagi bila anda mengenal PHP terlebih dahulu kemudian mengenal konsep OOP.

CodeIgniter 2.0

CodeIgniter versi 2.0 dan keatasnya atau versi terbaru ini banyak sekali perubahan yang

ada dalam framework ini dari versi sebelumnya. Perubahan ini membuat CodeIgniter menjadikan lebih segar , kaya dan matang bandingkan framework lainnya. Perubahan tersebut diantaranya adalah :

1. Menghilangkan PHP4 , karena php4 sudah tidak didukung oleh tim pengembang php dan apabila masih menggunakan php4 CodeIgniter ini akan terlihat ketinggalan jaman.
2. Menghilangkan fitur plugin, karena plugin mirip dengan helper yang menyebabkan kerancuan dalam pemprogramannya maka fitur ini dihilangkan.
3. Menghilangkan fitus Scaffolding , karena fitur ini nyaris tidak pernah digunakan oleh develop framework codeIgniter serta impelentasinya masih kurang bagus.
4. Menambahkan library drive, ini library khusus dimana kita dapat membuat drive dari library yang telah kita buat.
5. Penambahan Support Query String dan Command Line Exeution, dengan ini akan menjawab kesulitan kesulitan yang dialami pada versi sebelumnya.
6. Penambahan Library Cache, untuk meningkatkan kwalitas aplikasi makan library cache baik menggunakan apc, mencached maupun file base.
7. Penambahan fitur package , bertujuan untuk mempermudah distribusi dalam sebuah folder.

BAB II

MENGENAL PHP DAN OOP (OBJECT ORIENTED PROGRAMMING)

Syarat utama dalam belajar pemprograman website , anda diharuskan untuk mengenal bahasa pemprograman web yaitu php. Konsep – konsep dasar dalam belajar php ini

2.1. Pengertian PHP

PHP dikembangkan oleh Rasmus Lerdorf pada tahun 1994 yang pada awalnya mengembangkan sebuah perkakas yang digunakan sebagai engine parsing sebagai penterjemah beberapa macro.

PHP bukan istilah yang ada pada anak anak jaman sekarang ini. PHP (Hypertext Preprocessor)adalah bahasa pemprograman yang digunakan untuk membuat website, atau bisa disebut bahasa pemprograman yang ada disisi server. Ketika anda sudah mengakses sebuah URL , maka sebuah web browser akan melakukan request kesebuah web server, maka web server akan melakukan parsing terhadap file php tersebut. PHP parser yang menjalankan kode kode yang terdapat file index.php kemudian parser akan memenggil web browser untuk menampilkan hasil kode.

Ketika anda ingin mengunakan php maka anda diwajibkan untuk menginstall sebuah web server. Banyak sekali jenis webserver yang digunakan antara lain Apache, IIS, iplanet dan lain lain.

Jika tidak ingin merasa repot – repot dalam menginstall, ada beberapa software paketan yang bisa anda gunakan tanpa harus menginstall satu persatu seperti XAMPP, WAMP, PHPTRIAD, FOX Server dan lain lainnya.

2.2. Pengertian Object Oriented Progamming (OOP)

Object Oriented Programming (OOP) merupakan paradikma pemprograman yang berorientasi pada obyek. Semua paradigma ini akan dibungkus dalam beberapa kelas – kelas atau object object yang terlihat terstuktur yang nantinya setiap obyek dapat menerima pesan,memproses pesan dan mengirim pesan ke obyek lain. OOP ini digunakan untuk mengatasi keterbatasan pada bahasa pemprograman tradisional. Dalam konsep OOP, semua masalah dibagi dalam obyek obyek karena konsep data dan fungsi – fungsi yang akan mengoperasikan digabungkan menjadi satu kesatuan yang dapat disebut sebagai obyek.

2.3. Mengenal Object

Secara sederhana obyek sendiri adalah kumpulan variable dan fungsi yang dibungkus menjadi satu tempat atau kesatuan. Dalam sebuah objek diciptakan memalui sebuah kelas atau dikenal dengan istilah *instan of class* . Didalam obyek sendiri mempunyai 2 elemen utama :

- **Attributes** : merupakan nilai – nilai yang tersimpan dalam obyek tersebut dan secara langsung maupun tidak langsung untuk menentukan karakteristik dari obyek tersebut.
- **Method** : ini merupakan suatu aksi yang akan dijalankan atau dikerjakan obyek tersebut.

Untuk mengenal namanya Attribute / Property dan Method maka kita harus mengenal juga namanya Class dan Inheritance.

2.4. Mengenal Struktur OOP

Pada tutorial kali ini akan dibahas dasar-dasar OOP di dalam bahasa PHP. Versi PHP yang sudah mendukung OOP adalah PHP versi 5 ke atas, sedangkan versi 4 dan sebelumnya belum sepenuhnya mendukung OOP. Tutorial ini akan membahas dasar-dasar OOP sambil langsung mempraktekkan bagaimana OOP tersebut diimplementasikan di PHP.

2.5. Enkapsulasi

Didalam OOP, semua elemen program merupakan objek-objek yang saling berinteraksi

satu sama lain. Dalam interaksinya, tiap objek mempunyai informasi yang bisa diakses oleh objek lain ataupun yang disembunyikan dari objek lain. Sebagai contoh, dalam dunia nyata, objek mobil memiliki steer, pedal, yang langsung berinteraksi dengan sopir. Tetapi sopir tidak secara langsung berinteraksi dengan mesin.

Sama halnya seperti objek-objek dalam OOP. Tiap objek memiliki informasi yang harus disembunyikan maupun yang tersedia untuk public (objek lainnya). Dalam OOP ini disebut enkapsulasi (encapsulation) yang merupakan salah satu konsep OOP yang sangat penting.

2.6. Class

Class adalah merupakan “blueprint” dari objek. Sedangkan objek adalah instance dari class. Class adalah class sendiri adalah struktur dari suaru objek yang didalamnya terdapat Attribute dan juga Method, atau bisa disebut sebuah struktur yang mendefinisikan sebuah variable dan juga method umum pada semua object. class juga merupakan grup suatu object dengan kemiripan attributes/properties, behaviour dan relasi ke object lain.

Sesuai dengan keterangan diatas dapat diartikan bahwa class itu bahan dasar sebelum kita membuat sesuatu, atau juga merupakan sebuah wadah dari apa yang kita buat. Misalkan kita ingin membuat tempe, tentu saja harus ada bahan dasarnya yaitu kedelai, atau ketika kita mempunyai mangga,apel,dan durian maka ketiga benda tadi masuk dalam kumpulan yang bernama buah. Dari kedua contoh tersebut kita bisa menentukan mana “class” dan mana “object”, pada Tempe merupakan object yang dibuat dari class Kedelai.Dan mangga,apel,durian merupakan kumpulan object dari class Buah.

Sebagai contoh di dunia nyata, class seperti cetakan kue, dan objek adalah kuenya sendiri. Dari satu class bisa dibuat berapa pun objek yang diperlukan. Sebuah class dalam php diawali dengan keyword classdiikuti dengan nama class. Pada contoh di bawah adalah deklarasi sebuah

```
// class dalam php dengan nama “artikel”.
// deklarasi class dalam php
class artikel {

   // di sini isi code dari class akan ditulis
```

```
}
```

Sebuah class biasanya memiliki ciri-ciri tertentu, seperti dalam dunia nyata mobil miliki warna, ukuran, jumlah ban, dll. Ciri-ciri ini disebut dengan attribute/property. Selain attribute, class juga memiliki operasi-operasi yang bisa dilakukan oleh class tersebut. Seperti pada mobil memiliki operasi maju, mundur, pindah gigi, dll. Operasi-operasi ini disebut dengan method.

Dalam program OOP pun sama halnya seperti di dunia nyata. Tiap class memiliki attribute dan method. Attribute dan method nya pun bisa satu ataupun sebanyak yang diperlukan. Contoh: class artikel, memiliki attribute: judul, nama_penulis, tanggal_terbit, kategori, dll. Artikel juga memiliki method terbitkan(), arsipkan(), hapus(), edit(), dll.

Berikut ini contoh class “artikel” dalam php dengan method dan attribute nya.

```
class artikel {

    // attribute
    public $tanggal_terbit;

    // method
    function terbitkan() {
        echo 'artikel berhasil diterbitkan ';
    }

}
```

Keterangan: Seperti kita lihat pada contoh di atas, attribute adalah variable biasa dan diawali dengan keyword public. Public merupakan access modifier (access modifier memerlukan pembahasan lanjut, yang berkaitan dengan encapsulation). Sedangkan method merupakan fungsi biasa yang memiki argumen ataupun tidak. Method juga bisa memiliki access modifier sama halnya seperti attribute.

2.7. Object

Objek adalah instance dari class, ini berarti class harus di-instantiate (dibuat objeknya) terlebih dahulu agar bisa digunakan dalam program (kecuali static class). Untuk

membuat objek baru dalam PHP, digunakan keyword new.

Berikut adalah contoh pembuatan objek dalam php:

```
// deklarasi class
class artikel {

    public $tanggal_terbit;

    function terbitkan() {
        echo 'artikel berhasil diterbitkan ';
    }
}
// instantiate objek baru
$artikel_a = new artikel();
// instantiate objek lainnya
$artikel_b = new artikel();
```

Keterangan: Pada contoh di atas, $artikel_a merupakan objek dari class artikel. Objek ini otomatis telah memiliki seluruh atribut dan method yang dimiliki oleh class nya, yaitu: attribute ‘tanggal_terbit’ dan method ‘terbitkan()’. Sama halnya juga dengan $artikel_b memiliki atribut dan method yang sama. Untuk mengakses suatu atribut/method dari objek, maka kita bisa menggunakan

tanda -> (strip, lebih besar dari)Contohnya:

```
// mengakses attribute
$artikel_a->tanggal_terbit
// mengakses method
$artikel_a->terbitkan()
```

2.8. Inheritance

Sebuah class dapat di-inherit (diwariskan) ke class turunannya, sehingga class turunannya memiliki attribute dan method yang sama dengan class induknya, Sebagai contoh, class artikel bisa diwariskan menjadi class berita. Class artikel disebut parent class, dan class berita disebut child class. Dalam inheritance, keyword yang digunakan adalah extends. Berikut contoh inheritance dalam PHP:

```
// parent class
class artikel {
```

```
    public $tanggal_terbit;

    function terbitkan() {
        echo 'artikel berhasil diterbitkan ';
    }
}
// child class
class berita extends artikel {

    // attribute tambahan
    public $nama_wartawan;
    public $nara_sumber;

    // method tambahan
    function cek_nara_sumber() {
        echo 'nara sumber berhasil dicek ';
    }
}
```

Keterangan: Pada contoh di atas, class berita yang merupakan child class akan otomatis memiliki atribut ‘$tanggal_terbit’ dan method ‘terbitkan()’ sebagai warisan dari parent classnya, yaitu class artikel. Sebagai tambahan, class berita memiliki atribut ‘$nama_wartawan’, ‘$nara_sumber’ dan method ‘cek_nara_sumber()’ yang didefinisikan tersendiri pada child class tersebut.

BAB III

MENGENAL BAGIAN UMUM CODEIGNITER

3.1. URL CodeIgniter

Secara default, URL di CodeIgniter dirancang untuk search-engine dan ramah manusia. Alih-alih menggunakan standar "string kueri" pendekatan ke URL yang identik dengan sistem dinamis, CodeIgniter menggunakan pendekatan segment-based:

```
example.com/news/article/my_article
```

Catatan: *URL String Query dapat secara optional diaktifkan, seperti yang dijelaskan di bawah ini.*

URI Segmen

Segmen dalam URL, dalam mengikuti dengan pendekatan Model-View-Controller, biasanya mewakili:

```
example.com/class/function/ID
```

Segmen pertama mewakili kelas controller yang harus dipanggil.

Segmen kedua merupakan fungsi kelas, atau metode, yang harus dipanggil.

Yang ketiga, dan setiap segmen tambahan, mewakili ID dan variabel yang akan diteruskan ke controller.

URI Class dan URL Helper mengandung fungsi yang membuatnya mudah untuk bekerja dengan data URI Anda. Selain itu, URL Anda dapat dipetakan dengan menggunakan fitur URI Routing untuk fleksibilitas yang lebih.

Menghapus file index.php

Secara default, file index.php akan dimasukkan dalam URL Anda:

```
example.com/index.php/news/article/my_article
```

Anda dapat dengan mudah menghapus file ini dengan menggunakan file .htaccess

dengan beberapa aturan sederhana. Berikut adalah contoh dari file tersebut, dengan menggunakan metode "negatif" di mana semuanya diarahkan kecuali item tertentu:

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

Dalam contoh di atas, permintaan HTTP selain yang untuk index.php, gambar, dan robots.txt diperlakukan sebagai permintaan untuk file index.php Anda.

Menambahkan Sufiks URL

Dalam konfigurasi / config.php file yang dapat menentukan akhiran yang akan ditambahkan ke semua URL yang dihasilkan oleh CodeIgniter. Misalnya, jika URL ini:

```
example.com/index.php/products/view/shoes
```

Anda dapat menambahkan akhiran, seperti html, membuat halaman tampak dari jenis tertentu:

```
example.com/index.php/products/view/shoes.html
```

Mengaktifkan Query String

Dalam beberapa kasus, Anda mungkin lebih memilih untuk menggunakan query string URL:

```
index.php?c=products&m=view&id=345
```

CodeIgniter opsional mendukung kemampuan ini, yang dapat diaktifkan dalam aplikasi file config.php / Anda. Jika Anda membuka file konfigurasi Anda, Anda akan melihat item ini:

```
$ config ['enable_query_strings'] = FALSE;
$ config ['controller_trigger'] = 'c';
$ config ['function_trigger'] = 'm';
```

Jika Anda mengubah "enable_query_strings" TRUE fitur ini akan menjadi aktif. Controller dan fungsi maka akan dapat diakses dengan menggunakan "pemicu" kata yang sudah ditetapkan untuk memanggil controller dan metode:

index.php? c = controller & m = metode

Catatan: *Jika Anda menggunakan query string Anda akan harus membangun URL Anda sendiri, daripada memanfaatkan pembantu URL (dan pembantu lain yang menghasilkan URL, seperti beberapa pembantu bentuk) karena ini dirancang untuk bekerja dengan URL berbasis segmen.*

3.2. Controller

Controller adalah jantung dari aplikasi Anda, karena mereka menentukan bagaimana permintaan HTTP harus ditangani.

Apa Controller?

Sebuah controller hanyalah sebuah file kelas yang bernama dengan cara yang dapat dikaitkan dengan URI.

Pertimbangkan URI ini:

example.com/index.php/blog/

Dalam contoh di atas, CodeIgniter akan berusaha untuk menemukan kontroler bernama blog.php dan beban itu.

Ketika nama controller sesuai dengan segmen pertama dari URI, itu akan dimuat.

Mari kita coba: Hello World!

Mari kita membuat controller sederhana sehingga Anda dapat melihatnya dalam tindakan. Menggunakan editor teks Anda, buat file bernama blog.php, dan menaruh kode berikut di dalamnya:

```
<?php
class Blog extends CI_Controller {

        public function index()
        {
                echo 'Hello World!';
        }
}
?>
```

Kemudian simpan file tersebut ke aplikasi Anda

application/controllers/folder.

Sekarang kunjungi situs Anda menggunakan URL seperti ini:

example.com/index.php/blog/

Jika Anda melakukannya dengan benar, Anda akan melihat Hello World !.

Catatan: Nama Kelas harus dimulai dengan huruf besar. Dengan kata lain, ini berlaku:

```
<?php
class Blog extends CI_Controller {

}
?>
```

Hal ini tidak berlaku:

```
<?php
class blog extends CI_Controller {

}
?>
```

Juga, selalu pastikan controller meluas kelas induk kontroler sehingga dapat mewarisi semua fungsinya.

Fungsi

Dalam contoh di atas nama fungsi adalah index (). "Indeks" fungsi selalu dimuat secara default jika segmen kedua dari URI kosong. Cara lain untuk menunjukkan "Hello World" pesan Anda akan ini:

example.com/index.php/blog/index/

Segmen kedua dari URI menentukan fungsi di controller dipanggil.

```
<?php
class Blog extends CI_Controller {
	public function index()
	{
		echo 'Hello World!';
	}
	public function comments()
	{
		echo 'Look at this!';
	}
}
?>
```

Mari kita mencobanya. Menambahkan fungsi baru untuk kontroler Anda:

Sekarang memuat URL berikut untuk melihat fungsi comment:

example.com/index.php/blog/comments/

Anda akan melihat pesan baru Anda.

Passing URI Segmen ke Fungsi Anda

Jika URI Anda berisi lebih dari dua segmen mereka akan dilewatkan ke fungsi sebagai parameter.

Sebagai contoh, katakanlah Anda memiliki URI seperti ini:

example.com/index.php/products/shoes/sandals/123

Fungsi Anda akan melewati segmen URI 3 dan 4 ("sandal" dan "123"):

```
<?php
class Products extends CI_Controller {

    public function shoes($sandals, $id)
       {
       echo $sandals;
       echo $id;
       }
}
?>
```

Penting: *Jika Anda menggunakan fitur URI Routing, segmen dilewatkan ke fungsi Anda akan menjadi orang-orang kembali diarahkan.*

Mendefinisikan default controller

CodeIgniter dapat dikatakan untuk memuat controller default ketika URI tidak hadir, karena akan terjadi ketika hanya URL akar situs Anda diminta. Untuk menentukan controller default, buka aplikasi / config file / routes.php dan mengatur variabel ini:

$ rute ['default_controller'] = 'Blog';

Dimana Blog adalah nama kelas controller yang ingin digunakan. Jika Anda sekarang memuat file index.php utama Anda tanpa menentukan segmen URI Anda akan melihat pesan Hello World Anda secara default.

Remapping Panggilan Fungsi

Seperti disebutkan di atas, segmen kedua dari URI biasanya menentukan fungsi di controller dipanggil. CodeIgniter memungkinkan Anda untuk menimpa perilaku ini melalui penggunaan _remap () function:

```
fungsi publik _remap ()
{
    // Beberapa kode di sini ...
}
```

Penting: *Jika controller berisi fungsi bernama _remap (), selalu akan dipanggil terlepas dari apa URI Anda berisi. Ini menimpa perilaku normal di mana URI menentukan fungsi yang disebut, memungkinkan Anda untuk menentukan sendiri aturan fungsi routing.*

Fungsi panggilan ditimpa (biasanya segmen kedua dari URI) akan diteruskan sebagai parameter untuk **_remap ()** fungsi:

```
public function _remap($method)
{
    if ($method == 'some_method')
    {
       $this->$method();
    }
    else
    {
       $this->default_method();
    }
}
```

Setiap segmen tambahan setelah nama metode yang dilewatkan ke _remap () sebagai parameter opsional kedua. Array ini dapat digunakan dalam kombinasi dengan call_user_func_array PHP untuk meniru perilaku default CodeIgniter.

```
fpublic function _remap($method, $params = array())
{
    $method = 'process_'.$method;
```

```
    if (method_exists($this, $method))
    {
    return call_user_func_array(array($this, $method), $params);
    }
    show_404();
}
```

Pengolahan Output

CodeIgniter memiliki class output yang menangani pengiriman data akhir Anda diberikan ke browser web secara otomatis. Informasi lebih lanjut tentang hal ini dapat ditemukan di Tampilan dan halaman kelas Output. Dalam beberapa kasus, bagaimanapun, Anda mungkin ingin pasca-proses data selesai dalam beberapa cara dan mengirimkannya ke browser sendiri. CodeIgniter memungkinkan Anda untuk menambahkan fungsi bernama _output () ke controller yang akan menerima data output diselesaikan.

Penting: *Jika controller berisi fungsi bernama _output (), itu akan selalu disebut oleh kelas keluaran bukan bergema data diselesaikan langsung. Parameter pertama dari fungsi akan berisi output selesai.*

Berikut adalah contoh:

```
public function _output($output)
{
     echo $output;
}
```

Harap dicatat bahwa **_output ()** fungsi Anda akan menerima data dalam keadaan selesai. Data penggunaan patokan dan memori akan diberikan, file cache tertulis (jika Anda memiliki caching diaktifkan), dan header akan dikirim (jika Anda menggunakan fitur itu) sebelum diserahkan ke **_output ()** fungsi.

Untuk memiliki controller output cache benar, metode _output nya () dapat menggunakan:

```
      if ($this->output->cache_expiration > 0)
```

```
{
     $this->output->_write_cache($output);
}
```

Jika Anda menggunakan fitur ini timer eksekusi halaman dan statistik penggunaan memori mungkin tidak sempurna akurat karena mereka tidak akan memperhitungkan acccount setiap proses lebih lanjut yang Anda lakukan. Untuk cara alternatif untuk mengontrol output sebelum pengolahan akhir dilakukan, silakan lihat metode yang tersedia di Kelas Output.

Private Function

Dalam beberapa kasus, Anda mungkin ingin fungsi tertentu tersembunyi dari akses publik. Untuk membuat fungsi pribadi, cukup menambahkan garis bawah sebagai nama awalan dan tidak akan dilayani melalui permintaan URL. Sebagai contoh, jika Anda memiliki fungsi seperti ini:

```
private function _utility()
{
   // some code
}
```

Mencoba untuk mengaksesnya melalui URL, seperti ini, tidak akan bekerja:

example.com/index.php/blog/_utility/

Pengorganisasian Controller Anda ke Sub-folder

Jika Anda sedang membangun sebuah aplikasi besar Anda mungkin merasa nyaman untuk mengatur controller ke dalam sub-folder. CodeIgniter memungkinkan Anda untuk melakukan hal ini.

Cukup membuat folder dalam aplikasi / direktori controllers dan menempatkan kelas controller dalam diri mereka.

Catatan: *Bila menggunakan fitur ini segmen pertama dari URI Anda harus menentukan folder. Sebagai contoh, katakanlah Anda memiliki controller terletak di sini:*

application / controllers / produk / shoes.php

Untuk memanggil controller di atas URI Anda akan terlihat seperti ini:

```
example.com/index.php/products/shoes/show/123
```

Masing-masing sub-folder Anda mungkin berisi controller standar yang akan dipanggil jika URL hanya berisi sub-folder. Cukup nama kontroler default sebagaimana ditentukan dalam aplikasi / config / file routes.php Anda

CodeIgniter juga memungkinkan Anda untuk remap URI menggunakan nya fitur URI Routing.

Class Constructors

Jika Anda berniat untuk menggunakan konstruktor dalam salah satu Controller Anda, Anda harus menempatkan baris kode berikut di dalamnya:

parent :: __ construct ();

Alasan baris ini diperlukan karena konstruktor lokal Anda akan menimpa satu di kelas induk controller sehingga kita perlu menelepon secara manual.

```
<?php
class Blog extends CI_Controller {

public function __construct()
	{
	parent::__construct();
	// Your own constructor code
	}
}
?>
```

Konstruktor berguna jika Anda perlu mengatur beberapa nilai default, atau menjalankan proses default ketika kelas Anda diberi contoh. Konstruktor tidak bisa mengembalikan nilai, tetapi mereka dapat melakukan beberapa pekerjaan default.

Reserved Nama Fungsi

Sejak kelas controller akan memperpanjang aplikasi pengendali utama Anda harus berhati-hati untuk tidak nama fungsi Anda identik dengan yang digunakan oleh kelas itu, jika fungsi lokal Anda akan menimpa mereka. Lihat Nama Reserved untuk daftar lengkap.

3.3. View

Sebuah View hanyalah sebuah halaman web, atau fragmen halaman, seperti header, footer, sidebar, dll Bahkan, pandangan bisa fleksibel tertanam dalam tampilan lain (dalam pandangan lain, dll, dll) jika Anda membutuhkan jenis hirarki.

Tampilan tidak pernah dibuat secara langsung, mereka harus dimuat oleh controller. Ingat bahwa dalam kerangka MVC, tindakan Controller sebagai polisi lalu lintas, sehingga bertanggung jawab untuk mengambil pandangan tertentu. Jika Anda belum membaca halaman Controller Anda harus melakukannya sebelum melanjutkan.

Menggunakan contoh kontroler yang Anda buat di halaman controller, mari kita tambahkan maksud untuk itu.

Membuat View

Menggunakan editor teks Anda, buat file bernama blogview.php, dan menempatkan ini di dalamnya:

```
<html>
<head>
<title>My Blog</title>
</head>
<body>
      <h1>Welcome to my Blog!</h1>
</body>
</html>
```

Kemudian simpan file tersebut dalam aplikasi Anda */ views / folder.*

Loading View

Untuk memuat file pandangan tertentu Anda akan menggunakan fungsi berikut:

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

Mana nama adalah nama file pandangan Anda. Catatan: Ekstensi file php tidak perlu ditentukan kecuali jika Anda menggunakan sesuatu selain php.

Sekarang, buka file kontroler yang Anda buat sebelumnya disebut blog.php, dan mengganti pernyataan gema dengan fungsi tampilan memuat:

```
<?php
class Blog extends CI_Controller {

	function index()
	{
		$this->load->view('blogview');
	}
}
?>
```

Jika Anda mengunjungi situs Anda menggunakan URL yang Anda lakukan sebelumnya Anda akan melihat tampilan baru Anda. URL yang mirip dengan ini:

example.com/index.php/blog/

Loading Multiple Views

CodeIgniter cerdas akan menangani beberapa panggilan ke *$ this-> load-> view* from dalam controller. Jika lebih dari satu panggilan yang terjadi mereka akan ditambahkan bersama-sama. Sebagai contoh, Anda mungkin ingin memiliki tampilan judul, tampilan menu, tampilan konten, dan pandangan footer. Yang mungkin terlihat seperti ini:

```
<?php

class Page extends CI_Controller {

function index()
   {
       $data['page_title'] = 'Your title';
       $this->load->view('header');
       $this->load->view('menu');
       $this->load->view('content', $data);
       $this->load->view('footer');
   }

}
?>
```

Dalam contoh di atas, kita menggunakan "Data dinamis ditambahkan", yang akan Anda lihat di bawah.

Menyimpan Tampilan dalam Sub-folder

Melihat file Anda juga dapat disimpan dalam sub-folder jika Anda lebih suka jenis organisasi. Ketika melakukan sehingga Anda akan perlu untuk memasukkan nama folder memuat pandangan. contoh:

```
$ this-> load-> view ('folder_name / nama_file');
```

Menambahkan Dynamic Data ke View

Data dilewatkan dari controller ke tampilan dengan cara array atau obyek dalam parameter kedua dari fungsi tampilan pemuatan. Berikut adalah contoh menggunakan sebuah array:

```
$ data = array (
       'title' => 'Judul Saya',
       'pos' => 'My Pos',
       'message' => 'My Pesan'
      );
$ this-> load-> view ('blogview', $ data);
```

Dan inilah contoh menggunakan obyek:

```
$ data = new SomeClass ();
$ this-> load-> view ('blogview', $ data);
```

Catatan: *Jika Anda menggunakan objek, variabel kelas akan berubah menjadi elemen array.*

Mari kita coba dengan berkas controller. Buka menambahkan kode ini:

```
<?php
class Blog extends CI_Controller {

       function index()
       {
              $data['title'] = "My Real Title";
              $data['heading'] = "My Real Heading";

              $this->load->view('blogview', $data);
       }
}
?>
```

Sekarang buka file pandangan Anda dan mengubah teks ke variabel yang sesuai dengan

kunci array dalam data Anda:

```
<html>
<head>
<title><?php echo $title;?></title>
</head>
<body>
	<h1><?php echo $heading;?></h1>
</body>
</html>
```

Kemudian buka halaman pada URL yang Anda telah menggunakan dan Anda akan melihat variabel diganti.

Membuat Loops

Data Array Anda lulus untuk melihat file Anda tidak terbatas pada variabel sederhana. Anda dapat melewati array multi dimensi, yang dapat diulang untuk menghasilkan beberapa baris. Sebagai contoh, jika Anda menarik data dari database Anda biasanya akan berada dalam bentuk array multi-dimensi.

Berikut adalah contoh sederhana. Menambahkan ini ke controller Anda:

```
<?php
class Blog extends CI_Controller {

	function index()
	{
		$data['todo_list'] = array('Clean House', 'Call Mom', 'Run Errands');

		$data['title'] = "My Real Title";
		$data['heading'] = "My Real Heading";

		$this->load->view('blogview', $data);
	}
}
?>
```

Sekarang buka file pandangan Anda dan membuat lingkaran:

```
<html>
<head>
<title><?php echo $title;?></title>
```

```
</head>
<body>
<h1><?php echo $heading;?></h1>

<h3>My Todo List</h3>

<ul>
<?php foreach ($todo_list as $item):?>

<li><?php echo $item;?></li>

<?php endforeach;?>
</ul>

</body>
</html>
```

Catatan: *Anda akan melihat bahwa dalam contoh di atas kita menggunakan sintaks alternatif PHP. Jika Anda tidak terbiasa dengan hal itu Anda dapat membaca tentang itu di sini.*

Returning View Data

Ada parameter opsional ketiga memungkinkan Anda mengubah perilaku fungsi sehingga mengembalikan data string daripada mengirimkannya ke browser Anda. Hal ini dapat berguna jika Anda ingin mengolah data dalam beberapa cara. Jika Anda mengatur parameter untuk benar (boolean) itu akan kembali data. Perilaku default adalah palsu, yang mengirimkannya ke browser Anda. Ingatlah untuk menetapkan ke variabel jika Anda ingin data yang dikembalikan:

```
$string = $this->load->view('myfile', '', true);
```

3.4. Model

Model yang opsional tersedia bagi mereka yang ingin menggunakan pendekatan yang lebih tradisional MVC.

1. *Apa itu Model?*
2. *Anatomi Model*

3. *Memuat Model*
4. *Auto-Memuat Model*
5. *Menghubungkan ke Database Anda*
6. *Apa itu Model?*

Model yang kelas PHP yang dirancang untuk bekerja dengan informasi dalam database Anda. Sebagai contoh, katakanlah Anda menggunakan CodeIgniter untuk mengelola blog. Anda mungkin memiliki kelas model yang berisi fungsi untuk memasukkan, update, dan mengambil data blog Anda. Berikut adalah contoh dari apa yang seperti kelas model akan terlihat seperti:

```
class Blogmodel extends CI_Model {

      var $title   = '';
      var $content = '';
      var $date    = '';

function__construct()
      {
      // Call the Model constructor
      parent::__construct();
      }
function get_last_ten_entries()
      {
      $query = $this->db->get('entries', 10);
      return $query->result();
      }

function insert_entry()
      {
      $this->title = $_POST['title']; // please read the below note
      $this->content      = $_POST['content'];
      $this->date  = time();

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

```
function update_entry()
	{
	$this->title = $_POST['title'];
	$this->content	= $_POST['content'];
	$this->date  = time();

	$this->db->update('entries',$this, array('id' => $_POST['id']));
	}

}
```

Catatan: *Fungsi dalam contoh di atas menggunakan Aktif fungsi database Rekam.*

Catatan: *Demi kesederhanaan dalam contoh ini kita menggunakan $ _POST langsung. Ini adalah praktek umumnya buruk, dan pendekatan yang lebih umum akan menggunakan Input Class $ this-> input-> posting ('title')*

Anatomi Model

Kelas model disimpan dalam aplikasi Anda / model / folder. Mereka dapat bersarang dalam sub-folder jika Anda ingin jenis organisasi.

Prototipe dasar untuk kelas model ini:

```
class Model_name extends CI_Model {

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

Dimana Model_name adalah nama dari kelas Anda. Nama kelas harus memiliki huruf pertama dikapitalisasi dengan sisa nama huruf kecil. Pastikan kelas meluas kelas dasar Model.

Nama file akan menjadi versi huruf kecil dari nama kelas Anda. Sebagai contoh, jika kelas Anda adalah ini:

```
class User_model extends CI_Model {
function __construct()
	{
	parent::__construct();
	}
}
```

File Anda akan ini:

application / model / user_model.php

Memuat Model

Model Anda biasanya akan dimuat dan dipanggil dari dalam fungsi controller. Untuk memuat model Anda akan menggunakan fungsi berikut:

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

Jika model Anda terletak di sub-folder, menyertakan path relatif dari folder model Anda. Sebagai contoh, jika Anda memiliki model yang terletak di application / model / blog / queries.php Anda akan memuatnya menggunakan:

- *$ this-> load-> model ('blog / query');*

Setelah dimuat, Anda akan mengakses fungsi model Anda menggunakan objek dengan nama yang sama dengan kelas Anda:

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

- *$ this-> Model_name-> function ();*

Jika Anda ingin model Anda ditugaskan untuk nama objek yang berbeda Anda dapat menentukan melalui parameter kedua dari fungsi pemuatan:

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

Berikut adalah contoh dari controller, yang memuat model, kemudian menyajikan tampilan:

```
class Blog_controller meluas CI_Controller {
```

```
function blog ()
{
  $ this-> load-> model ('Blog');

  $ data ['permintaan'] = $ this-> blog-> get_last_ten_entries ();

  $ this-> load-> view ('blog', $ data);
     }
}
```

Auto-loading Model

Jika Anda menemukan bahwa Anda memerlukan model khusus secara global di seluruh aplikasi Anda, Anda dapat memberitahu CodeIgniter untuk auto-load selama inisialisasi sistem. Hal ini dilakukan dengan membuka *application/config/autoload.php* dan menambahkan model untuk array autoload.

Menghubungkan ke Database Anda

Ketika model dimuat itu TIDAK tersambung secara otomatis ke database Anda. Berikut pilihan untuk menghubungkan tersedia untuk Anda:

Anda dapat menghubungkan menggunakan metode database standar yang dijelaskan di sini, baik dari dalam kelas Controller Anda atau kelas Model Anda.

Anda dapat memberitahu fungsi model loading otomatis terhubung dengan melewati BENAR (boolean) melalui parameter ketiga, dan pengaturan konektivitas, sebagaimana didefinisikan dalam file konfigurasi database Anda akan digunakan:

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

Anda dapat secara manual lewat pengaturan konektivitas database melalui parameter ketiga:

```
$ config ['hostname']     = "localhost";
$ config ['username']     = "myusername";
$ config ['password']     = "mypassword";
$ config ['database']     = "mydatabase";
$ config ['dbdriver']     = "mysql";
```

```
$ config ['dbprefix']     = "";
$ config ['pconnect']     = FALSE;
$ config ['db_debug']     = TRUE;

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

3.5. Fungsi Helper

Helper (Pembantu), seperti namanya, membantu Anda dengan tugas-tugas. Setiap file helper hanyalah kumpulan fungsi dalam kategori tertentu. Ada URL Helpers, yang membantu dalam menciptakan link, ada Form Helper yang membantu Anda membuat elemen bentuk, Helpers Teks melakukan berbagai rutinitas format teks, Cookie Pembantu mengatur dan membaca cookie, file Pembantu membantu Anda menangani file, dll

Tidak seperti kebanyakan sistem lain dalam CodeIgniter, Helper tidak ditulis dalam format Object Oriented. Mereka sederhana, fungsi prosedural. Setiap fungsi pembantu melakukan satu tugas tertentu, tanpa ketergantungan pada fungsi lainnya.

CodeIgniter tidak memuat Helper Files secara default, sehingga langkah pertama dalam menggunakan Helper adalah untuk memuatnya. Setelah dimuat, itu menjadi tersedia secara global di controller Anda dan pemandangan.

Helper biasanya disimpan dalam sistem / pembantu, atau aplikasi / direktori pembantu Anda. CodeIgniter akan melihat pertama dalam aplikasi / direktori helper Anda. Jika direktori tidak ada atau penolong ditentukan tidak terletak di sana CI malah akan terlihat dalam sistem / folder global yang pembantu Anda.

Loading a Helper

Memuat file helper cukup sederhana dengan menggunakan fungsi berikut:

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

Mana nama adalah nama file penolong, tanpa ekstensi file php atau "pembantu" bagian.

Misalnya, untuk memuat file URL Helper, yang diberi nama url_helper.php, Anda akan

melakukan ini:

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

Seorang Helper dapat dimuat di mana saja dalam fungsi controller (atau bahkan di dalam View file Anda, walaupun itu bukan praktik yang baik), asalkan Anda memuat sebelum Anda menggunakannya. Anda dapat memuat helper di konstruktor controller sehingga mereka menjadi tersedia secara otomatis di setiap fungsi, atau Anda dapat memuat helper dalam fungsi tertentu yang memerlukannya.

Catatan: *Fungsi Helper pemuatan di atas tidak mengembalikan nilai, jadi jangan mencoba untuk menetapkan ke variabel. Hanya menggunakannya seperti yang ditunjukkan.*

Loading Multiple Helpers

Jika Anda perlu memuat lebih dari satu penolong Anda dapat menentukan mereka dalam sebuah array, seperti ini:

```
$ this-> load-> helper (array ('helper1', 'helper2', 'helper3'));
```

Auto-loading Helper

Jika Anda menemukan bahwa Anda perlu helper tertentu secara global di seluruh aplikasi Anda, Anda dapat memberitahu CodeIgniter untuk auto-load selama inisialisasi sistem.

Hal ini dilakukan dengan membuka **application/config/autoload.php** dan menambahkan helper untuk array autoload.

Menggunakan Helper sebuah

Setelah Anda dimuat File Penolong yang berisi fungsi yang ingin Anda gunakan, Anda akan menyebutnya cara Anda akan fungsi PHP standar.

Misalnya, untuk membuat link menggunakan anchor () fungsi dalam satu file pandangan Anda, Anda akan melakukan hal ini:

```
<? php echo anchor ('blog/comments ', 'Klik Disini');?>
```

Dimana "Klik Disini" adalah nama dari link, dan "blog / komentar" adalah URI ke controller / fungsi yang ingin Anda link ke.

"Memperluas" Helper

Untuk "memperpanjang" Helper, membuat file dalam aplikasi Anda / pembantu / folder dengan nama identik dengan Penolong yang ada, tetapi diawali dengan MY_ (item ini dikonfigurasi. Lihat di bawah.).

Jika semua yang perlu Anda lakukan adalah menambahkan beberapa fungsi ke pembantu yang ada - mungkin menambahkan satu atau dua fungsi, atau mengubah cara fungsi pembantu tertentu beroperasi - maka itu berlebihan untuk mengganti seluruh pembantu dengan versi. Dalam hal ini lebih baik untuk hanya "memperpanjang" Penolong. Istilah "memperpanjang" digunakan secara longgar karena fungsi Helper yang prosedural dan diskrit dan tidak dapat diperpanjang dalam arti program tradisional. Di bawah tenda, ini memberikan Anda kemampuan untuk menambah fungsi Helper yang menyediakan, atau memodifikasi bagaimana fungsi Helper asli beroperasi.

Misalnya, untuk memperpanjang Array Helper asli Anda akan membuat sebuah file bernama **application/helpers/MY_array_helper.php**, dan menambah atau mengganti fungsi:

```
// any_in_array() is not in the Array Helper, so it defines a new function
function any_in_array($needle, $haystack)
{
    $needle = (is_array($needle)) ? $needle : array($needle);
    foreach ($needle as $item)
      {
    if (in_array($item, $haystack))
      {
            return TRUE;
      }
      }

    return FALSE;
}

// random_element() is included in Array Helper, so it overrides the native
```

```
function
function random_element($array)
{
    shuffle($array);
    return array_pop($array);
}
```

Setting Your Own Prefix

Nama file awalan untuk "memperpanjang" Pembantu adalah sama dengan yang digunakan untuk memperluas perpustakaan dan kelas Core. Untuk mengatur awalan Anda sendiri, membuka aplikasi konfigurasi config.php Anda / / dan mencari item ini:

$config['subclass_prefix'] = 'MY_';

Harap dicatat bahwa semua CodeIgniter perpustakaan asli diawali dengan CI_ jadi JANGAN menggunakannya sebagai awalan Anda.

3.6. Fungsi umum

CodeIgniter menggunakan beberapa fungsi untuk operasi yang didefinisikan secara global, dan tersedia untuk Anda di setiap titik. Ini tidak memerlukan memuat libraries atau helper.

- **is_php('version_number')**

is_php () menentukan versi PHP yang digunakan lebih besar dari VERSION_NUMBER disediakan.

```
if (is_php ('5.3.0'))
{
     $ str = quoted_printable_encode ($ str);
}
```

Pengembalian boolean TRUE jika versi diinstal PHP sama dengan atau lebih besar dari nomor versi yang disediakan. Mengembalikan FALSE jika versi diinstal PHP lebih rendah dari nomor versi yang disediakan.

- **is_really_writable('path/to/file')**

is_writable () mengembalikan *TRUE* pada server Windows ketika Anda benar-benar tidak bisa menulis ke file sebagai laporan OS untuk PHP sebagai palsu hanya jika atribut read-only ditandai. Fungsi ini menentukan apakah file sebenarnya ditulis oleh

mencoba untuk menulis terlebih dahulu. Umumnya hanya dianjurkan pada platform di mana informasi ini mungkin tidak dapat diandalkan.

```
if (is_really_writable ('file.txt'))
{
        echo "Saya bisa menulis ini jika saya ingin";
}
    lain
{
        echo "File tidak dapat ditulis";
}
```

- **config_item('item_key')**

Config perpustakaan adalah cara yang lebih disukai untuk mengakses informasi konfigurasi, namun config_item () dapat digunakan untuk mengambil kunci tunggal. Lihat dokumentasi perpustakaan Config untuk informasi lebih lanjut.

- **show_error('message'), show_404('page'), log_message('level', 'message')**

Ini masing-masing diuraikan pada halaman Kesalahan Penanganan.

- **set_status_header(code, 'text');**

Memungkinkan Anda untuk secara manual mengatur header status server. contoh:

```
set_status_header (401);
// Mengatur header sebagai: tidak sah
```

Lihat di sini untuk daftar lengkap header.

- **remove_invisible_characters($str);**

Fungsi ini mencegah memasukkan karakter nol antara karakter ascii, seperti Java \ 0script.

- **html_escape($mixed)**

Fungsi ini memberikan jalan pintas untuk htmlspecialchars () fungsi. Ini menerima tali dan berbagai. Untuk mencegah Cross Site Scripting (XSS), sangat berguna.

3.7. URI Routing

Biasanya ada hubungan satu-ke-satu antara string URL dan kelas controller / metode yang sesuai. Segmen di URI biasanya mengikuti pola ini:

```
example.com/class/function/id/
```

Dalam beberapa kasus, bagaimanapun, Anda mungkin ingin remap hubungan ini sehingga kelas / fungsi yang berbeda bisa disebut bukan satu sesuai dengan URL.
Sebagai contoh, katakanlah Anda ingin URL Anda memiliki prototipe ini:

```
example.com/product/1/
example.com/product/2/
example.com/product/3/
example.com/product/4/
```

Biasanya segmen kedua dari URL yang disediakan untuk nama fungsi, tetapi dalam contoh di atas itu bukan memiliki ID produk. Untuk mengatasi hal ini, CodeIgniter memungkinkan Anda untuk remap handler URI.

Menetapkan Aturan Routing Sendiri

Aturan routing didefinisikan dalam aplikasi / config file / routes.php. Di dalamnya Anda akan melihat sebuah array disebut $ rute yang memungkinkan Anda untuk menentukan kriteria routing sendiri. Rute baik dapat ditentukan menggunakan wildcard atau Regular Expressions

wildcard

Sebuah wildcard rute khas mungkin terlihat seperti ini:

```
$route['product/:num'] = "catalog/product_lookup";
```

Dalam rute, kunci array berisi URI untuk dicocokkan, sementara nilai array berisi tujuan itu harus kembali dialihkan ke. Dalam contoh di atas, jika kata literal "produk" ditemukan di segmen pertama dari URL, dan nomor ditemukan di segmen kedua, "katalog" kelas dan metode "product_lookup" malah digunakan.

Anda dapat menyesuaikan nilai literal atau Anda dapat menggunakan dua jenis wildcard:

(: num) akan cocok dengan segmen yang hanya berisi angka.
(: any) akan cocok dengan segmen yang berisi karakter apapun.

Catatan: *Rute akan berjalan dalam urutan mereka didefinisikan. Tinggi rute akan selalu lebih diutamakan daripada yang lebih rendah.*

Berikut adalah beberapa contoh routing yang:

```
$route['journals'] = "blogs";
```

URL yang mengandung kata "jurnal" di segmen pertama akan dipetakan ke "blog" kelas.

```
$route['blog/joe'] = "blogs/users/34";
```

URL yang berisi segmen blog / joe akan dipetakan ke "blog" kelas dan "pengguna" metode. ID akan diatur ke "34".

```
$route['product/(:any)'] = "catalog/product_lookup";
```

Sebuah URL dengan "produk" sebagai segmen pertama, dan apa pun di kedua akan dipetakan ke "katalog" kelas dan metode "product_lookup".

```
$route['product/(:num)'] = "catalog/product_lookup_by_id/$1";
```

Sebuah URL dengan "produk" sebagai segmen pertama, dan nomor dalam kedua akan dipetakan ke "katalog" kelas dan "product_lookup_by_id" Metode lewat di pertandingan sebagai variabel ke fungsi.

Penting: Jangan gunakan terkemuka / mengikuti garis miring.

Regular Expressions

Jika Anda lebih suka Anda dapat menggunakan ekspresi reguler untuk mendefinisikan aturan routing. Setiap ekspresi reguler yang valid diperbolehkan, seperti back-referensi.

Catatan: *Jika Anda menggunakan back-referensi Anda harus menggunakan sintaks dolar daripada sintaks backslash ganda.*

Sebuah regex rute khas mungkin terlihat seperti ini:

```
$ rute ['produk / ([a-z] +) / (\ d +)'] = "$ 1 / id_ $ 2";
```

Dalam contoh di atas, URI mirip dengan produk / kemeja / 123 akan sebaliknya memanggil kelas kemeja controller dan fungsi id_123.

Anda juga dapat mencampur dan mencocokkan wildcard dengan ekspresi reguler.

Rute Reserved

Ada dua rute yang dicadangkan:

- `$ rute ['default_controller'] = 'Selamat datang';`

  Rute ini menunjukkan controller yang kelas harus dimuat jika URI tidak berisi data, yang akan terjadi ketika orang-orang beban URL root anda. Dalam contoh di atas, "selamat datang" kelas akan dimuat. Anda dianjurkan untuk selalu memiliki rute default sebaliknya halaman 404 akan muncul secara default.

- `$ rute ['404_override'] = '';`

  Rute ini menunjukkan controller yang kelas harus dimuat jika controller yang diminta tidak ditemukan. Ini akan menimpa halaman kesalahan standar 404. Ini tidak akan berpengaruh terhadap *show_404 () fungsi, yang akan terus memuat file error_404.php default aplikasi / kesalahan / error_404.php.*

3.8. Error Handling

CodeIgniter memungkinkan Anda membangun melaporkan kesalahan dalam aplikasi Anda menggunakan fungsi yang dijelaskan di bawah ini. Selain itu, ia memiliki kesalahan kelas logging yang memungkinkan kesalahan dan debugging pesan yang akan disimpan sebagai file teks.

Catatan: *Secara default, CodeIgniter menampilkan semua kesalahan PHP. Anda mungkin ingin mengubah perilaku ini setelah pembangunan selesai. Anda akan menemukan error_reporting () fungsi yang terletak di bagian atas file index.php utama Anda. Menonaktifkan melaporkan kesalahan TIDAK akan mencegah file log dari ditulis jika ada kesalahan.*

Tidak seperti kebanyakan sistem di CodeIgniter, fungsi kesalahan adalah antarmuka prosedural sederhana yang tersedia secara global di seluruh aplikasi. Pendekatan ini memungkinkan pesan error untuk mendapatkan memicu tanpa harus khawatir tentang kelas / fungsi scoping.

Fungsi-fungsi berikut membiarkan Anda menghasilkan kesalahan:

---

show_error('message' [, int $status_code= 500])

Fungsi ini akan menampilkan pesan kesalahan yang disediakan untuk menggunakan template kesalahan berikut:

application/errors/error_general.php

Opsional parameter $ status_code menentukan apa kode status HTTP harus dikirim dengan kesalahan.

---

show_404('page' [, 'log_error'])

Fungsi ini akan menampilkan pesan kesalahan 404 disediakan untuk itu menggunakan template kesalahan berikut:

application/errors/error_404.php

Fungsi mengharapkan string berlalu untuk itu menjadi path file ke halaman yang tidak ditemukan. Perhatikan bahwa CodeIgniter secara otomatis menunjukkan 404 pesan jika controller tidak ditemukan.

CodeIgniter secara otomatis log show_404 apapun () panggilan. Mengatur parameter opsional kedua ke FALSE akan melewatkan logging.

---

log_message('level', 'message')

Fungsi ini memungkinkan Anda menulis pesan ke file log Anda. Anda harus menyediakan salah satu dari tiga "level" di parameter pertama, menunjukkan jenis pesan itu (debug, error, info), dengan pesan itu sendiri di parameter kedua. contoh:

```
if ($some_var == "")
{
       log_message('error', 'Some variable did not contain a value.');
}
     else
{
       log_message('debug', 'Some variable was correctly set');
}

log_message('info', 'The purpose of some variable is to provide some value.');
```

log_message (info ',' Tujuan dari beberapa variabel adalah untuk memberikan beberapa nilai. ');
Ada tiga jenis pesan:

1. **Pesan Kesalahan.** Ini adalah kesalahan yang sebenarnya, seperti kesalahan PHP atau kesalahan pengguna.
2. **Pesan Debug**. Ini adalah pesan yang membantu dalam debugging. Sebagai contoh, jika kelas telah diinisialisasi, Anda bisa log ini sebagai debug info.
3. **Pesan Informasi**. Ini adalah pesan prioritas terendah, hanya memberikan informasi mengenai beberapa proses. CodeIgniter tidak native menghasilkan pesan informasi tetapi Anda mungkin ingin dalam aplikasi Anda.

Catatan: *Agar file log untuk benar-benar ditulis, "log" folder harus ditulis. Selain itu, Anda harus mengatur "threshold" untuk masuk dalam aplikasi / config / config.php. Anda mungkin, misalnya, hanya ingin pesan kesalahan untuk login, dan bukan dua jenis lainnya. Jika Anda menetapkan ke nol logging akan dinonaktifkan.*

3.9. Style dan Syntax Umum

Halaman berikut ini menjelaskan penggunaan aturan dipatuhi saat mengembangkan CodeIgniter coding.
Daftar isi

- File Format

- PHP Closing Tag
- Class and Method Naming
- Variable Names
- Commenting
- Constants
- TRUE, FALSE, and NULL
- Logical Operators
- Comparing Return Values and Typecasting
- Debugging Code
- Whitespace in Files
- Compatibility
- Class and File Names using Common Words
- Database Table Names
- One File per Class
- Whitespace
- Line Breaks
- Code Indenting
- Bracket and Parenthetic Spacing
- Localized Text
- Private Methods and Variables
- PHP Errors
- Short Open Tags
- One Statement Per Line
- Strings
- SQL Queries
- Default Function Arguments

File Format

File harus disimpan dengan Unicode (UTF-8) encoding. BOM tidak boleh digunakan.

Tidak seperti UTF-16 dan UTF-32, tidak ada urutan byte untuk menunjukkan dalam UTF-8 file disandikan, dan BOM dapat memiliki efek samping negatif di PHP mengirim output, mencegah aplikasi dari mampu untuk mengatur header sendiri. Unix akhir baris harus digunakan (LF).

Berikut adalah cara untuk menerapkan pengaturan ini dalam beberapa editor teks yang lebih umum. Petunjuk untuk editor teks Anda mungkin berbeda; memeriksa dokumentasi editor teks Anda.

- **TextMate**

1. Buka Preferensi Aplikasi
2. Klik Advanced, dan kemudian "Saving" tab
3. Dalam "File Encoding", pilih "UTF-8 (disarankan)"
4. Dalam "Line Endings", pilih "LF (dianjurkan)"
5. Opsional: Periksa "Gunakan untuk file yang ada juga" jika Anda ingin mengubah akhir baris dari file yang Anda buka dengan preferensi baru Anda.

- **BBEdit**

1. Buka Preferensi Aplikasi
2. Pilih "Text Encodings" di sebelah kiri.
3. Dalam "encoding teks default untuk dokumen baru", pilih "Unicode (UTF-8, tidak ada BOM)"
4. Opsional: Dalam "Jika encoding file tidak bisa ditebak, gunakan", pilih "Unicode (UTF-8, tidak ada BOM)"
5. Pilih "Text File" di sebelah kiri.
6. Dalam "garis default istirahat", pilih "Mac OS X dan Unix (LF)"

PHP Closing Tag

Tag penutup PHP pada dokumen PHP?> Adalah opsional untuk parser PHP. Namun, jika digunakan, setiap spasi setelah tag penutup, apakah diperkenalkan oleh pengembang, pengguna, atau aplikasi FTP, dapat menyebabkan hasil yang tidak diinginkan, kesalahan PHP, atau jika yang terakhir ditekan, halaman kosong. Untuk alasan ini, semua file PHP harus menghilangkan tag PHP penutupan, dan sebagai gantinya menggunakan komentar blok untuk menandai akhir file dan lokasi relatif

terhadap akar aplikasi. Hal ini memungkinkan Anda untuk tetap mengidentifikasi file sebagai lengkap dan tidak terpotong.

SALAH:

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

BENAR:

```
<?php
echo "Here's my code!";
/* End of file myfile.php */
/* Location: ./system/modules/mymodule/myfile.php */
```

Penamaan Class and Method

Nama kelas harus selalu dimulai dengan huruf besar. Beberapa kata-kata harus dipisahkan dengan tanda garis bawah, dan tidak CamelCased. Semua metode kelas lainnya harus seluruhnya lowercased dan diberi nama dengan jelas menunjukkan fungsinya, sebaiknya termasuk kata kerja. Cobalah untuk menghindari nama-nama yang terlalu panjang dan bertele-tele.

```
INCORRECT:
class superclass
class SuperClass
```

```
CORRECT:
class Super_class
```

```
class Super_class {

function __construct()
      {

      }
}
```

Contoh yang tidak tepat dan tepat metode penamaan:

SALAH:

```
function fileproperties ()       // tidak deskriptif dan kebutuhan
menggarisbawahi pemisah
function fileProperties ()       // tidak deskriptif dan menggunakan CamelCase

function  getfileproperties ()         // Lebih Baik! Tapi masih hilang
pemisah garis bawah
function  getFileProperties ()   // menggunakan CamelCase
Fungsi get_the_file_properties_from_the_file ()     // bertele-tele
```

BENAR:

```
get_file_properties function () // deskriptif, menekankan pemisah, dan semua
huruf kecil
```

Variable Name

Pedoman untuk variabel penamaan sangat mirip dengan yang digunakan untuk metode kelas. Yakni, variabel harus berisi hanya huruf kecil, pemisah penggunaan garis bawah, dan secara wajar bernama untuk menunjukkan tujuan dan isinya. Sangat singkat, variabel non-kata seharusnya hanya digunakan sebagai iterator in untuk () loop.

SALAH:

```
$ j = 'foo';        // Variabel huruf hanya boleh digunakan di untuk () loop
$ Str               // berisi huruf besar
$ bufferedText            // menggunakan CamelCasing, dan bisa dipersingkat
tanpa kehilangan makna semantik
$ groupid           // beberapa kata, kebutuhan menggarisbawahi pemisah
$ name_of_last_city_used  // terlalu lama
```

BENAR:

```
for ($ j = 0; $ j <10; $ j ++)
$ str
$ buffer
$ group_id
```

```
$ last_city
```

Commenting

Secara umum, kode harus berkomentar subur. Ini tidak hanya membantu menggambarkan aliran dan maksud dari kode kurang programer berpengalaman, tetapi dapat membuktikan tak ternilai ketika kembali ke bulan kode Anda sendiri di telepon. Tidak ada format yang dibutuhkan untuk komentar, tapi berikut ini adalah dianjurkan.

Gaya komentar DocBlock sebelumnya kelas dan deklarasi metode sehingga mereka dapat dijemput oleh IDE:

```
/**
* Super Class
*
* @package Package Name
* @subpackage Subpackage
* @category Category
* @author Author Name
* @link http://example.com
*/
class Super_class {
```

```
/**
* Encodes string for use in XML
*
* @access public
* @param string
* @return string
*/
function xml_encode($str)
```

Gunakan komentar baris tunggal dalam kode, meninggalkan baris kosong antara blok komentar besar dan kode.

```
// Memecah string dengan baris
$ part = explode ("\ n", $ str);
```

```
// Sebuah komentar lagi yang perlu untuk memberikan rincian lebih jelas
mengenai apa yang
// Terjadi dan mengapa bisa menggunakan beberapa komentar single-line. Cobalah
untuk
// Menyimpan lebar wajar, sekitar 70 karakter yang paling mudah untuk
// Baca. Jangan ragu untuk link ke sumber daya permanen eksternal
// Yang dapat memberikan lebih rinci:
//
// http://example.com/information_about_something/in_particular/
```

```
$ bagian = $ this-> foo ($ part);
```

Constants

Constants mengikuti pedoman yang sama seperti halnya variabel, kecuali konstanta harus selalu sepenuhnya huruf besar. Selalu menggunakan CodeIgniter konstanta saat yang tepat, yaitu SLASH, LD, RD, PATH_CACHE, dll

SALAH:

```
myConstant // hilang pemisah garis bawah dan tidak sepenuhnya huruf besar
N // tidak ada satu huruf konstanta
S_C_VER // tidak deskriptif
$ str = str_replace ('{foo}', 'bar', $ str); // Harus menggunakan LD dan RD konstanta
```

BENAR:

```
MY_CONSTANT
NEWLINE
SUPER_CLASS_VERSION
$ str = str_replace (LD.'foo'.RD, 'bar', $ str);
```

TRUE, FALSE, and NULL

Kata kunci TRUE, FALSE, dan NULL selalu harus sepenuhnya huruf besar.

SALAH:

```
if ($ foo == true)
$ bar = false;
fungsi foo ($ bar = null)
```

BENAR:

```
if ($ foo == TRUE)
$ bar = FALSE;
fungsi foo ($ bar = NULL)
Operator logis
```

Penggunaan || tidak disarankan sebagai kejelasan pada beberapa perangkat output rendah (tampak seperti nomor 11 misalnya). && Lebih disukai daripada DAN tapi entah dapat diterima, dan ruang harus selalu mendahului dan mengikuti!.

SALAH:

```
if ($ foo || $ bar)
if ($ foo and $ bar) // oke tapi tidak dianjurkan untuk aplikasi sintaks umum menyoroti
if (! $ foo)
if (! is_array ($ foo))
```

BENAR:

```
if ($ foo or $ bar)
if ($ foo && $ bar) // direkomendasikan
if (! $ foo)
if (! is_array ($ foo))
```

Comparing Return Value dan Typecasting

Beberapa fungsi PHP kembali FALSE pada kegagalan, tetapi juga mungkin memiliki nilai kembali yang valid "" atau 0, yang akan mengevaluasi ke FALSE dalam perbandingan longgar. Eksplisit dengan membandingkan jenis variabel ketika menggunakan kembali nilai-nilai tersebut dalam conditional untuk memastikan nilai kembali memang apa yang Anda harapkan, dan bukan nilai yang memiliki evaluasi longgar tipe setara.

Gunakan keketatan yang sama dalam kembali dan memeriksa variabel Anda sendiri. Gunakan === dan! == Seperlunya.

SALAH:

```
// Jika 'foo' adalah di awal string, strpos akan mengembalikan 0,
// Menghasilkan ini kondisional mengevaluasi sebagai BENAR
if (strpos ($ str, 'foo') == FALSE)

BENAR:
if (strpos ($ str, 'foo') === FALSE)

SALAH:
Fungsi build_string ($ str = "")
{
if ($ str == "") // eh-oh! Bagaimana jika SALAH atau integer 0 dilewatkan sebagai argumen?
        {
        }
}

BENAR:
Fungsi build_string ($ str = "")
{
if ($ str === "")
{

        }
}
```

Lihat juga informasi mengenai typecasting, yang bisa sangat berguna. Typecasting memiliki efek yang sedikit berbeda yang mungkin diinginkan. Ketika casting variabel sebagai string, misalnya, variabel SALAH NULL dan boolean menjadi string kosong, 0 (dan nomor lain) menjadi string angka, dan boolean BENAR menjadi "1":

```
$ str = (string) $ str; // Cor $ str sebagai string
```

Debugging Kode

Tidak ada kode debug dapat dibiarkan di tempat untuk disampaikan add-ons kecuali komentar, yaitu tidak ada var_dump (), print_r (), mati (), dan keluar () panggilan yang digunakan saat membuat add-on, kecuali mereka adalah komentar.

```
// Print_r ($ foo);
```

Spasi di File

Tidak ada spasi dapat mendahului pembukaan tag PHP atau mengikuti tag penutup PHP.

Output buffer, sehingga spasi dalam file Anda dapat menyebabkan output dimulai sebelum CodeIgniter output isinya, menyebabkan kesalahan dan ketidakmampuan untuk CodeIgniter untuk mengirim header yang tepat. Dalam contoh di bawah, pilih teks dengan mouse anda untuk mengungkapkan spasi yang salah.

SALAH:

```
<? php
// ... Ada spasi dan linebreak di atas tag pembuka PHP
// Serta spasi setelah penutupan tag PHP
?>
```

BENAR:

```
<?php
// Sampel ini tidak memiliki spasi sebelum atau sesudah pembukaan dan penutupan tag PHP
?>
```

Compatibility

Kecuali secara khusus disebutkan dalam add-on Anda dokumentasi, semua kode harus sesuai dengan PHP versi 5.1+. Selain itu, jangan gunakan fungsi PHP yang membutuhkan perpustakaan non-default untuk diinstal kecuali kode Anda berisi metode alternatif ketika fungsi tidak tersedia, atau Anda secara implisit mendokumentasikan bahwa add-on Anda membutuhkan kata perpustakaan PHP.

Class dan Nama File menggunakan Kata Umum

Saat kelas atau nama file adalah kata umum, atau mungkin sangat mungkin akan identik disebut dalam naskah lain PHP, memberikan awalan yang unik untuk membantu mencegah tabrakan. Selalu menyadari bahwa pengguna akhir Anda mungkin menjalankan add-ons atau script PHP pihak ketiga. Pilih prefix yang unik untuk identitas Anda sebagai pengembang atau perusahaan.

```
SALAH:
class Email      pi.email.php
class Xml        ext.xml.php
class Impor      mod.import.php

BENAR:
```

```
class Pre_email      pi.pre_email.php
class Pre_xml        ext.pre_xml.php
class Pre_import     mod.pre_import.php
```

Database Tabel Nama

Setiap tabel yang Anda add-on mungkin menggunakan harus menggunakan 'exp_' awalan, diikuti oleh awalan unik mengidentifikasi Anda sebagai pengembang atau perusahaan, dan kemudian pendek nama tabel deskriptif. Anda tidak perlu khawatir tentang awalan database yang digunakan pada instalasi pengguna, seperti kelas database CodeIgniter secara otomatis akan mengkonversi 'exp_' dengan apa yang benar-benar digunakan.

```
SALAH:
EMAIL_ADDRESSES // hilang kedua prefiks
pre_email_addresses // hilang exp_ awalan
exp_email_addresses // hilang awalan unik

BENAR:
exp_pre_email_addresses
```

CATATAN: *Berhati-hati bahwa MySQL memiliki batas 64 karakter untuk nama tabel. Hal ini seharusnya tidak menjadi masalah karena nama tabel yang akan melebihi ini kemungkinan akan memiliki nama yang tidak masuk akal. Misalnya, nama tabel berikut melebihi keterbatasan ini dengan satu karakter. Konyol, bukan?*
exp_pre_email_addresses_of_registered_users_in_seattle_washington

Satu File per Kelas

Menggunakan file terpisah untuk masing-masing kelas add-on Anda menggunakan, kecuali kelas yang erat kaitannya. Contoh file CodeIgniter yang berisi beberapa kelas adalah file kelas database, yang berisi baik kelas DB dan kelas DB_Cache, dan plugin Magpie, yang berisi baik Magpie dan Snoopy kelas.

Spasi

Menggunakan tab untuk spasi dalam kode Anda, bukan ruang. Hal ini mungkin tampak seperti hal kecil, tetapi menggunakan tab bukannya spasi memungkinkan pengembang melihat kode Anda untuk memiliki lekukan pada tingkat bahwa mereka lebih suka dan menyesuaikan aplikasi apa pun yang mereka gunakan. Dan sebagai manfaat samping, itu menghasilkan file (sedikit) lebih kompak, menyimpan satu karakter tab dibandingkan, katakanlah, empat karakter ruang.

Line Breaks

File harus disimpan dengan Unix line break. Ini lebih merupakan masalah bagi pengembang yang bekerja pada Windows, tetapi dalam hal apapun memastikan bahwa editor teks Anda adalah setup untuk menyimpan file dengan Unix line break.

Kode Indentasi

Gunakan Allman gaya Indentasi. Dengan pengecualian dari deklarasi kelas, kawat gigi selalu ditempatkan pada garis sendiri, dan menjorok pada tingkat yang sama seperti pernyataan kontrol yang "memiliki" mereka.

```
SALAH:
fungsi foo ($ bar) {
// ...
}

foreach ($ arr as $ key => $ val) {
// ...
}

if ($ foo == $ bar) {
// ...
} Else {
// ...
}

for ($ i = 0; $ i <10; $ i ++)
{
for ($ j = 0; $ j <10; $ j ++)
```

```
{
// ...
}
}

BENAR:
fungsi foo ($ bar)
{
// ...
}

foreach ($ arr as $ key => $ val)
{
// ...
}

if ($ foo == $ bar)
{
// ...
}
else
{
// ...
}

for ($ i = 0; $ i <10; $ i ++)
{
for ($ j = 0; $ j <10; $ j ++)
{
// ...
}
}
```

Bracket and Parenthetic Spacing

Secara umum, tanda kurung tidak harus menggunakan spasi tambahan. Pengecualian adalah bahwa ruang harus selalu mengikuti struktur kontrol PHP yang menerima argumen dengan kurung (menyatakan, do-while, elseif, untuk, foreach, jika, switch, sementara), untuk membantu membedakan mereka dari fungsi dan meningkatkan keterbacaan.

```
SALAH:
$ arr [$ foo] = 'foo';

BENAR:
$ arr [$ foo] = 'foo'; // Tanpa spasi sekitar kunci array

SALAH:
function foo ($ bar)
{

}

BENAR:
function foo ($ bar) // tanpa spasi sekitar kurung dalam deklarasi fungsi
{

}

SALAH:
foreach ($ query-> result () as $ row)

BENAR:
foreach ($ query-> result () as $ row) // ruang tunggal berikut struktur kontrol PHP, tapi tidak dalam kurung interior
```

Localized Text

Setiap teks yang output dalam panel kontrol harus menggunakan variabel bahasa dalam file lang Anda untuk memungkinkan lokalisasi.

```
SALAH:
kembali "Invalid Selection";

BENAR:
kembali $ this-> Langdon> garis ('invalid_selection');
```

Metode Private dan Variabel

Metode dan variabel yang hanya diakses secara internal oleh kelas Anda, seperti utilitas

dan penolong fungsi yang metode umum Anda gunakan untuk kode abstraksi, harus diawali dengan tanda garis bawah.

```
convert_text () // metode umum
_convert_text () // metode private
```

Kesalahan PHP

Kode harus menjalankan bebas dari kesalahan dan tidak bergantung pada peringatan dan pemberitahuan disembunyikan untuk memenuhi kebutuhan ini. Misalnya, pernah mengakses variabel yang tidak mengatur diri sendiri (seperti $ _POST kunci array) tanpa terlebih dahulu memeriksa untuk melihat bahwa isset ().

Pastikan bahwa ketika mengembangkan add-on Anda, pelaporan kesalahan diaktifkan untuk semua pengguna, dan display_errors diaktifkan dalam lingkungan PHP. Anda dapat memeriksa pengaturan ini dengan:

```
if (ini_get ('display_errors') == 1)
{
exit "Diaktifkan";
}
```

Pada beberapa server di mana display_errors dinonaktifkan, dan Anda tidak memiliki kemampuan untuk mengubah ini dalam php.ini, Anda sering dapat mengaktifkannya dengan:

```
ini_set ('display_errors', 1);
```

CATATAN: *Pengaturan pengaturan dengan ini_set () saat runtime display_errors tidak identik dengan memiliki itu diaktifkan dalam lingkungan PHP. Yaitu, tidak akan memiliki efek apapun jika script memiliki kesalahan yang fatal*

Short Open Tags

Selalu gunakan penuh tag pembuka PHP, dalam kasus server tidak memiliki short_open_tag diaktifkan.

```
SALAH:
<? echo $ foo; ?>

<? = $ foo?>

BENAR:
<? php echo $ foo; ?>
```

Satu Statement Per Baris

Jangan menggabungkan Statement pada satu baris.

```
SALAH:
$ foo = 'ini'; $ bar = 'itu'; $ kelelawar = str_replace ($ foo, $ bar, $ tas);

BENAR:
$ foo = 'ini';
$ bar = 'itu';
$ kelelawar = str_replace ($ foo, $ bar, $ tas);
```

String

Selalu gunakan string dikutip tunggal kecuali jika Anda perlu variabel diurai, dan dalam kasus di mana Anda perlu variabel diurai, penggunaan kawat gigi untuk mencegah serakah tanda parsing. Anda juga dapat menggunakan string dikutip ganda jika string berisi tanda kutip tunggal, sehingga Anda tidak harus menggunakan karakter escape.

```
SALAH:
"Saya String"              // tidak ada parsing variabel, jadi tidak ada gunanya untuk tanda kutip ganda
"Saya string $ foo"        // membutuhkan kawat gigi
"SELECT foo FROM bar WHERE baz = \ 'tas \' '   // jelek

BENAR:
'My String'
"String saya {$ foo}"
"SELECT foo FROM bar WHERE baz = 'tas'"
```

SQL Query

Kata kunci MySQL selalu dikapitalisasi: SELECT, INSERT, UPDATE, WHERE, AS, JOIN, ON, IN, dll

Break up query panjang menjadi beberapa baris untuk keterbacaan, sebaiknya melanggar untuk setiap klausa.

```
SALAH:
// Kata kunci huruf kecil dan permintaan terlalu panjang untuk
// Satu baris (... menunjukkan kelanjutan dari garis)

$ query = $ this-> db-> query ("select foo, bar, baz, foofoo, foobar sebagai raboof, foobaz
form exp_pre_email_addresses
... where foo != 'oof' and baz != ketertiban 'Zab' order by foobaz 5, 100 ")!;

BENAR:
$ query = $ this-> db-> query ("SELECT foo, bar, baz, foofoo, foobar AS raboof, foobaz
FROM exp_pre_email_addresses
WHERE foo! = 'Oof'
AND baz! = 'Zab'
ORDER BY foobaz
LIMIT 5, 100 ");
```

Argumen standar Fungsi

Setiap kali tepat, memberikan argumen default fungsi, yang membantu mencegah kesalahan PHP dengan panggilan keliru dan memberikan nilai-nilai fallback umum yang dapat menyimpan beberapa baris kode. contoh:

```
fungsi foo ($ bar = '', $ baz = FALSE)
```

3.10. Alternatif PHP Syntax untuk View File

Jika Anda tidak menggunakan template engine CodeIgniter, Anda akan menggunakan PHP murni View file Anda. Untuk meminimalkan kode PHP dalam file tersebut, dan untuk membuatnya lebih mudah untuk mengidentifikasi blok kode dianjurkan bahwa Anda menggunakan PHP – PHP sintaks alternatif untuk struktur kontrol dan pernyataan gema tag singkat. Jika Anda tidak akrab dengan sintaks ini, memungkinkan Anda untuk menghilangkan koma - koma dari kode Anda, dan menghilangkan pernyataan "echo".

Otomatis Pendek Tag Dukungan

Catatan: *Jika Anda menemukan bahwa sintaks yang dijelaskan dalam halaman ini tidak bekerja pada server Anda mungkin bahwa "tag pendek" dinonaktifkan dalam Suami file PHP Anda. CodeIgniter opsional akan menulis ulang tag pendek on-the-fly, memungkinkan Anda untuk menggunakan sintaks bahwa bahkan jika server Anda tidak mendukung itu. Fitur ini dapat diaktifkan pada file konfigurasi /* ***config.php*** *Anda.*

Perlu diketahui bahwa saat Anda menggunakan fitur ini, jika kesalahan PHP yang dihadapi dalam menemukan file Anda, pesan kesalahan dan nomor baris tidak akan akurat ditampilkan. Sebaliknya, semua kesalahan akan ditampilkan sebagai **eval () error**.

Alternatif Echos

Biasanya untuk echo, atau mencetak variabel yang akan melakukan hal ini:

<? php echo $ variabel; ?>

Dengan sintaks alternatif Anda malah bisa melakukannya dengan cara ini:

<? = $ variabel?>

Struktur Kontrol Alternatif

Kontrol struktur, seperti jika, untuk, foreach, dan sementara dapat ditulis dalam format yang sederhana juga. Berikut adalah contoh menggunakan foreach:

```
<ul>

<? php foreach ($ todo sebagai $ item):?>

<li> <? = $ item?> </ li>

? <endforeach php; ?>

</ ul>
```

Perhatikan bahwa tidak ada titik koma. Sebaliknya, penutup akhir diganti dengan **endforeach**. Setiap struktur kontrol yang tercantum di atas memiliki sintaks penutupan

yang sama: **endif**, **endfor**, **endforeach**, dan **endwhile**

Juga perhatikan bahwa alih-alih menggunakan titik koma setelah setiap struktur (kecuali yang terakhir), ada titik dua. Hal ini penting!

Berikut ini adalah contoh lain, menggunakan jika / elseif / lain. Perhatikan titik dua:

```
<? php if ($ username == 'sally'):?>

        <h3> Hi Sally </ h3>

<? php elseif ($ username == 'joe'):?>

         <h3> Hi Joe </ h3>

<? php lain:?>

         <h3> Hi pengguna yang tidak diketahui </ h3>

? <endif php; ?>
```

Class Reference

Didalam Class Reference ini terdapat beberapa class yang wajib anda ketahui dalam membangun sebuah website dengan framework CodeIgniter (CI). Kami akan membahas class class yang umum digunakan atau sering dalam pembuatan sebauh website.

Calendaring Class

- **Calendar Class**

  Class Kalender memungkinkan Anda untuk secara dinamis membuat kalender. Kalender Anda dapat diformat melalui penggunaan template kalender, yang memungkinkan kontrol 100% atas setiap aspek desain. Selain itu, Anda dapat melewatkan data ke sel kalender Anda.

Memulai Membangun Class

Seperti kebanyakan class - class lain di CodeIgniter, kelas Kalender diinisialisasi dalam controller menggunakan fungsi **$ this-> load-> library**:

$ this-> load-> library ('calendar');

Setelah dimuat, objek Kalender akan tersedia dengan menggunakan: **$ this->calendar**

Menampilkan Sebuah Kalender

Berikut ini adalah contoh yang sangat sederhana yang menunjukkan bagaimana Anda dapat menampilkan kalender:

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

echo $this->calendar->generate();
```

Kode di atas akan menghasilkan kalender untuk saat bulan / tahun berdasarkan waktu server Anda. Untuk menampilkan kalender selama satu bulan dan tahun tertentu Anda akan melewati informasi ini ke fungsi pembangkit kalender:

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

echo $this->calendar->generate(2006, 6);
```

Kode di atas akan menghasilkan kalender yang menunjukkan bulan Juni tahun 2006. Parameter pertama menentukan tahun, parameter kedua menentukan bulan.

Untuk menambahkan data ke sel kalender Anda harus melibatkan dalam menciptakan array asosiatif di mana kode yang sesuai dengan hari-hari Anda ingin mengisi dan nilai array berisi data. Array dilewatkan ke parameter ketiga fungsi pembangkit kalender. Pertimbangkan contoh ini:

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

$data = array(
      3  => 'http://example.com/news/article/2006/03/',
      7  => 'http://example.com/news/article/2006/07/',
      13 => 'http://example.com/news/article/2006/13/',
      26 => 'http://example.com/news/article/2006/26/'
      );

echo $this->calendar->generate(2006, 6, $data);
```

Menggunakan contoh di atas, hari nomor 3, 7, 13, dan 26 akan menjadi link yang

menunjuk ke URL yang Anda berikan.

Catatan: *Secara default diasumsikan bahwa array akan berisi link. Pada bagian yang menjelaskan template kalender di bawah ini Anda akan melihat bagaimana Anda dapat menyesuaikan bagaimana data dilewatkan ke sel-sel Anda ditangani sehingga Anda dapat melewati berbagai jenis informasi.*

Mengatur Preferensi Tampilan

Ada tujuh preferensi Anda dapat mengatur untuk mengontrol berbagai aspek kalender. Preferensi ditetapkan dengan melewati berbagai preferensi dalam parameter kedua dari fungsi pemuatan. Berikut adalah contoh:

```
$ prefs = array (
          'START_DAY' => 'Sabtu',
          'month_type' => 'long',
          'day_type' => 'short'
           );

$ this-> load-> library ('calendar', $ prefs);

echo $ this->calendar-> generate ();
```

Kode di atas akan mulai kalender pada hari Sabtu, gunakan "long" bulan, dan "short" nama hari. Informasi lebih lanjut mengenai preferensi bawah.

Preference	Default Value	Options	Description
template	None	None	A string containing your calendar template. See the template section below.
local_time	time()	None	A Unix timestamp corresponding to the current time.
start_day	sunday	Any week day (sunday, monday, tuesday, etc.)	Sets the day of the week the calendar should start on.
month_type	long	long, short	Determines what version of the month name to use in the header. long = January, short = Jan.
day_type	abr	long, short, abr	Determines what version of the weekday names to use in the column headers. long = Sunday, short = Sun, abr = Su.
show_next_prev	FALSE	TRUE/FALSE (boolean)	Determines whether to display links allowing you to toggle to next/previous months. See information on this feature below.
next_prev_url	None	A URL	Sets the basepath used in the next/previous calendar links.

Menampilkan Berikutnya / Sebelumnya Bulan

Untuk memungkinkan kalender Anda untuk secara dinamis kenaikan / penurunan melalui link berikutnya / sebelumnya mengharuskan Anda mengatur kode kalender mirip dengan contoh ini:

```
$prefs = array (
       'show_next_prev'=> TRUE,
       'next_prev_url' => 'http://example.com/index.php/calendar/show/'
       );
$this->load->library('calendar', $prefs);
echo $this->calendar->generate($this->uri->segment(3),$this->uri->segment(4));
```

Anda akan melihat beberapa hal tentang contoh di atas:

- Anda harus mengatur "show_next_prev" dengan TRUE.
- Anda harus menyediakan URL ke controller yang berisi kalender Anda dalam "next_prev_url" preferensi.
- Anda harus memberikan "tahun" dan "bulan" untuk fungsi pembangkit kalender melalui segmen URI mana mereka muncul
  (**Catatan**:. Kelas kalender otomatis menambahkan tahun / bulan untuk URL dasar yang anda berikan).

Membuat Template Kalender

Dengan membuat template kalender Anda memiliki kontrol 100% atas desain kalender Anda. Setiap komponen dari kalender Anda akan ditempatkan dalam sepasang pseudo-variabel seperti yang ditunjukkan di sini:

```
$prefs['template'] = '

  {table_open}<table border="0" cellpadding="0" cellspacing="0">{/table_open}

  {heading_row_start}<tr>{/heading_row_start}

  {heading_previous_cell}<th><a
href="{previous_url}">&lt;&lt;</a></th>{/heading_previous_cell}
  {heading_title_cell}<th
```

```
colspan="{colspan}">{heading}</th>{/heading_title_cell}
  {heading_next_cell}<th><a
href="{next_url}">&gt;&gt;</a></th>{/heading_next_cell}

  {heading_row_end}</tr>{/heading_row_end}

  {week_row_start}<tr>{/week_row_start}
  {week_day_cell}<td>{week_day}</td>{/week_day_cell}
  {week_row_end}</tr>{/week_row_end}

  {cal_row_start}<tr>{/cal_row_start}
  {cal_cell_start}<td>{/cal_cell_start}

  {cal_cell_content}<a href="{content}">{day}</a>{/cal_cell_content}
  {cal_cell_content_today}<div class="highlight"><a
href="{content}">{day}</a></div>{/cal_cell_content_today}

  {cal_cell_no_content}{day}{/cal_cell_no_content}
  {cal_cell_no_content_today}<div
class="highlight">{day}</div>{/cal_cell_no_content_today}

  {cal_cell_blank} {/cal_cell_blank}

  {cal_cell_end}</td>{/cal_cell_end}
  {cal_row_end}</tr>{/cal_row_end}

  {table_close}</table>{/table_close}
';

$this->load->library('calendar', $prefs);

echo $this->calendar->generate();
```

Shopping Cart Class

Class Cart mengijinkan kepada sebuah produk untuk ditambahkan ke dalam session yang aktif ,dan user akan bisa melakukan penambahan atau pengurangan hak beli produk di situs Anda. Fungsi Item ini dapat diambil dan ditampilkan dalam format standar "shopping cart" , yang memungkinkan pengguna untuk memperbarui kuantitas atau menghapus item dari keranjang.

Harap dicatat bahwa Class Cart HANYA memberikan inti "keranjang" fungsionalitas. Jadi Ini tidak memberikan pengiriman, otorisasi kartu kredit, atau komponen pengolahan lainnya.

Penting: *Class Cart menggunakan CodeIgniter Session Class untuk menyimpan informasi keranjang ke database, jadi sebelum menggunakan Class Cart Anda harus mengatur tabel database seperti yang ditunjukkan dalam Dokumentasi Sesi, dan mengatur preferensi sesi dalam file* ***application / config/config.php*** *untuk memanfaatkan database.*

Untuk menginisialisasi Belanja Kelas dalam konstruktor controller, gunakan fungsi **$ this-> load-> library:**

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

Setelah dimuat, objek Cart akan tersedia dengan menggunakan: **$ this->cart**

Catatan: *Sebuah Cart Class akan memuat dan menginisialisasi Session Class secara otomatis, jadi tanpa Anda menggunakan Session di tempat lain dalam application, Anda tidak perlu membuat Session Class baru.*

Menambahkan Item didalam Cart

Untuk menambahkan item ke keranjang belanja, hanya menggunakan array dengan data informasi produk ke **$ this->cart->insert () function**, seperti yang ditunjukkan di bawah ini:

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

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

Penting*: Yang pertama dari empat indeks array di atas (id, qty, harga, dan nama) yang*

diperlukan. Jika Anda menghilangkan salah satu dari mereka data tidak akan disimpan ke keranjang. Indeks kelima (pilihan) adalah opsional. Hal ini dimaksudkan untuk digunakan dalam kasus-kasus di mana produk Anda memiliki pilihan yang terkait dengannya. Menggunakan array untuk pilihan, seperti yang ditunjukkan di atas.

Lima indeks dilindungi adalah:

- id - Setiap produk di toko Anda harus memiliki pengenal yang unik. Biasanya ini akan menjadi "sku" atau pengenal lainnya seperti.
- qty - Kuantitas yang dibeli.
- Harga - Harga item.
- Nama - Nama item.
- Pilihan - Setiap atribut tambahan yang diperlukan untuk mengidentifikasi produk. Ini harus dilalui melalui array.

Selain lima indeks di atas, ada dua kata yang dicadangkan: rowid dan subtotal. Ini digunakan secara internal oleh kelas Cart, jadi harap jangan menggunakan kata-kata sebagai nama indeks saat memasukkan data ke dalam gerobak.

Array mungkin berisi data tambahan. Apa pun yang anda masukkan dalam array anda akan disimpan dalam sesi. Namun, yang terbaik adalah untuk standarisasi data anda di antara semua produk anda untuk membuat menampilkan informasi dalam tabel lebih mudah.

Insert() metode akan mengembalikan **$rowid** jika Anda berhasil memasukkan satu item.

Menambahkan Beberapa Produk kedalam Cart

Dengan menggunakan array multi-dimensi, seperti yang ditunjukkan di bawah ini, adalah mungkin untuk menambahkan beberapa produk ke keranjang dalam satu tindakan. Hal ini berguna dalam kasus di mana Anda ingin untuk memungkinkan orang untuk memilih dari antara beberapa item pada halaman yang sama.

```
$data = array(
      array(
```

```
            'id'    => 'sku_123ABC',
            'qty'   => 1,
            'price'       => 39.95,
            'name'  => 'T-Shirt',
            'options' => array('Size' => 'L', 'Color' => 'Red')
            ),
      array(
            'id'    => 'sku_567ZYX',
            'qty'   => 1,
            'price'       => 9.95,
            'name'  => 'Coffee Mug'
            ),
      array(
            'id'    => 'sku_965QRS',
            'qty'   => 1,
            'price'       => 29.95,
            'name'  => 'Shot Glass'
            )
      );

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

Menampilkan Cart

Untuk menampilkan Cart anda akan membuat file tampilan dengan kode yang mirip dengan yang ditunjukkan di bawah ini.

Harap dicatat bahwa contoh ini menggunakan Form Helper.

```
<?php echo form_open('path/to/controller/update/function'); ?>

<table cellpadding="6" cellspacing="1" style="width:100%" border="0">

<tr>
  <th>QTY</th>
  <th>Item Description</th>
  <th style="text-align:right">Item Price</th>
  <th style="text-align:right">Sub-Total</th>
</tr>
```

```
<?php $i = 1; ?>

<?php foreach ($this->cart->contents() as $items): ?>

        <?php echo form_hidden($i.'[rowid]', $items['rowid']); ?>

        <tr>
         <td><?php echo form_input(array('name' => $i.'[qty]', 'value' => $items['qty'],
                'maxlength' => '3', 'size' => '5')); ?></td>
          <td>
                <?php echo $items['name']; ?>

                        <?php if ($this->cart->has_options($items['rowid']) == TRUE): ?>

                                <p>
        <?php foreach
        ($this->cart->product_options($items['rowid']) as $option_name =>
$option_value): ?>

        <strong><?php echo $option_name; ?>:</strong> <?php echo $option_value; ?
><br />

        <?php endforeach; ?>
                </p>

        <?php endif; ?>

          </td>
          <td style="text-align:right"><?php echo $this->cart-
>format_number($items['price']); ?></td>
          <td style="text-align:right">$<?php echo $this->cart-
>format_number($items['subtotal']); ?></td>
        </tr>

<?php $i++; ?>

<?php endforeach; ?>

<tr>
 <td colspan="2"> </td>
 <td class="right"><strong>Total</strong></td>
```

```
 <td class="right">$<?php echo $this->cart->format_number($this->cart->total()); ?
></td>
</tr>

</table>

<p><?php echo form_submit('', 'Update your Cart'); ?></p>
```

Memperbarui Cart

Untuk memperbarui informasi dalam Cart Anda, Anda harus bisa membuat array yang berisi ID Row dan Quantity ke $ this-> cart-> update () fungsi:

Catatan: Jika Quantity diatur ke nol, item akan dihapus dari Cart.

```
$data = array(
       'rowid' => 'b99ccdf16028f015540f341130b6d8ec',
       'qty'   => 3
       );

$this->cart->update($data);

// Or a multi-dimensional array

$data = array(
       array(
              'rowid'       => 'b99ccdf16028f015540f341130b6d8ec',
              'qty'   => 3
       ),
       array(
              'rowid'       => 'xw82g9q3r495893iajdh473990rikw23',
              'qty'   => 4
       ),
       array(
       'rowid'       => 'fh4kdkkkaoe30njgoe92rkdkkobec333',
       'qty'   => 2
              )
       );

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

Apa yang dimaksud dengan ID Row? Row ID adalah sebuah identifikasi unik yang dihasilkan oleh kode Cart ketika item ditambahkan ke keranjang. Alasan ID unik dibuat adalah agar produk yang identik dengan pilihan yang berbeda dapat dikelola oleh Cart.

Sebagai contoh, katakanlah seseorang membeli dua identik t-shirt (ID produk yang sama), tetapi dalam ukuran yang berbeda. ID produk (dan atribut lainnya) akan sama untuk kedua ukuran karena kemeja yang sama. Satu-satunya perbedaan akan ukuran. Oleh karena itu Cart harus memiliki cara untuk mengidentifikasi perbedaan ini sehingga dua ukuran kemeja dapat dikelola secara independen. Ia melakukannya dengan menciptakan unik "baris ID" didasarkan pada ID produk dan pilihan yang terkait dengannya.

Keranjang Pesan Anda :

Baju A ukuran S : 50.000

Baju A ukuran M : 60.000

Dalam hampir semua kasus, memperbarui Cart akan menjadi sesuatu pengguna tidak melalui halaman "tampilan Cart", sehingga pengembang, tidak mungkin bahwa Anda akan pernah menyibukkan diri dengan "ID baris", lain kemudian memastikan Anda "Tampilan Cart" halaman berisi informasi ini dalam bidang formulir tersembunyi, dan memastikan itu akan diteruskan ke fungsi update ketika bentuk pembaruan diajukan. Perhatikan pembangunan "tampilan Cart" halaman di atas untuk informasi lebih lanjut.

Function Referensi

1. ***$ this-> cart-> insert ();***

   Memungkinkan Anda untuk menambahkan item ke keranjang belanja, seperti diuraikan di atas.

2. ***$ this-> cart-> update ();***

   Memungkinkan Anda untuk memperbarui item dalam keranjang belanja, seperti diuraikan di atas.

3. ***$ this-> cart-> total ();***

   Menampilkan jumlah total dalam keranjang.

4. ***$ this-> cart-> total_items ();***

   Menampilkan jumlah item dalam Cart.

5. ***$ this-> cart-> contents ();***

   Pengembalian array yang berisi segala sesuatu di Cart.

6. ***$ this-> cart-> has_options (rowid);***

   Pengembalian BENAR (boolean) jika baris tertentu dalam keranjang berisi opsi. Fungsi ini dirancang untuk digunakan dalam satu lingkaran dengan ***$ this-> cart-> contents ()***, karena Anda harus membuat rowid untuk fungsi ini, seperti yang ditunjukkan dalam Menampilkan contoh Cart atas.

7. ***$ this-> cart-> product_options (rowid);***

   Mengembalikan array pilihan untuk produk tertentu. Fungsi ini dirancang untuk digunakan dalam satu lingkaran dengan ***$ this-> cart-> contents ()***, karena Anda harus membuat rowid untuk fungsi ini, seperti yang ditunjukkan dalam Menampilkan contoh Cart atas.

8. ***$ this-> cart-> destroy ();***

   Memungkinkan Anda untuk menghancurkan Cart (Melakukan Reset Ulang) .Fungsi ini kemungkinan akan dipanggil saat Anda selesai memproses pesanan pelanggan.

Config Class

Config Class menyediakan sarana untuk mengambil preferensi konfigurasi. Preferensi ini dapat berasal dari file konfigurasi default (**application / config / config.php**) atau dari file konfigurasi kustom anda sendiri.

Catatan: Kelas ini diinisialisasi secara otomatis oleh sistem sehingga tidak perlu melakukannya secara manual.

Anatomi File Config

Secara default, CodeIgniter memiliki satu file konfigurasi utama, yang terletak di **application / config / config.php.** Jika Anda membuka file dengan menggunakan editor teks anda, anda akan melihat bahwa produk konfigurasi disimpan dalam sebuah array disebut $ **config**.

Anda dapat menambahkan item konfigurasi anda sendiri untuk file ini, atau jika anda lebih memilih untuk menyimpan item konfigurasi anda terpisah (dengan asumsi Anda bahkan perlu produk konfigurasi), cukup membuat file anda sendiri dan menyimpannya dalam folder **config**.

Catatan: *Jika Anda membuat file konfigurasi Anda sendiri menggunakan format yang sama seperti satu primer, menyimpan produk Anda dalam sebuah array disebut $ config. CodeIgniter dengan cerdas akan mengelola file tersebut sehingga tidak akan ada konflik meskipun array memiliki nama yang sama (dengan asumsi indeks array tidak disebutkan sama dengan yang lain).*

Memuat File Config

Catatan: *CodeIgniter secara otomatis memuat file konfigurasi utama (**application / config / config.php)**, sehingga anda hanya perlu memuat file konfigurasi jika anda telah membuat anda sendiri.*

Ada dua cara untuk memuat file konfigurasi:

1. Manual Loading

Untuk memuat satu file konfigurasi kustom anda, anda akan menggunakan fungsi berikut dalam controller yang membutuhkannya:

$ this-> config> load ('nama file');

Dimana nama file adalah nama file konfigurasi Anda, tanpa ekstensi file php.

Jika Anda perlu memuat beberapa file konfigurasi biasanya mereka akan digabung menjadi satu master konfigurasi array . Namanya kesamaa/ konflik dapat terjadi, namun, jika Anda telah identik bernama indeks array dalam file konfigurasi yang berbeda. Untuk menghindari tabrakan nama anda dapat mengatur parameter kedua TRUE dan semua file konfigurasi akan disimpan dalam indeks array yang sesuai dengan nama file konfigurasi. contoh:

// Disimpan dalam array dengan prototipe ini:

$ this-> config ['blog_settings'] = $ config

$ this-> config> load ('blog_settings', TRUE);

Silakan lihat bagian yang berjudul Mengambil Config Produk di bawah ini untuk belajar bagaimana untuk mengambil produk konfigurasi diatur dengan cara ini.

Parameter ketiga memungkinkan Anda untuk menekan kesalahan dalam hal file konfigurasi tidak ada:

$ this-> config> load ('blog_settings', FALSE, TRUE);

2. Auto-loading

Jika Anda menemukan bahwa Anda memerlukan file konfigurasi tertentu secara global, Anda dapat memilikinya dimuat secara otomatis oleh sistem. Untuk melakukan hal ini, buka file autoload.php, yang terletak di application / config / autoload.php, dan menambahkan file konfigurasi seperti ditunjukkan dalam file.

Mengambil Config Produk

Untuk mengambil item dari file konfigurasi Anda, gunakan fungsi berikut:

$ this-> config> item ('nama item');

Dimana nama item adalah $ indeks config array yang akan diambil. Misalnya, untuk mengambil pilihan bahasa Anda, Anda akan melakukan hal ini:

$ lang = $ this-> config> item ('bahasa');

Fungsi mengembalikan FALSE (boolean) jika item yang Anda mencoba untuk mengambil tidak ada.

Jika Anda menggunakan parameter kedua dari **this-> config> load** function $ untuk menetapkan produk konfigurasi anda ke indeks tertentu. Anda dapat mengambilnya dengan menentukan nama indeks pada parameter kedua dari **$ this-> config > item ()** function.

Contoh:

```
// Memuat file konfigurasi bernama blog_settings.php dan memberikan ke indeks
bernama "blog_settings"
$this->config->load('blog_settings', TRUE);

// Ambil item konfigurasi bernama site_name terkandung dalam blog_settings
Array
$site_name = $this->config->item('site_name', 'blog_settings');
```

```
// Cara alternatif untuk menentukan item yang sama:
$blog_config = $this->config->item('blog_settings');
$site_name = $blog_config['site_name'];
```

Mengatur Item Config

Jika Anda ingin untuk secara dinamis mengatur item konfigurasi atau mengubah yang sudah ada, Anda dapat melakukannya dengan menggunakan:

$ this-> config> set_item ('ITEM_NAME', 'item_value');

Dimana ITEM_NAME adalah $ indeks config Array yang ingin anda ubah, dan nilai sebuah item_value .

Environment

Anda dapat memasukkan file konfigurasi yang berbeda tergantung pada kondisi saat itu. ENVIRONMENT yang konstan didefinisikan dalam index.php, dan dijelaskan secara rinci pada bagian Penanganan Environment.

Untuk membuat file konfigurasi lingkungan tertentu, membuat atau menyalin file konfigurasi dalam **application/config/{ ENVIRONMENT} / { FILENAME }.php**

Misalnya, untuk membuat produksi hanya config.php, Anda akan:

1. Buat aplikasi direktori / config / produksi /
2. Salin config.php yang ada ke dalam direktori di atas
3. Mengedit application / config / produksi / config.php sehingga berisi pengaturan produksi Anda

Ketika Anda mengatur Environment konstan 'produksi', pengaturan hanya produksi yang baru config.php anda yang akan dimuat.

Anda dapat menempatkan file-file konfigurasi berikut dalam folder lingkungan spesifik:

- Standar file konfigurasi CodeIgniter
- File konfigurasi kustom Anda sendiri

Catatan: *CodeIgniter selalu mencoba untuk memuat file-file konfigurasi untuk*

lingkungan saat ini terlebih dahulu. Jika file tidak ada, file konfigurasi global (yaitu, satu di aplikasi / config /) dimuat. Ini berarti Anda tidak diwajibkan untuk menempatkan semua file konfigurasi Anda dalam folder lingkungan - hanya file yang berubah per lingkungan.

Fungsi helper

Kelas config memiliki fungsi pembantu berikut:

- **$ this-> config> site_url ();**

Fungsi ini mengambil URL ke situs Anda, bersama dengan "index" nilai yang Anda tetapkan dalam file konfigurasi.

- **$ this-> config> base_url ();**

Fungsi ini mengambil URL ke situs Anda, ditambah jalur opsional seperti untuk stylesheet atau gambar.

Kedua fungsi di atas biasanya diakses melalui fungsi yang sesuai dalam Helper URL.

- **$ this-> config> system_url ();**

Fungsi ini mengambil URL ke folder sistem anda.

Email Class

Email Kelas CodeIgniter yang kuat mendukung fitur berikut:

1. Multiple Protocols: Mail, Sendmail, and SMTP
2. Multiple recipients
3. CC and BCCs
4. HTML or Plaintext email
5. Attachments
6. Word wrapping
7. Priorities
8. BCC Batch Mode, yang memungkinkan email yang besar daftar untuk dipecah menjadi batch BCC kecil.
9. Email Debugging Tools

Mengirim Email

Mengirim email tidak hanya sederhana, tetapi anda dapat mengkonfigurasinya dengan cepat atau mengatur preferensi anda dalam file konfigurasi.

Berikut ini adalah contoh dasar menunjukkan bagaimana Anda bisa mengirim email.

Catatan: Contoh ini mengasumsikan Anda mengirim email dari salah satu pengendali Anda.

```
$this->load->library('email');
$this->email->from('your@example.com', 'Your Name');
$this->email->to('someone@example.com');
$this->email->cc('another@another-example.com');
$this->email->bcc('them@their-example.com');

$this->email->subject('Email Test');
$this->email->message('Testing the email class.');

$this->email->send();
echo $this->email->print_debugger();
```

Mengatur Preferensi Email

Ada 17 preferensi yang berbeda yang tersedia untuk menyesuaikan bagaimana pesan email anda dikirim. Anda dapat mengatur secara manual seperti dijelaskan di sini, atau secara otomatis melalui preferensi disimpan dalam file konfigurasi Anda, untuk lebih jelasnya akan dijelaskan di bawah ini:

Preferensi ditetapkan dengan melewatkan sebuah array nilai preferensi untuk fungsi inisialisasi email . Berikut adalah contoh bagaimana Anda dapat mengatur beberapa preferensi:

```
$config['protocol'] = 'sendmail';
$config['mailpath'] = '/usr/sbin/sendmail';
$config['charset'] = 'iso-8859-1';
$config['wordwrap'] = TRUE;

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

Catatan: *Sebagian besar preferensi memiliki nilai default yang akan digunakan jika Anda tidak menetapkan mereka.*

Mengatur Preferensi Email di File Config

Jika Anda memilih untuk tidak menetapkan preferensi menggunakan metode di atas, Anda bisa menempatkan mereka ke dalam file konfigurasi. Cukup membuat file baru

yang disebut email.php, tambahkan array $ config dalam file tersebut. Kemudian simpan file pada config / email.php dan akan digunakan secara otomatis. Anda tidak perlu menggunakan **$ this-> email-> initialize () function** jika Anda menyimpan preferensi Anda dalam file konfigurasi.

Preferensi email

Berikut ini adalah daftar semua preferensi yang dapat diatur saat mengirim email.

Preference	Default Value	Options	Description
useragent	CodeIgniter	None	The "user agent".
protocol	mail	mail, sendmail, or smtp	The mail sending protocol.
mailpath	usr/sbin/sendmail	None	The server path to Sendmail.
smtp_host	No Default	None	SMTP Server Address.
smtp_user	No Default	None	SMTP Username.
smtp_pass	No Default	None	SMTP Password.
smtp_port	25	None	SMTP Port.
smtp_timeout	5	None	SMTP Timeout (in seconds).
wordwrap	TRUE	TRUE or FALSE (boolean)	Enable word-wrap.
wrapchars	76		Character count to wrap at.
mailtype	text	text or html	Type of mail. If you send HTML email you must send it as a complete web page. Make sure you don't have any relative links or relative image paths otherwise they will not work.
charset	utf-8		Character set (utf-8, iso-8859-1, etc.).
validate	FALSE	TRUE or FALSE (boolean)	Whether to validate the email address.
priority	3	1, 2, 3, 4, 5	Email Priority. 1 = highest. 5 = lowest. 3 = normal.
crlf	\n	"\r\n" or	Newline character. (Use "\r\n" to comply with RFC 822).

newline	\n	"\n" or "\r" "\r\n" or "\n" or "\r"	Newline character. (Use "\r\n" to comply with RFC 822).
bcc_batch_mode	FALSE	TRUE or FALSE (boolean)	Enable BCC Batch Mode.
bcc_batch_size	200	None	Number of emails in each BCC batch.

Email Fungsi Referensi

$ this-> email-> from ()

Mengatur alamat email dan nama orang yang mengirim email:

```
$this->email->from('you@example.com', 'Your Name');
```

$ this-> email-> reply_to ()

Mengatur balasan-untuk mengatasi. Jika informasi tidak memberikan informasi dalam "dari" fungsi digunakan. contoh:

```
$this->email->reply_to('you@example.com', 'Your Name');
```

$ this-> Email-> to ()

Mengatur alamat email (s) penerima (s). Bisa satu email, daftar dipisahkan koma atau array:

```
$this->email->to('someone@example.com');
$this->email->to('one@example.com, two@example.com, three@example.com');
$list = array('one@example.com', 'two@example.com', 'three@example.com');
$this->email->to($list);
```

$ this-> email-> cc ()

Mengatur alamat email CC (s). Sama seperti "untuk", bisa menjadi satu email, daftar dipisahkan koma atau array.

$ this-> email-> bcc ()

Mengatur alamat email BCC (s). Sama seperti "untuk", bisa menjadi satu email, daftar dipisahkan koma atau array.

$ this-> email-> subject ()

Mengatur subjek email:

```
$this->email->subject('This is my subject');
```

$ this-> email->message()

Mengatur isi pesan email:

```
$this->email->message('This is my message');
```

$ this-> Email-> set_alt_message ()

Mengatur alternatif isi pesan email:

```
$this->email->set_alt_message('This is the alternative message');
```

Ini adalah string pesan opsional yang dapat digunakan jika Anda mengirim email format HTML . Ini memungkinkan Anda menentukan pesan alternatif tanpa HTML format yang ditambahkan ke string header untuk orang-orang yang tidak menerima email HTML. Jika Anda tidak menetapkan pesan Anda sendiri. CodeIgniter akan mengekstrak pesan dari email HTML Anda dan strip tag.

$ this-> email-> clear ()

Menginisialisasi semua variabel email ke keadaan kosong. Fungsi ini dimaksudkan untuk digunakan jika Anda menjalankan email pengiriman fungsi dalam satu lingkaran, yang memungkinkan data yang akan berulang antara siklus.

```
foreach ($list as $name => $address)
{
	$this->email->clear();

	$this->email->to($address);
	$this->email->from('your@example.com');
	$this->email->subject('Here is your info '.$name);
	$this->email->message('Hi '.$name.' Here is the info you requested.');
	$this->email->send();
}
```

Jika Anda mengatur parameter TRUE lampiran akan dihapus juga:

```
$this->email->clear(TRUE);
```

$ this->email->send()

Email mengirim fungsi. Pengembalian boolean TRUE atau FALSE berdasarkan keberhasilan atau kegagalan, yang memungkinkan untuk digunakan bersyarat:

```
if (!$this-> email-> send ())
{
       // Hasilkan kesalahan
}
$this-> email-> attach ()
```

Memungkinkan Anda untuk mengirim lampiran. Letakkan file path / nama pada parameter pertama. Catatan: Gunakan path file, bukan URL. Untuk beberapa lampiran menggunakan fungsi beberapa kali. Sebagai contoh:

```
$this->email->attach('/path/to/photo1.jpg');
$this->email->attach('/path/to/photo2.jpg');
$this->email->attach('/path/to/photo3.jpg');

$this->email->send();
```

$ this-> Email-> print_debugger ()

Mengembalikan string yang berisi pesan server, header email, dan messsage email. Berguna untuk debugging.

Melebihi Kata Wrapping

Jika Anda memiliki kata wrapping diaktifkan (dianjurkan untuk mematuhi RFC 822) dan Anda memiliki link yang sangat panjang di email Anda itu bisa di wrapping juga, menyebabkan ia menjadi un-diklik oleh orang yang menerima itu. CodeIgniter memungkinkan Anda secara manual menimpa wrapping kata dalam bagian dari pesan Anda seperti ini:

```
The text of your email that gets wrapped normally.

{unwrap}http://example.com/a_long_link_that_should_not_be_wrapped.html{/unwrap
```

```
}

More text that will be wrapped normally.
```

Encryption Class

Encryption Class menyediakan enkripsi data dua arah. Hal ini dienkripsi menggunakan perpustakaan Mcrypt. Enkripsi Kelas membutuhkan ekstensi Mcrypt untuk menjalankan.

Mengatur Kunci

Key adalah sepotong informasi yang mengontrol proses kriptografi dan izin string terenkripsi yang akan diterjemahkan. Bahkan, kunci yang Anda pilih akan memberikan satu-satunya cara untuk memecahkan kode data yang telah dienkripsi dengan kunci itu, jadi tidak diharuskan Anda membuat kode kunci dengan hati-hati, Anda diharuskan untuk tidak mengubahnya jika Anda berniat menggunakannya untuk data sistem.

Tak usah dikatakan bahwa anda ditugaskan untuk melindungi password dengan hati-hati. Haruskah seseorang mendapatkan akses secara bebas dengan kunci kode anda ?. Data akan mudah diterjemahkan. Jika server Anda tidak sepenuhnya di bawah kendali Anda .Jadi tidak mungkin anda menjamin memberikan keamanan yang tinggi pada sebuah website tanpa adanya keingin tahuan anda kegunaan keamanan yang tinggi, seperti menyimpan nomor kartu kredit.

Untuk mengambil keuntungan maksimum dari algoritma enkripsi, kunci Anda harus 32 karakter (128 bit). Kuncinya harus dengan bilangan acak seperti yang Anda susun, dengan angka dan huruf besar dan huruf kecil. Kunci Anda tidak harus menjadi string teks sederhana. Agar kriptografi aman itu perlu sebagai acak mungkin.

Kunci Anda dapat disimpan dalam **application / config / config.php**, atau Anda dapat merancang mekanisme penyimpanan menurut anda dan membentuk kunci dinamis ketika encoding / decoding.

Untuk menyimpan kunci untuk aplikasi Anda / config / config.php, buka file tersebut dan menetapkan:

```
$config['encryption_key'] = "YOUR KEY";
$config['encryption_key'] = "S3lal7yiI85AvFrsQm"; // diisi kode acak anda
```

Pesan panjang

Sangat penting bagi Anda untuk mengetahui bahwa pesan disandikan fungsi enkripsi akan menghasilkan sekitar 2,6 kali lebih lama dari pesan asli. Sebagai contoh, jika Anda mengenkripsi string "saya yang super Data rahasia", yang merupakan 21 karakter, Anda akan berakhir dengan string disandikan yang kira-kira 55 karakter (kita mengatakan "kasar" karena string kenaikan panjang dikodekan dalam 64 cluster bit, sehingga tidak persis linear). Simpanlah informasi ini dalam pikiran ketika memilih mekanisme penyimpanan data Anda. Cookies, misalnya, hanya bisa menampung 4K informasi.

Contoh : lembaha airputih / hjaso67da97d9ad97datda57da8698gduad88798ada57gdg7fdwvJHVJ6t

Seperti kebanyakan kelas-kelas lain di CodeIgniter, kelas Enkripsi diinisialisasi dalam controller menggunakan fungsi **$ this-> load-> library:**

$ this-> load-> library ('encrypt');

Setelah dimuat, objek Library encrypt akan tersedia dengan menggunakan: $ this-> Encrypt

$ this-> encrypt-> encode ()

Melakukan enkripsi data dan mengembalikannya sebagai string. Contoh:

```
$msg = 'My secret message';
$encrypted_string = $this->encrypt->encode($msg);
```

Anda dapat melewati kunci enkripsi Anda melalui parameter kedua jika Anda tidak ingin menggunakan salah satu di file konfigurasi Anda:

```
$msg = 'My secret message';
$key = 'super-secret-key';

$encrypted_string = $this->encrypt->encode($msg, $key);
```

$this->encrypt->decode()

Mendekripsi string disandikan. Contoh:

```
$encrypted_string = 'APANtByIGI1BpVXZTJgcsAG8GZl8pdwwa84';
$plaintext_string = $this->encrypt->decode($encrypted_string);
```

Anda dapat melewati kunci enkripsi Anda melalui parameter kedua jika Anda tidak ingin menggunakan salah satu di file konfigurasi Anda:

```
$msg = 'My secret message';
$key = 'super-secret-key';

$encrypted_string = $this->encrypt->decode($msg, $key);
```

$ this-> encrypt-> set_cipher ();

Memungkinkan Anda untuk mengatur cipher Mcrypt. Secara default menggunakan MCRYPT_RIJNDAEL_256. Contoh:

```
$this->encrypt->set_cipher(MCRYPT_BLOWFISH);
```

Silahkan kunjungi php.net untuk daftar cipher yang tersedia.

Jika Anda ingin menguji secara manual apakah server Anda mendukung Mcrypt Anda dapat menggunakan:

```
echo ( ! function_exists('mcrypt_encrypt')) ? 'Nope' : 'Yup';
```

$ this-> encrypt-> set_mode ();

Memungkinkan Anda untuk mengatur modus Mcrypt. Secara default menggunakan MCRYPT_MODE_CBC. contoh:

```
$this->encrypt->set_mode(MCRYPT_MODE_CFB);
```

$ this-> encrypt-> sha1 ();

SHA1 fungsi encoding. Memberikan string dan akan mengembalikan 160 bit salah satu

cara hash. Catatan: SHA1, seperti MD5 adalah non-decodable. contoh:

```
$hash = $this->encrypt->sha1('Some string');
```

Banyak instalasi PHP memiliki dukungan SHA1 secara default, jadi jika semua yang Anda butuhkan adalah untuk menyandikan hash itu sederhana untuk menggunakan fungsi asli:

```
$hash = sha1('Some string');
```

Jika server Anda tidak mendukung SHA1 Anda dapat menggunakan fungsi yang disediakan.

```
$ this-> encrypt-> encode_from_legacy ($ orig_data, $ legacy_mode =
MCRYPT_MODE_ECB, $ key = '');
```

Data memungkinkan Anda untuk kembali encode yang awalnya dienkripsi dengan CodeIgniter 1.x agar kompatibel dengan perpustakaan Enkripsi di CodeIgniter 2.x. Hal ini hanya diperlukan untuk menggunakan metode ini jika Anda telah mengenkripsi data yang tersimpan secara permanen seperti dalam sebuah file atau database dan berada pada server yang mendukung Mcrypt. "Cahaya" menggunakan enkripsi seperti data sesi dienkripsi atau sementara dienkripsi flashdata tidak memerlukan intervensi dari pihak Anda. Namun, Sesi dienkripsi ada akan hancur karena data dienkripsi sebelum 2.x tidak akan diterjemahkan.

Mengapa hanya metode untuk mengkodekan kembali data bukan mempertahankan metode warisan untuk kedua encoding dan decoding? Algoritma di perpustakaan Enkripsi telah meningkat di CodeIgniter 2.x baik untuk kinerja dan keamanan, dan kami tidak ingin mendorong terus menggunakan metode yang lebih tua. Anda tentu saja dapat memperpanjang Library Enkripsi jika Anda inginkan dan mengganti metode baru dengan yang lama dan mempertahankan kompatibilitas mulus dengan CodeIgniter 1.x dienkripsi data, tapi ini keputusan yang pengembang harus membuat hati-hati dan

sengaja, jika sama sekali.

$ new_data = $ this-> encrypt-> encode_from_legacy ($ old_encrypted_string);

Parameter	Default	Description
$orig_data	n/a	The original encrypted data from CodeIgniter 1.x's Encryption library
$legacy_mode	MCRYPT_MODE_ECB	The Mcrypt mode that was used to generate the original encrypted data. CodeIgniter 1.x's default was MCRYPT_MODE_ECB, and it will assume that to be the case unless overridden by this parameter.
$key	n/a	The encryption key. This it typically specified in your config file as outlined above.

Form Validation

Sebelum menjelaskan pendekatan CodeIgniter untuk validasi data, mari kita menggambarkan skenario yang ideal:

1. Suatu bentuk ditampilkan.
2. Anda mengisinya dan mengirimkannya.
3. Jika Anda mengajukan sesuatu yang tidak valid, atau mungkin melewatkan item yang diperlukan, formulir tersebut redisplayed berisi data Anda bersama dengan pesan kesalahan yang menjelaskan masalah.
4. Proses ini berlanjut sampai Anda telah mengirimkan formulir yang valid.

Pada sisi penerima, script harus:

- Periksa data yang diperlukan.
- Pastikan bahwa data dari jenis yang tepat, dan memenuhi kriteria yang benar. Sebagai contoh, jika username yang disampaikan harus divalidasi mengandung karakter hanya diizinkan. Ini harus menjadi panjang minimum, dan tidak melebihi panjang maksimum. Username tidak bisa menjadi username yang ada orang lain, atau bahkan mungkin kata reserved. Dan lain-lain
- Membersihkan data untuk keamanan.
- Pra-format data jika diperlukan (Apakah data perlu dipangkas? HTML dikodekan? Dll)

- Persiapan data untuk dimasukkan dalam database.

  Meskipun tidak ada yang sangat kompleks tentang proses di atas, biasanya membutuhkan sejumlah besar kode, dan untuk menampilkan pesan error, berbagai struktur kontrol biasanya ditempatkan dalam bentuk HTML. Validasi form, sementara sederhana untuk membuat, umumnya sangat berantakan dan membosankan untuk melaksanakan.

Form Tutorial Validasi

Berikut ini adalah "tangan di atas" tutorial untuk melaksanakan CodeIgniters Validasi Form.

Dalam rangka melaksanakan validasi form Anda akan membutuhkan tiga hal:

1. Sebuah View file yang berisi formulir.
2. Sebuah View file yang berisi "sukses" pesan yang akan ditampilkan pada sukses pengajuan.
3. Fungsi controller untuk menerima dan memproses data yang diajukan.

Mari kita buat tiga hal, menggunakan sign-up bentuk anggota sebagai contoh.

Formulir

Menggunakan editor teks, membuat bentuk yang disebut myform.php. Di dalamnya, menempatkan kode ini dan simpan ke **application / views / folder**:

```
<html>
<head>
<title>My Form</title>
</head>
<body>

<?php echo validation_errors(); ?>

<?php echo form_open('form'); ?>

<h5>Username</h5>
<input type="text" name="username" value="" size="50" />
```

```
<h5>Password</h5>
<input type="text" name="password" value="" size="50" />

<h5>Password Confirm</h5>
<input type="text" name="passconf" value="" size="50" />

<h5>Email Address</h5>
<input type="text" name="email" value="" size="50" />

<div><input type="submit" value="Submit" /></div>

</form>

</body>
</html>
```

Success Page

Menggunakan editor teks, membuat bentuk yang disebut formsuccess.php. Di dalamnya, menempatkan kode ini dan simpan ke **application / views / folder**:

```
<html>
<head>
<title>My Form</title>
</head>
<body>

<h3>Your form was successfully submitted!</h3>

<p><?php echo anchor('form', 'Try it again!'); ?></p>

</body>
</html>
```

Controller

Menggunakan editor teks, membuat controller yang disebut form.php. Di dalamnya,

menempatkan kode ini dan simpan ke **application / controllers / folder**:

```
<?php

class Form extends CI_Controller {

	function index()
	{
		$this->load->helper(array('form', 'url'));

		$this->load->library('form_validation');

		if ($this->form_validation->run() == FALSE)
		{
			$this->load->view('myform');
		}
		else
		{
			$this->load->view('formsuccess');
		}
	}
}
?>
```

Percobaan

Untuk mencoba formulir Anda, kunjungi situs Anda menggunakan URL yang mirip dengan yang satu ini:

```
example.com/index.php/form/
```

Jika Anda mengirimkan formulir Anda hanya akan melihat bentuk reload. Itu karena Anda belum membuat aturan validasi belum.

Karena Anda belum diberitahu kelas Validasi Form untuk memvalidasi apa pun, ia mengembalikan FALSE (boolean false) secara default. **run ()** function hanya mengembalikan TRUE jika telah berhasil menerapkan aturan tanpa salah satu dari

mereka gagal.

Penjelasan

Anda akan melihat beberapa hal tentang halaman di atas:

Bentuk (myform.php) adalah bentuk web standar dengan beberapa pengecualian:

1. Ini menggunakan bentuk pembantu untuk membuat pembukaan bentuk. Secara teknis, hal ini tidak diperlukan. Anda bisa membuat bentuk menggunakan HTML standar. Namun, manfaat menggunakan penolong adalah bahwa ia menghasilkan URL tindakan untuk Anda, berdasarkan pada URL dalam file konfigurasi Anda. Hal ini membuat aplikasi Anda lebih portabel dalam hal URL Anda berubah.
2. Di bagian atas formulir Anda akan melihat fungsi panggilan berikut:

```
<?php echo validation_errors(); ?>
```

Fungsi ini akan mengembalikan pesan kesalahan dikirim kembali oleh validator. Jika tidak ada pesan yang mengembalikan sebuah string kosong.

Controller (form.php) memiliki satu fungsi: index (). Fungsi ini menginisialisasi kelas validasi dan beban bentuk helper dan URL helper yang digunakan oleh file view Anda. Hal ini juga menjalankan rutinitas validasi. Berdasarkan apakah validasi berhasil itu baik menyajikan bentuk atau halaman keberhasilan.

Mengatur Aturan Validasi

CodeIgniter memungkinkan Anda menetapkan sebanyak aturan validasi yang anda butuhkan untuk bidang tertentu, menuju ke dalam rangka HTML, dan bahkan memungkinkan anda persiapan dan pra-proses data lapangan pada saat yang sama. Untuk menetapkan aturan validasi anda akan menggunakan set_rules () function:

```
$this->form_validation->set_rules();
```

Fungsi di atas mengambil tiga parameter sebagai masukan:

1. Nama field - nama yang tepat Anda diberi formulir isian.
2. "manusia" nama untuk bidang ini, yang akan dimasukkan ke dalam pesan kesalahan. Sebagai contoh, jika bidang Anda adalah bernama "user" Anda mungkin memberikan nama manusia "Username". Catatan: Jika Anda ingin nama field yang akan disimpan dalam file bahasa, silakan lihat Translating Nama Field.
3. Validasi aturan untuk bidang formulir ini.

Berikut adalah contoh. Dalam controller (form.php), tambahkan kode ini tepat di bawah fungsi inisialisasi validasi:

```
$this->form_validation->set_rules('username', 'Username', 'required');
$this->form_validation->set_rules('password', 'Password', 'required');
$this->form_validation->set_rules('passconf', 'Password Confirmation',
'required');
$this->form_validation->set_rules('email', 'Email', 'required');
```

Controller Anda sekarang harus terlihat seperti ini:

```
<?php

class Form extends CI_Controller {

	function index()
	{
		$this->load->helper(array('form', 'url'));

		$this->load->library('form_validation');

		$this->form_validation->set_rules('username', 'Username',
'required');
		$this->form_validation->set_rules('password', 'Password',
```

```
'required');
            $this->form_validation->set_rules('passconf', 'Password
Confirmation', 'required');
            $this->form_validation->set_rules('email', 'Email', 'required');

            if ($this->form_validation->run() == FALSE)
            {
                  $this->load->view('myform');
            }
            else
            {
                  $this->load->view('formsuccess');
            }
      }
}
?>
```

Menetapkan Aturan Menggunakan Array

Sebelum pindah perlu dicatat bahwa fungsi pengaturan aturan dapat melewati array jika Anda lebih memilih untuk mengatur semua aturan dalam satu tindakan. Jika Anda menggunakan pendekatan ini Anda harus nama kunci array Anda seperti ditunjukkan:

```
$config = array(
            array(
            'field'        => 'username',
            'label'   => 'Username',
            'rules'   => 'required'
            ),
      array(
            'field'   => 'password',
            'label'   => 'Password',
            'rules'   => 'required'
      ),
      array(
            'field'        => 'passconf',
            'label'   => 'Password Confirmation',
            'rules'   => 'required'
      ),
      array(
```

```
            'field'   => 'email',
            'label'   => 'Email',
            'rules'        => 'required'
            )
      );

$this->form_validation->set_rules($config);
```

Cascading Rules

CodeIgniter memungkinkan membuat beberapa aturan bersama-sama. Mari kita mencobanya. Mengubah aturan dalam parameter ketiga fungsi pengaturan aturan, seperti ini:

```
$this->form_validation->set_rules('username', 'Username', 'required|
min_length[5]|max_length[12]|is_unique[users.username]');
$this->form_validation->set_rules('password', 'Password', 'required|
matches[passconf]');
$this->form_validation->set_rules('passconf', 'Password Confirmation',
'required');
$this->form_validation->set_rules('email', 'Email', 'required|valid_email|
is_unique[users.email]');
```

Kode di atas menetapkan aturan berikut:

1. Username lapangan ada lebih pendek dari 5 karakter dan tidak lebih dari 12.
2. Bidang sandi harus sesuai dengan bidang konfirmasi sandi.
3. Bidang email harus berisi alamat email yang valid.

Cobalah! Mengirimkan formulir Anda tanpa data yang tepat dan Anda akan melihat pesan kesalahan baru yang sesuai dengan aturan baru. Ada banyak aturan yang tersedia yang dapat Anda baca di referensi validasi.

Prepping Data

Selain fungsi validasi seperti yang kita gunakan di atas, Anda juga dapat persiapan data Anda dalam berbagai cara. Sebagai contoh, Anda dapat mengatur aturan seperti ini:

```
$this->form_validation->set_rules('username', 'Username', 'trim|required|
min_length[5]|max_length[12]|xss_clean');
$this->form_validation->set_rules('password', 'Password', 'trim|required|
matches[passconf]|md5');
$this->form_validation->set_rules('passconf', 'Password Confirmation', 'trim|
required');
$this->form_validation->set_rules('email', 'Email', 'trim|required|
valid_email');
```

Dalam contoh di atas, kita "pemangkasan" bagian, mengubah password untuk MD5, dan berjalan username melalui "xss_clean" fungsi, yang menghilangkan data berbahaya.

Fungsi PHP asli yang menerima satu parameter dapat digunakan sebagai aturan, seperti htmlspecialchars, langsing, MD5, dll

Catatan: Anda biasanya akan ingin menggunakan fungsi prepping setelah peraturan validasi jadi jika ada kesalahan, data asli akan ditampilkan dalam form.

Pengisian Ulang formulir

Sejauh ini kita hanya telah berurusan dengan kesalahan. Sudah waktunya untuk terisi kembali formulir isian dengan data yang diajukan. CodeIgniter menawarkan beberapa fungsi pembantu yang memungkinkan Anda untuk melakukan hal ini. Yang akan Anda gunakan paling sering adalah:

set_value('field name')

Buka **myform.php** tampilan file dan memperbarui nilai dalam setiap bidang menggunakan set_value () fungsi:

Jangan lupa untuk menyertakan setiap nama field dalam set_value yang () fungsi!

```
<html>
<head>
```

```
<title>My Form</title>
</head>
<body>

<?php echo validation_errors(); ?>

<?php echo form_open('form'); ?>

<h5>Username</h5>
<input type="text" name="username" value="<?php echo set_value('username'); ?>"
size="50" />

<h5>Password</h5>
<input type="text" name="password" value="<?php echo set_value('password'); ?>"
size="50" />

<h5>Password Confirm</h5>
<input type="text" name="passconf" value="<?php echo set_value('passconf'); ?>"
size="50" />

<h5>Email Address</h5>
<input type="text" name="email" value="<?php echo set_value('email'); ?>" size="50" />

<div><input type="submit" value="Submit" /></div>

</form>

</body>
</html>
```

Catatan Penting: Jika Anda menggunakan sebuah array sebagai nama bidang formulir, Anda harus menyediakan sebagai sebuah array ke fungsi. contoh:

```
<input type = "text" name = "warna []" value = "<php echo set_value ('warna []');?>" size =
"50" />
```

Untuk info lebih lanjut silakan lihat Array Menggunakan sebagai bagian Nama

Lapangan bawah.

Callback: Fungsi Validasi Anda sendiri

Sistem validasi mendukung callback untuk fungsi validasi Anda sendiri. Hal ini memungkinkan Anda untuk memperpanjang kelas validasi untuk memenuhi kebutuhan Anda. Sebagai contoh, jika Anda perlu untuk menjalankan query database untuk melihat apakah pengguna memilih nama pengguna yang unik, Anda dapat membuat fungsi callback yang melakukan itu. Mari kita membuat contoh ini.

Dalam controller Anda, mengubah "username" aturan untuk ini:

```
$this->form_validation->set_rules('username', 'Username', 'callback_username_check');
```

Kemudian tambahkan fungsi baru yang disebut username_check ke controller. Berikut adalah cara controller sekarang harus melihat:

```
<?php

class Form extends CI_Controller {

	public function index()
	{
		$this->load->helper(array('form', 'url'));

		$this->load->library('form_validation');

		$this->form_validation->set_rules('username', 'Username',
'callback_username_check');
		$this->form_validation->set_rules('password', 'Password', 'required');
		$this->form_validation->set_rules('passconf', 'Password Confirmation',
'required');
		$this->form_validation->set_rules('email', 'Email', 'required|
is_unique[users.email]');

		if ($this->form_validation->run() == FALSE)
		{
			$this->load->view('myform');
```

```
                }
                else
                {
                        $this->load->view('formsuccess');
                }
        }

        public function username_check($str)
        {
                if ($str == 'test')
                {
                        $this->form_validation->set_message('username_check', 'The %s
field can not be the word "test"');
                        return FALSE;
                }
                else
                {
                        return TRUE;
                }
        }

}
?>
```

Reload formulir dan mengirimkannya dengan kata "test" sebagai username. Anda dapat melihat bahwa data lapangan berupa disahkan untuk fungsi callback Anda untuk Anda untuk memproses.

Untuk memanggil callback hanya menempatkan nama fungsi dalam aturan, dengan "callback_" sebagai aturan awalan. Jika Anda harus menerima parameter tambahan dalam fungsi callback Anda, tambahkan biasanya setelah nama fungsi antara tanda kurung, seperti dalam: "callback_foo [bar]", maka akan diteruskan sebagai argumen kedua fungsi callback Anda.

Catatan: Anda juga dapat mengolah data formulir yang dikirimkan ke callback Anda dan mengembalikannya. Jika callback Anda kembali apa pun selain boolean BENAR /

SALAH diasumsikan bahwa data data formulir yang baru diproses Anda.

Mengatur Pesan Kesalahan

Semua pesan kesalahan asli berada di file bahasa berikut: l**anguage / english / form_validation_lang.php**

Untuk mengatur pesan khusus Anda sendiri Anda dapat mengedit file tersebut, atau gunakan fungsi berikut:

- `$this->form_validation->set_message('rule', 'Error Message');`

Dimana aturan sesuai dengan nama aturan tertentu, dan Error Message adalah teks yang ingin ditampilkan.

Jika Anda termasuk% s dalam string kesalahan Anda, maka akan diganti dengan "nama orang" yang Anda gunakan untuk bidang Anda ketika Anda menetapkan aturan Anda.

Dalam "callback" contoh di atas, pesan kesalahan ditetapkan dengan melewatkan nama fungsi:

- `$this->form_validation->set_message('username_check')`

Anda juga dapat menimpa setiap pesan kesalahan yang ditemukan dalam file bahasa. Sebagai contoh, untuk mengubah pesan untuk "diperlukan" aturan Anda akan melakukan hal ini:

- `$this->form_validation->set_message('required', 'Your custom message here');`

Menerjemahkan Nama Lapangan

Jika Anda ingin menyimpan "nama orang" kedalam fungsi set_rules () dalam file bahasa, dan dengan itu akan membuat nama dapat diterjemahkan, begini caranya:

Pertama, awalan Anda "orang" nama dengan lang :, seperti dalam contoh ini:

- `$this->form_validation->set_rules('first_name', 'lang:first_name', 'required');`

Kemudian, simpan nama di salah satu kedalam array file bahasa anda (tanpa awalan):

- `$lang['first_name'] = 'First Name';`

Catatan: Jika Anda menyimpan array item dalam file bahasa yang tidak dimuat secara otomatis oleh CI, Anda harus ingat untuk memuatnya di controller Anda menggunakan:

- `$this->lang->load('file_name');`

Lihat Halaman Language Class untuk info lebih lanjut mengenai file bahasa.

Mengubah Error Delimiters

Secara default, kelas Validasi Form menambahkan tag ayat (<p>) di masing-masing pesan kesalahan ditampilkan. Anda dapat mengubah pembatas ini secara global atau secara individu.

1. Mengubah pembatas global

   Untuk secara global mengubah pembatas kesalahan, dalam fungsi controller, setelah memuat kelas Validasi Form, tambahkan ini:

   ```
   $this->form_validation->set_error_delimiters('<div class="error">', '</div>');
   ```

   Dalam contoh ini, kita sudah beralih menggunakan tag div.
2. Mengubah pembatas Individual

   Masing-masing dari dua fungsi menghasilkan kesalahan yang ditunjukkan dalam tutorial ini dapat diberikan pembatas mereka sendiri sebagai berikut:

```
<?php echo form_error('field name', '<div class="error">', '</div>'); ?
>
<?php echo validation_errors('<div class="error">', '</div>'); ?>
```

Menampilkan Kesalahan Individual

Jika Anda lebih memilih untuk menunjukkan pesan kesalahan di sebelah setiap bentuk lapangan, bukan sebagai daftar, Anda dapat menggunakan form_error () fungsi.

Cobalah! Mengubah bentuk sehingga terlihat seperti ini:

```
<h5>Username</h5>
<?php echo form_error('username'); ?>
<input type="text" name="username" value="<?php echo set_value('username'); ?>"
size="50" />

<h5>Password</h5>
<?php echo form_error('password'); ?>
<input type="text" name="password" value="<?php echo set_value('password'); ?>"
size="50" />

<h5>Password Confirm</h5>
<?php echo form_error('passconf'); ?>
<input type="text" name="passconf" value="<?php echo set_value('passconf'); ?>"
size="50" />

<h5>Email Address</h5>
<?php echo form_error('email'); ?>
<input type="text" name="email" value="<?php echo set_value('email'); ?>" size="50" />
```

Jika tidak ada kesalahan, tidak ada yang akan ditampilkan. Jika ada kesalahan, pesan akan muncul.

Catatan Penting: Jika Anda menggunakan sebuah array sebagai nama bidang formulir, Anda harus menyediakan sebagai sebuah array ke fungsi. Contoh:

```
<?php echo form_error('options[size]'); ?>
```

```
<input type="text" name="options[size]" value="<?php echo
set_value("options[size]"); ?>" size="50" />
```

Untuk info lebih lanjut silakan lihat Array Menggunakan sebagai bagian Nama Lapangan bawah.

Penyimpan Sets dari Validasi Rules ke Config File

Sebuah fitur bagus dari kelas Validasi Form adalah bahwa hal itu memungkinkan Anda untuk menyimpan semua aturan validasi kedalam seluruh aplikasi dalam file konfigurasi. Anda dapat mengatur aturan-aturan ini menjadi "kelompok". Kelompok-kelompok ini dapat baik dimuat secara otomatis ketika pencocokan controller / fungsi dipanggil, atau Anda dapat secara manual memanggil setiap set sesuai kebutuhan.

Bagaimana Cara Menyimpan Aturan

Untuk menyimpan aturan validasi Anda, cukup membuat file bernama **form_validation.php** dalam application / config / folder Anda. Dalam file yang Anda akan menempatkan sebuah array bernama $ config dengan aturan Anda. Seperti yang ditunjukkan sebelumnya, array validasi akan memiliki prototipe ini:

```
$config = array(
      array(
            'field'            => 'username',
            'label'            => 'Username',
            'rules'            => 'required'
      ),
      array(
            'field'            => 'password',
            'label'            => 'Password',
            'rules'            => 'required'
      ),
      array(
            'field'            => 'passconf',
            'label'            => 'Password Confirmation',
            'rules'            => 'required'
      ),
```

```
      array(
            'field'            => 'email',
            'label'            => 'Email',
            'rules'            => 'required'
      )
      );
```

Aturan file validasi Anda akan dimuat secara otomatis dan digunakan ketika Anda memanggil **run () function.**

Harap dicatat bahwa Anda harus memberi nama array $ config.

Menciptakan Set Rules

Dalam rangka untuk mengatur aturan Anda ke "set" mengharuskan Anda menempatkan mereka ke dalam "sub array". Perhatikan contoh berikut, menampilkan dua set aturan. Kami telah sewenang-wenang disebut dua aturan "pendaftaran" dan "email". Anda dapat nama Anda apa aturan yang Anda inginkan:

```
$config = array(
      'signup' => array(
                  array(
                        'field' => 'username',
                        'label' => 'Username',
                        'rules' => 'required'
                  ),
                  array(
                        'field' => 'password',
                        'label' => 'Password',
                        'rules' => 'required'
                  ),
                  array(
                        'field' => 'passconf',
                        'label' => 'PasswordConfirmation',
                        'rules' => 'required'
                  ),
                  array(
                        'field' => 'email',
                        'label' => 'Email',
                        'rules' => 'required'
                  )
```

```
                ),
        'email' => array(
                array(
                        'field' => 'emailaddress',
                        'label' => 'EmailAddress',
                        'rules' => 'required|valid_email'
                        ),
                        array(
                        'field' => 'name',
                        'label' => 'Name',
                        'rules' => 'required|alpha'
                        ),
                array(
                        'field' => 'title',
                        'label' => 'Title',
                        'rules' => 'required'
                        ),
                array(
                        'field' => 'message',
                        'label' => 'MessageBody',
                        'rules' => 'required'
                )
            )
        );
```

Memanggil Peraturan Grup Tertentu

Dalam rangka untuk memanggil kelompok tertentu Anda akan melewati namanya menjadi run () function. Misalnya, untuk memanggil aturan pendaftaran Anda akan melakukan hal ini:

```
if ($this->form_validation->run('signup') == FALSE)
{
        $this->load->view('myform');
}
else
```

```
{
	$this->load->view('formsuccess');
}
```

Mengaitkan Fungsi Controller dengan Peraturan Grup

Alternatif (dan lebih otomatis) metode memanggil kelompok aturan adalah nama sesuai dengan kelas controller / fungsi Anda berniat untuk menggunakannya dengan. Sebagai contoh, katakanlah Anda memiliki controller bernama Anggota dan function bernama pendaftaran. Inilah yang kelas Anda akan terlihat seperti:

```
<?php

class Member extends CI_Controller {

 function signup()
 {
	$this->load->library('form_validation');

 if ($this->form_validation->run() == FALSE)
 {
	$this->load->view('myform');
 }
 else
 {
	$this->load->view('formsuccess');
	}
 }
}
?>
```

Dalam file konfigurasi validasi , Anda akan membuat nama aturan member/pendaftar grup kamu :

```
$config = array(
	'member/signup' => array(
		array(
				'field' => 'username',
```

```
                        'label' => 'Username',
                        'rules' => 'required'
                ),
                array(
                        'field' => 'password',
                        'label' => 'Password',
                        'rules' => 'required'
                ),
                array(
                        'field' => 'passconf',
                        'label' => 'PasswordConfirmation',
                        'rules' => 'required'
                ),
                array(
                        'field' => 'email',
                        'label' => 'Email',
                        'rules' => 'required'
                    )
                )
        );
```

Ketika sekelompok peraturan bernama identik dengan kelas controller / fungsi itu akan digunakan secara otomatis ketika run () fungsi dipanggil dari kelas / fungsi.

Function Reference

- **$this->form_validation->set_rules();**

Memungkinkan Anda untuk menetapkan aturan validasi, seperti yang dijelaskan dalam bagian tutorial di atas:

1. Mengatur Aturan Validasi
2. Menyimpan grup Validasi Rules ke Config File

- **$this->form_validation->run();**

Menjalankan rutinitas validasi. Pengembalian nilai boolean TRUE pada keberhasilan dan FALSE pada kegagalan. Anda dapat melewati nama grup validasi melalui fungsi, seperti yang dijelaskan dalam: **Saving grup Validasi Rules ke Config File**.

- **$ this-> form_validation-> set_message ();**

Memungkinkan Anda untuk mengatur pesan kesalahan kustom. Lihat Mengatur Pesan Kesalahan di atas.

Helper Reference

Fungsi helper berikut tersedia untuk digunakan dalam file tampilan yang berisi form Anda. Perhatikan bahwa ini adalah fungsi prosedural, sehingga mereka tidak mengharuskan Anda untuk tambahkan mereka dengan $ this-> form_validation.

- **form_error ()**

Menunjukkan pesan kesalahan individu yang terkait dengan nama field yang disediakan untuk fungsi. contoh:

<? php echo form_error ('username'); ?>

Pembatas kesalahan dapat opsional ditentukan. Lihat Mengubah bagian pembatas Kesalahan di atas.

- **validation_errors()**

Menunjukkan semua pesan error sebagai string: Contoh:

<? validation_errors php echo (); ?>

Pembatas kesalahan dapat opsional ditentukan. Lihat Mengubah bagian pembatas Kesalahan di atas.

- **set_value()**

Memungkinkan Anda untuk menetapkan nilai bentuk input atau textarea. Anda harus menyediakan nama field melalui parameter pertama fungsi. Yang kedua (opsional) parameter memungkinkan Anda untuk mengatur nilai default untuk formulir. Contoh:

```
<input type="text" name="quantity" value="<?php echo set_value('quantity',
'0'); ?>" size="50" />
```

Bentuk di atas akan menampilkan "0" ketika dimuat untuk pertama kalinya.

- **set_select()**

Jika Anda menggunakan <pilih> menu, fungsi ini memungkinkan Anda untuk menampilkan item menu yang dipilih. Parameter pertama harus berisi nama pilih menu, parameter kedua harus mengandung nilai setiap item, dan yang ketiga (opsional) parameter memungkinkan Anda mengatur item sebagai default (penggunaan boolean BENAR / SALAH).

```
<select name="myselect">
```

```
<option value="one"<?php echo set_select('myselect', 'one', TRUE); ?>
>One</option>
<option value="two"<?php echo set_select('myselect', 'two'); ?> >Two</option>
<option value="three"<?php echo set_select('myselect', 'three'); ?>
>Three</option>
</select>
```

- **set_checkbox()**

Memungkinkan Anda untuk menampilkan kotak centang dalam keadaan itu diserahkan. Parameter pertama harus berisi nama kotak centang, parameter kedua harus mengandung nilai, dan yang ketiga (opsional) parameter memungkinkan Anda mengatur item sebagai default (penggunaan boolean BENAR / SALAH). contoh:

```
<input type="checkbox" name="mycheck[]" value="1" <?php echo
set_checkbox('mycheck[]', '1'); ?> />
<input type="checkbox" name="mycheck[]" value="2" <?php echo
set_checkbox('mycheck[]', '2'); ?> />
```

- **set_radio()**

Memungkinkan Anda untuk menampilkan tombol radio dalam keadaan mereka diserahkan. Fungsi ini identik dengan set_checkbox () fungsi di atas.

```
<input type="radio" name="myradio" value="1" <?php echo set_radio('myradio',
'1', TRUE); ?> />
<input type="radio" name="myradio" value="2" <?php echo set_radio('myradio',
'2'); ?> />
```

Migrations Class

Migrations adalah cara mudah bagi Anda untuk mengubah database Anda dengan cara yang terstruktur dan terorganisir. Anda bisa mengedit pecahan SQL secara manual tetapi Anda kemudian harus bertanggung jawab memberitahu developer lain yang mereka butuhkan dengan memulai dan menjalankannya. Anda juga harus melacak perubahan mana harus dijalankan terhadap engine buatan anda saat Anda mengerahkannya.

Tabel database trek migrasi yang migrasi telah dijalankan sehingga yang harus Anda lakukan adalah memperbarui file aplikasi Anda dan memanggil **$ this-> migrate-> current ()** untuk bekerja keluar yang migrasi harus dijalankan. Versi saat ini ditemukan

dalam **config / migration.php.**

Buat Migrasi

Ini akan menjadi migrasi pertama untuk situs baru yang memiliki blog. Semua migrasi masuk dalam folder **application / migrations** / dan memiliki nama seperti: 001_add_blog.php.

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

class Migration_Add_blog extends CI_Migration {
        public function up()
        {
                $this->dbforge->add_field(array(
                        'blog_id' => array(
                                'type' => 'INT',
                                'constraint' => 5,
                                'unsigned' => TRUE,
                                'auto_increment' => TRUE
                          ),
                        'blog_title' => array(
                                'type' => 'VARCHAR',
                                'constraint' => '100',
                          ),
                        'blog_description' => array(
                                'type' => 'TEXT',
                                'null' => TRUE,
                          ),
                    ));

                $this->dbforge->create_table('blog');
        }

        public function down()
        {
                $this->dbforge->drop_table('blog');
        }
```

Kemudian dalam **application/config/migration.php** set $config['migration_version'] =

1;.

Contoh penggunaan

Dalam contoh ini beberapa kode sederhana ditempatkan dalam **application/controllers/ migrate.php** untuk memperbarui skema.

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

if ( ! $this->migration->current())
{
      show_error($this->migration->error_string());
}
```

Function Referensi

- **$ this-> migration-> current ()**

Migrasi saat ini apa pun yang ditetapkan sebesar $ config ['migration_version'] dalam **application / config / migration.php.**

- **$ this-> migration-> latest ()**

Ini bekerja dengan cara yang sama seperti saat ini () tapi bukannya mencari $ config ['migration_version'] kelas Migrasi akan menggunakan migrasi sangat terbaru yang ditemukan di filesystem.

- **$ this-> migration-> version ()**

Versi ini dapat digunakan untuk memutar kembali perubahan atau langkah ke depan pemrograman dengan versi tertentu. Ia bekerja seperti sekarang tetapi akan mengabaikan **$ config ['migration_version'].**

```
•   $this->load->library('migration');

•   $this->migration->version(5);
```

Pagination Class

Pagination class CodeIgniter adalah sangat mudah digunakan, dan itu adalah 100% disesuaikan, baik secara dinamis atau melalui preferensi disimpan.

Jika Anda tidak akrab dengan istilah "pagination", mengacu pada link yang memungkinkan Anda untuk menavigasi dari halaman ke halaman, seperti ini:

```
« First  < 1 2 3 4 5 >  Last »
```

Berikut ini adalah contoh sederhana yang menunjukkan bagaimana membuat pagination di salah satu fungsi controller Anda:

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

$config['base_url'] = 'http://example.com/index.php/test/page/';
$config['total_rows'] = 200;
$config['per_page'] = 20;

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

echo $this->pagination->create_links();
```

Catatan:

The $ config array berisi variabel konfigurasi Anda. Hal ini diteruskan ke **$this-> pagination-> initialize function** seperti yang ditunjukkan di atas. Walaupun ada beberapa puluh item yang dapat mengkonfigurasi, minimal Anda memerlukan tiga yang ditampilkan. Berikut ini adalah deskripsi dari apa yang item mewakili:

- **base_url** ini adalah URL lengkap dengan kelas controller / fungsi yang berisi pagination Anda. Dalam contoh di atas, itu menunjuk ke controller yang disebut "Test" dan fungsi yang disebut "halaman". Perlu diingat bahwa Anda dapat mengubah rute URI Anda jika Anda membutuhkan struktur yang berbeda.
- **total_row** Jumlah ini merupakan total baris di dalam hasil set Anda buat pagination untuk. Biasanya angka ini akan menjadi total baris yang query database Anda kembali.

- **per_page** Jumlah item Anda berniat untuk menunjukkan per halaman. Dalam contoh di atas, Anda akan menampilkan 20 di setiap halaman.

create_links () mengembalikan fungsi string kosong ketika tidak ada pagination untuk ditampilkan.

Setting Preferensi dalam file config

Jika Anda memilih untuk tidak menetapkan preferensi menggunakan metode di atas, Anda malah bisa menempatkan mereka ke dalam config file. Cukup membuat file baru yang disebut pagination.php, tambahkan array $ config dalam file tersebut. Kemudian simpan file dalam: **config / pagination.php** dan akan digunakan secara otomatis. Anda tidak perlu menggunakan **$ this-> pagination-> initialize** fungsi jika Anda menyimpan preferensi Anda dalam file konfigurasi.

Menyesuaikan Pagination

Berikut ini adalah daftar semua preferensi Anda dapat lulus dengan fungsi inisialisasi untuk menyesuaikan layar.

1. $ config ['uri_segment'] = 3;

   Fungsi pagination otomatis menentukan segmen URI Anda berisi nomor halaman. Jika Anda membutuhkan sesuatu yang berbeda Anda dapat menentukan hal itu.
2. $ config ['num_links'] = 2;

   Jumlah "digit" link Anda ingin sebelum dan setelah nomor halaman yang dipilih. Sebagai contoh, nomor 2 akan menempatkan dua digit di kedua sisi, seperti dalam contoh link di bagian paling atas dari halaman ini.
3. $ config ['use_page_numbers'] = TRUE;

   Secara default, segmen URI akan menggunakan indeks awal untuk item yang Anda paginating. Jika Anda lebih memilih untuk menunjukkan jumlah halaman yang sebenarnya, mengatur ini ke TRUE.
4. $ config ['page_query_string'] = TRUE;

   Secara default, libraries pagination menganggap anda menggunakan URI Segmen, dan membangun sesuatu link anda seperti :

http://example.com/index.php/test/page/20

Jika Anda memiliki **$ config ['enable_query_strings']** diatur ke TRUE tautan Anda secara otomatis akan ditulis kembali menggunakan Query String. Opsi ini juga dapat secara eksplisit diatur. Menggunakan **$ config ['page_query_string']** disetel ke TRUE, link pagination akan menjadi.

http://example.com/index.php?c=test&m=page&per_page=20

Perhatikan bahwa "**per_page**" adalah string bawaan yang disahkan, namun dapat dikonfigurasi dengan menggunakan $ config ['query_string_segment'] = 'your_string'

Menambahkan Melampirkan Markup

Jika Anda ingin mengelilingi seluruh pagination dengan beberapa markup Anda dapat melakukannya dengan dua Preferensi ini:

$ config ['full_tag_open'] = '<p>';

Tag pembuka ditempatkan di sisi kiri seluruh hasil.

$ config ['full_tag_close'] = '</ p>';

Tag penutup diletakkan di sisi kanan seluruh hasil.

Menyesuaikan First Link

$ config ['first_link'] = 'FIRST';

Teks yang akan suka ditampilkan dalam "first" link di sebelah kiri. Jika Anda tidak ingin link ini diberikan, Anda dapat mengatur nilai ke FALSE.

$ config ['first_tag_open'] = ' ';

Tag pembuka untuk "first" link.

$ config ['first_tag_close'] = '</ div>';

The tag penutup untuk "first" link.

Menyesuaikan Link Last

$ config ['last_link'] = 'last';

Teks yang akan suka ditampilkan dalam "terakhir" link di sebelah kanan. Jika Anda tidak ingin link ini diberikan, Anda dapat mengatur nilai ke FALSE.

$ config ['last_tag_open'] = ' ';

Tag pembuka untuk "last" link.

$ config ['last_tag_close'] = '</ div>';

Tag penutup untuk "last" link.

Menyesuaikan "Next" Link

$ config ['next_link'] = '& gt;';

Teks yang akan suka ditampilkan dalam "berikutnya" link halaman. Jika Anda tidak ingin link ini diberikan, Anda dapat mengatur nilai ke FALSE.

$ config ['next_tag_open'] = ' ';

Tag pembuka untuk "link" link.

$ config ['next_tag_close'] = '</ div>';

Tag penutup untuk "link" link.

Menyesuaikan "Previous" Link

$ config ['prev_link'] = '& lt;';

Teks yang akan suka ditampilkan dalam "sebelumnya" link halaman. Jika Anda tidak ingin link ini diberikan, Anda dapat mengatur nilai ke FALSE.

$ config ['prev_tag_open'] = ' ';

Tag pembuka untuk "Previous" link.

$ config ['prev_tag_close'] = '</ div>';

The tag penutup untuk "Previous" link.

Menyesuaikan "Current Page" Link

$ config ['cur_tag_open'] = '<b>';

Tag pembuka untuk "current" link.

$ config ['cur_tag_close'] = '</ b>';

The tag penutup untuk "current" link.

Menyesuaikan "Digit" Link

$ config ['num_tag_open'] = ' ';

Tag pembuka untuk "digit" link.

$ config ['num_tag_close'] = '</ div>';

The tag penutup untuk "digit" link.

Menyembunyikan Pages

Jika Anda ingin tidak mencantumkan halaman tertentu (misalnya, Anda hanya ingin "sebelumnya" link "di samping" dan), Anda dapat menekan render mereka dengan menambahkan:

$ config ['display_pages'] = FALSE;

Sessions Class

Class Session memungkinkan Anda mempertahankan "keadaan" user dan melacak aktivitas mereka sementara mereka menjelajahi situs Anda. Class Session menyimpan informasi Sessions untuk setiap bagian user serial (dan opsional dienkripsi) data didalam cookie. Hal ini juga dapat menyimpan data session dalam tabel database untuk keamanan tambahan, karena hal ini memungkinkan ID session di cookie pengguna untuk dicocokkan terhadap ID sesi yang tersimpan. Secara default hanya cookie yang disimpan. Jika Anda memilih untuk menggunakan pilihan database Anda harus membuat tabel sesi seperti yang ditunjukkan di bawah ini.

Catatan: Class Session tidak menggunakan sesi PHP asli. Ini menghasilkan data sesi sendiri, menawarkan fleksibilitas lebih untuk pengembang.

Catatan: Bahkan jika Anda tidak menggunakan sesi dienkripsi, Anda harus mengatur kunci enkripsi dalam file konfigurasi yang digunakan untuk membantu dalam mencegah manipulasi data sesi.

Catatan: Class Session bergantung pada kelas Encryption, sehingga Anda harus menginstal ekstensi Mcrypt

Sessions biasanya akan berjalan secara global dengan setiap muat laman, sehingga class session harus ditentukan diinisialisasi dalam constructor controller, atau bisa auto-load

oleh sistem. Untuk sebagian besar class session akan berjalan tanpa pengawasan di background, sehingga hanya menginisialisasi class akan menyebabkan untuk membaca, membuat, dan sesi pembaruan.

Untuk menginisialisasi class Session manual dalam konstruktor controller, gunakan fungsi

$ this-> load-> Library:

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

Setelah dimuat, objek Library Sessions akan tersedia dengan menggunakan: $ this-> session

Bagaimana cara kerja Sessions?

Ketika halaman dibuka, class session akan memeriksa untuk melihat apakah data sesi valid ada dalam sesi cookie pengguna. Jika sesi Data tidak ada (atau jika telah kedaluwarsa) sesi baru akan dibuat dan disimpan dalam cookie. Jika sesi memang ada, informasinya akan diperbarui dan cookie akan diperbarui. Dengan setiap update, session_id akan dibuat ulang.

Sangat penting bagi Anda untuk memahami bahwa setelah diinisialisasi, class Session berjalan secara otomatis. Tidak ada yang perlu Anda lakukan dengan menyebabkan perilaku di atas terjadi. Anda bisa, karena Anda akan lihat di bawah, bekerja dengan data session atau bahkan menambahkan data Anda sendiri untuk session pengguna, tapi proses membaca, menulis, dan memperbarui sesi secara otomatis.

Intinya Session adalah sebuah varibel default sementara yang diletakkan di server. Di mana PHP bisa mengambil nilai yang tersimpan di server walaupun kita membuka halaman baru. Biasanya session akan hilang jika kamu menutup browser.
Sebelum kamu menggunakan fungsi – fungsi tentang session di CI kamu harus memanggil library session terlebih dahulu dengan cara : **$this->load->library('session');**

Selanjutnya cara kita membuat atau memasukan nilai ke variabel sementara atau session :

$this->session->set_userdata('some_name', 'some_value');

Atau bisa juga dengan menggunakan array :

```
$data = array(
              'username'   => 'budi',
              'email'            => 'budi@airputih.or.id',
              'logged_in'  => TRUE
       );
$this->session->set_userdata($data);
```

Setelah itu jika anda ingin mengambil atau memakai nilai dari session maka cara memanggilnya seperti ini :

```
$session_id=$this->session->userdata('session_id');
```

Dan terakhir perintah untuk menghapus atau mengosongkan session :

$this->session->unset_userdata('some_name');

Database Class
Database Quick Start: Contoh Kode

man berikut berisi contoh kode yang menunjukkan bagaimana kelas database yang digunakan. Untuk rincian lengkap silahkan baca halaman individual menjelaskan fungsi masing-masing.

Standard Query Dengan Beberapa Hasil (Object Version)

```
$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();
```

Hasil result () mengembalikan fungsi array obyek. Contoh: $ row-> title

Standard Query Dengan Beberapa Hasil (Array Version)

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

foreach ($query->result_array() as $row)
{
       echo $row['title'];
       echo $row['name'];
       echo $row['email'];
}
```

Sebuah **result_array()** diatas mengembalikan fungsi array indeks array standar. Contoh: **$row ['title']**

Pengujian Hasil

Jika Anda menjalankan query yang mungkin tidak menghasilkan hasil, Anda dianjurkan untuk tes untuk hasil pertama menggunakan num_rows () function:

```
$query = $this->db->query("YOUR QUERY");

if ($query->num_rows() > 0)
{
       foreach ($query->result() as $row)
       {
              echo $row->title;
              echo $row->name;
              echo $row->body;
       }
}
```

Standard Query Dengan Hasil Tunggal

```
$query = $this->db->query('SELECT name FROM my_table LIMIT 1');
$row = $query->row();
```

```
echo $row->name;
```

row() diatas mengembalikan fungsi objek. Contoh: $ row-> name

Standard Query Dengan Hasil Tunggal (versi Array)

```
$query = $this->db->query('SELECT name FROM my_table LIMIT 1');

$row = $query->row_array();
echo $row['name'];
```

row_array() diatas mengembalikan fungsi array. Contoh: $ row ['name']

Standard Insert

```
$sql = "INSERT INTO mytable (title, name)
       VALUES (".$this->db->escape($title).", ".$this->db->escape($name).")";

$this->db->query($sql);

echo $this->db->affected_rows();
```

Active Record Query

Pola Active Record memberi Anda cara sederhana mengambil data:

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

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

Di atas fungsi get () mengambil semua hasil dari tabel yang disediakan. Class Active Record memuat penuh fungsi pelengkap untuk bekerja dengan data.

Active Record Insert

```
$data = array(
              'title' => $title,
              'name' => $name,
              'date' => $date
```

```
      );

$this->db->insert('mytable', $data);

// Produces: INSERT INTO mytable (title, name, date) VALUES ('{$title}',
'{$name}', '{$date}')
```

Running Queries
$this->db->query();
Masukkan sebuah Query, gunakan fungsi petunjuk dibawah ini.

```
$this->db->query('YOUR QUERY HERE');
```

Query () mengembalikan fungsi objek hasil database ketika "read" jenis query dijalankan, dimana dapat Anda gunakan untuk menunjukkan hasil Anda. Ketika "write" jenis query dijalankan itu hanya mengembalikan TRUE atau FALSE, tergantung pada keberhasilan atau kegagalan. Ketika mengambil data Anda biasanya akan menetapkan query dengan variabel Anda sendiri, seperti ini:

```
$query = $this->db->query('YOUR QUERY HERE');
```

$this->db->simple_query();
ini adalah versi sederhana dari fungsi **$ this-> db-> query ()**. Ini hanya mengembalikan TRUE / FALSE pada keberhasilan atau kegagalan. Ini TIDAK mengembalikan hasil database set, juga tidak mengatur waktu query, atau mengumpulkan data terikat, atau menyimpan permintaan Anda untuk melakukan debug. Ini memudahkan Anda mengirimkan permintaan. Sebagian besar pengguna jarang akan menggunakan fungsi ini.

Menjalankan dengan Database Secara Manual

Jika Anda telah mengkonfigurasi database awal dan ingin menambahkan nama tabel untuk digunakan dalam query SQL asli misalnya, maka Anda dapat menggunakan berikut:

```
$this->db->dbprefix('tablename');
// outputs prefix_tablename
```

Jika untuk alasan apapun Anda ingin mengubah awalan programatik tanpa perlu membuat koneksi baru, Anda dapat menggunakan metode ini:

```
$this->db->set_dbprefix('newprefix');

$this->db->dbprefix('tablename');
// outputs newprefix_tablename
```

Melindungi Pengidentifikasi

Dalam banyak database disarankan untuk melindungi tabel dan nama field - misalnya dengan tanda kutip mundur di MySQL. Aktif permintaan Rekaman secara otomatis dilindungi, namun jika Anda perlu untuk melindungi sebuah identifier dapat Anda gunakan secara manual:

```
$this->db->protect_identifiers('table_name');
```

Meskipun Active Record akan berusaha sebaik mungkin untuk benar mengutip setiap bidang dan nama tabel yang Anda masukkan, perhatikan bahwa itu tidak dirancang untuk bekerja dengan user input sembarangan. JANGAN memasukkan dengan data pengguna yang tidak digunakan.

Fungsi ini juga akan menambah prefix table ke table anda, dengan asumsi anda memiliki pandangan ditentukan dalam file database konfigurasi Anda. Untuk mengaktifkan set TRUE (Boolen) melalui parameter kedua:

```
$this->db->protect_identifiers('table_name',TRUE);
```

Escaping Queries

Escaping berguna untuk mengamankan basis data dari data-data berbahaya sebelum data dimasukan kedalam basis data, misalnya saja dari karakterkarakter seperti = “ dan lain-lain dengan memberikan tanda backslash (\) setiap ditemukan karakter yang berbahaya. Untuk melakukan escaping ada dua fungsi yang bisa digunakan, yaitu :

1. $ this-> db-> escaping () Fungsi ini menentukan tipe data yang dapat di escape hanya data string. Hal ini juga secara otomatis menambahkan tanda kutip tunggal sekitar data sehingga Anda tidak perlu:

   ```
   $sql = "INSERT INTO table (title) VALUES(".$this->db->escape($title).")";
   ```

2. $ this-> db-> escape_str () Fungsi ini memberikan escape data yang diberikan untuk itu, terlepas dari jenis. Sebagian besar waktu Anda akan menggunakan fungsi di atas daripada satu ini. Gunakan fungsi seperti ini:

```
$sql = "INSERT INTO table (title) VALUES('".$this->db->escape_str($title)."')";
```

3. $ this-> db-> escape_like_str () Metode ini harus digunakan ketika string yang akan digunakan dalam kondisi Like agar karakter Like seperti ('%', '_') dalam string perangkat Escape.

```
$search = '20% raise';
$sql = "SELECT id FROM table WHERE column LIKE '%".$this->db-
>escape_like_str($search)."%'";
```

Generating Query Result

Ada beberapa cara untuk menghasilkan hasil query:

Result ()

Fungsi ini mengembalikan hasil query sebagai array obyek, atau array kosong pada kegagalan. Biasanya Anda akan menggunakan ini dalam loop foreach, seperti ini:

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

foreach ($ query-> hasil () as $ row)
{
        echo $ baris-> title;
        echo $ baris-> nama;
        echo $ baris-> tubuh;
}
Fungsi di atas adalah alias dari result_object ().
```

Jika Anda menjalankan query yang mungkin tidak menghasilkan hasil, Anda dianjurkan untuk menguji hasil pertama:

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

if ($ query-> NUM_ROWS () 0>)
{
foreach ($ query-> hasil () as $ row)
```

```
        {
        echo $ baris-> title;
        echo $ baris-> nama;
        echo $ baris-> tubuh;
 }
}
```

Anda juga dapat melewati string hasil () yang merupakan kelas untuk instantiate untuk setiap objek hasil (catatan: kelas ini harus dimuat)

```
$ query = $ this-> db-> query ("SELECT * FROM pengguna,");

foreach ($ query-> hasil ('Pengguna') as $ row)
{
 echo $ baris-> nama; // Atribut panggilan
echo $ baris-> reverse_name (); // Atau metode didefinisikan pada 'Pengguna' class
}
```

result_array ()

Fungsi ini mengembalikan hasil query sebagai array murni, atau array kosong jika tidak ada hasil yang dihasilkan. Biasanya Anda akan menggunakan ini dalam loop foreach, seperti ini:

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

foreach ($ query-> result_array () as $ row)
{
        echo $ row ['title'];
        echo $ row ['nama'];
        echo $ row ['tubuh'];
}
```

row()

Fungsi ini mengembalikan hasil baris tunggal. Jika pertanyaan Anda memiliki lebih dari satu baris, ia mengembalikan hanya baris pertama. Hasilnya dikembalikan sebagai objek. Berikut adalah contoh penggunaan:

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

```
if ($ query-> NUM_ROWS () 0>)
{
	$ row = $ query-> row ();

	echo $ baris-> title;
	echo $ baris-> nama;
	echo $ baris-> tubuh;
}
```

Jika Anda ingin baris tertentu dikembalikan Anda bisa mengirimkan nomor baris sebagai digit pada parameter pertama:

$ row = $ query-> row (5);

Anda juga dapat menambahkan parameter String kedua, yang merupakan nama kelas untuk instantiate baris dengan:

```
$ query = $ this-> db-> query ("SELECT * FROM pengguna LIMIT 1;");

$ query-> row (0, 'Pengguna')
echo $ baris-> nama; // Atribut panggilan
echo $ baris-> reverse_name (); // Atau metode didefinisikan pada 'Pengguna' class
```

row_array ()

Identik dengan baris di atas () fungsi, kecuali ia mengembalikan array. contoh:

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

if ($ query-> NUM_ROWS () 0>)
{
	$ row = $ query-> row_array ();

	echo $ row ['title'];
	echo $ row ['nama'];
	echo $ row ['tubuh'];
}
```

Jika Anda ingin baris tertentu dikembalikan Anda bisa mengirimkan nomor baris sebagai digit pada parameter pertama:

$ row = $ query-> row_array (5);

Selain itu, Anda dapat berjalan maju / mundur / pertama / terakhir melalui hasil Anda menggunakan variasi ini:

```
$ row = $ query-> FIRST_ROW ()
$ row = $ query-> LAST_ROW ()
$ row = $ query-> next_row ()
$ row = $ query-> previous_row ()
```

Secara default mereka kembali obyek kecuali Anda menempatkan kata "array" dalam parameter:

```
$ row = $ query-> FIRST_ROW ('Array')
$ row = $ query-> LAST_ROW ('Array')
$ row = $ query-> next_row ('Array')
$ row = $ query-> previous_row ('Array')
```

Fungsi Hasil Helper

$ query-> num_rows ()

Jumlah baris yang dikembalikan oleh query. Catatan: Dalam contoh ini, $ query adalah variabel bahwa objek hasil query ditugaskan untuk:

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

echo $ query-> NUM_ROWS ();
```

$ query-> num_fields ()

Jumlah BIDANG (kolom) dikembalikan oleh query. Pastikan untuk memanggil fungsi menggunakan hasil objek query Anda:

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

```
echo $ query-> num_fields ();
```

$ query-> free_result ()

Ini membebaskan memori yang terkait dengan hasil dan menghapus ID hasil sumber daya. Biasanya PHP membebaskan memori secara otomatis pada akhir eksekusi script. Namun, jika Anda menjalankan banyak pertanyaan dalam naskah tertentu Anda mungkin ingin membebaskan hasilnya setelah setiap hasil query telah dihasilkan dalam rangka untuk mengurangi konsumsi memori. contoh:

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

foreach ($ query-> hasil () as $ row)
{
        echo $ baris-> title;
}
$ query-> free_result (); // Objek $ hasil query tidak akan lagi tersedia

$ query2 = $ this-> db-> query ("SELECT nama FROM some_table ');

$ row = $ query2-> row ();
echo $ baris-> nama;
$ query2-> free_result (); // The $ query2 hasil objek tidak akan lagi tersedia
```

BAB IV

Installasi Codeigniter

Memulai belajar framework CodeIgniter kita harus menyiapkan terlebih dahulu adalah web servernya. Anda bisa menggunakan salah satu web server yang menurut anda nyaman dengan menggunakan web server tersebut. Salah satu web server yang sering digunakan adalah Mysql, Oracle, PostgreeSQL, SQlite dan lain lain. CodeIgniter bisa dijalankan diberbagai sistem operasi mulai Windows, Linux , Macintos dan yang lainnya.

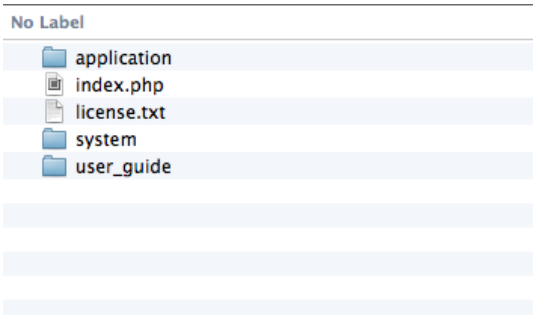
Untuk siapa CodeIgniter ?

CodeIgniter yang tepat untuk Anda jika:

- Anda ingin kerangka dengan tapak yang kecil.
- Anda membutuhkan kinerja yang luar biasa.
- Anda perlu kompatibilitas yang luas dengan standar account hosting yang menjalankan berbagai versi PHP dan konfigurasi.
- Anda ingin kerangka yang membutuhkan hampir nol konfigurasi.
- Anda ingin kerangka yang tidak mengharuskan Anda untuk menggunakan baris perintah.
- Anda ingin kerangka yang tidak mengharuskan Anda untuk mematuhi aturan pengkodean ketat.
- Anda tidak ingin dipaksa untuk belajar bahasa template (meskipun parser template opsional tersedia jika Anda inginkan satu).
- Anda menghindari kompleksitas, mendukung solusi sederhana.
- Anda perlu jelas, dokumentasi menyeluruh.

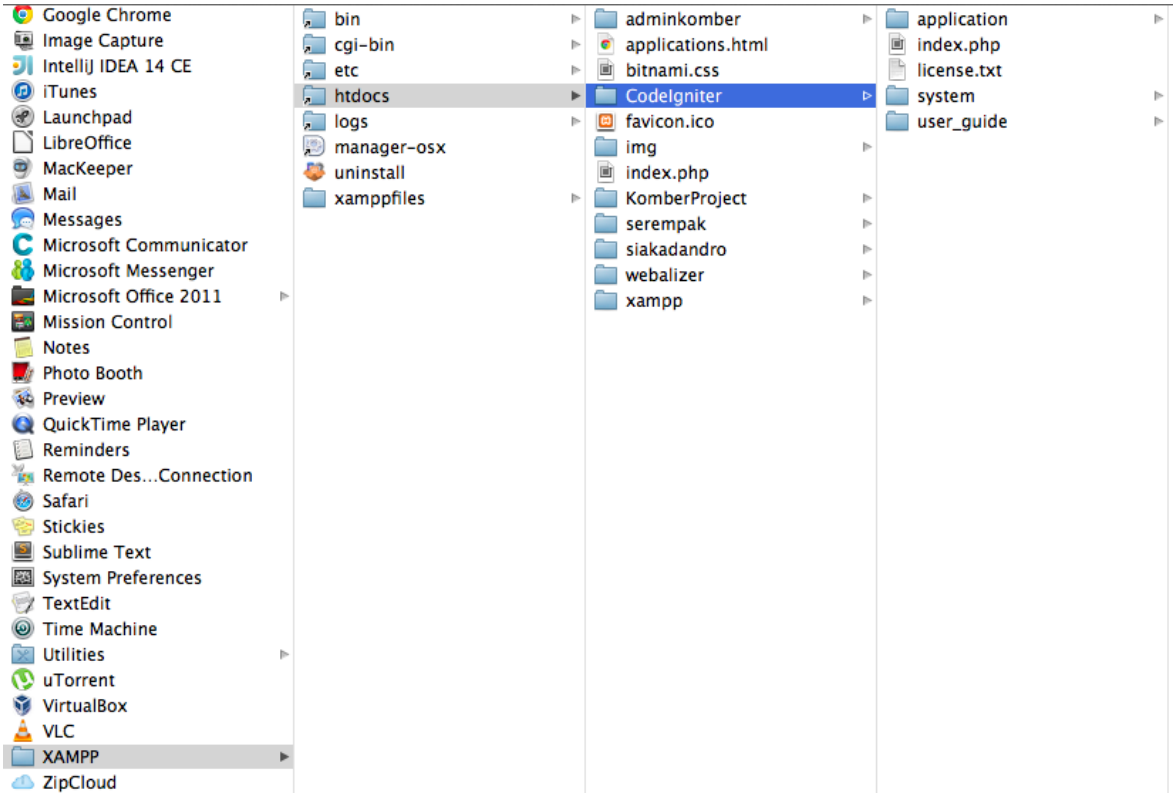
4.1. Tahap – tahap CodeIgniter.

Tahap memulai dalam menggunakan Framework CodeIgniter, anda diharuskan untuk mendownload terlebih dahulu frameworknya. Framework CodeIgniter bisa anda download di situsnya yaitu http://www.codeigniter.com/ , kemudian extrak dan didalamnya terdapat struktur folder framework sebagai berikut.



Framework CodeIgniter ini sudah berlisensi jadi anda tidak perlu kawatir , karena framework ini boleh anda gunakan untuk dikembangkan menurut keinginan anda dan yang pasti dalam mendownload framework codeIgniter ini tidak dikenakan biaya alias gratis.

Oke sekarang kita letakkan file hasil extrak codeIgniter tadi kedalam htdocs xampp. Untuk meletakkan file codeIgniter ke dalam htdocs anda harus menginstall web server di dalam komputer anda, kali ini saya menggunakan xampp, tujuannya dalam mengunakan web server adalah agar bisa diaccess oleh localhost.
Misal nama file codeIgniter adalah CodeIgniter.
XAMPP/htdocs/CodeIgniter bila anda menggunakan xampp



Jalankan browser anda dan ketik localhost/Codeigniter, maka akan muncul tampilan awal sebagai berikut :

Ini adalah tampilan awal framework codeIgniter yang sudah anda download tadi. Desain tampilan ini ada pada menu view didalam folder application yang bisa anda rubah tampilannya.

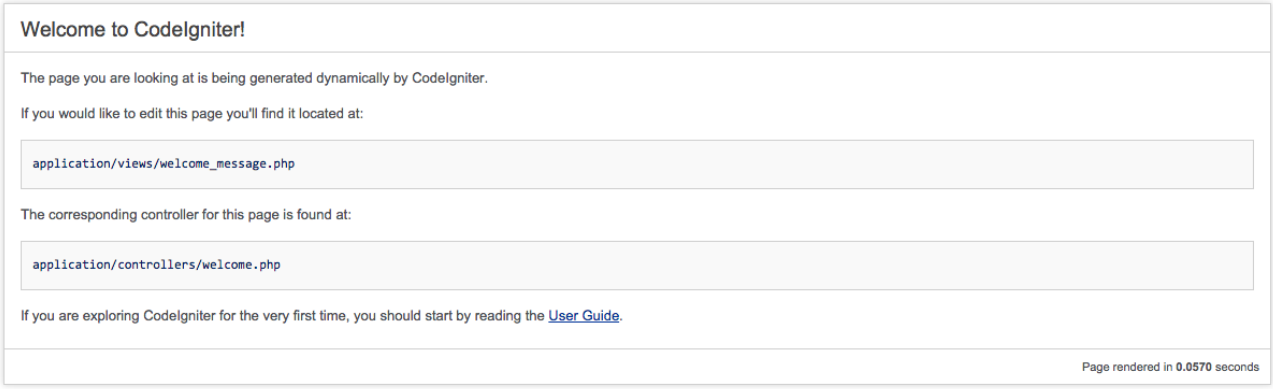
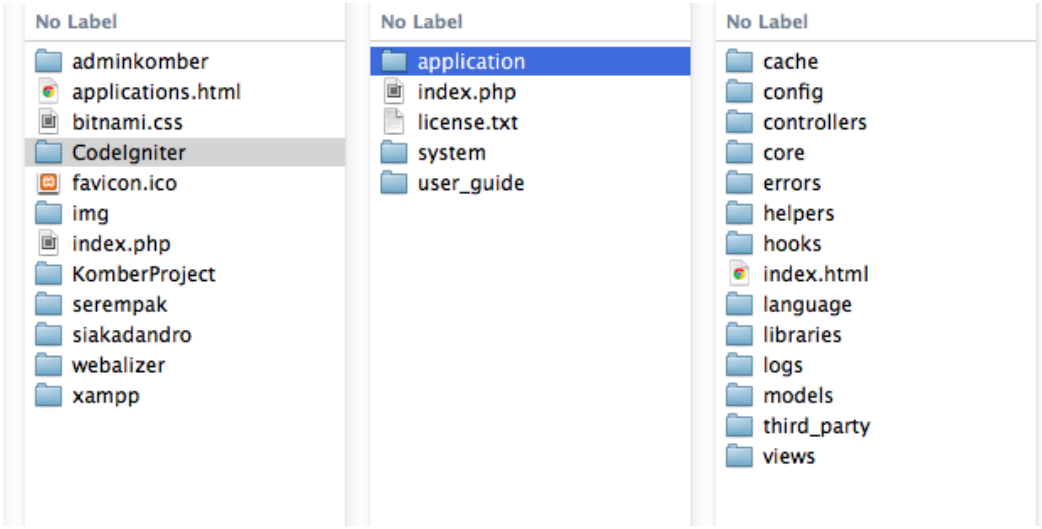


Image : Tampilan Awal CodeIgniter

4.2. Struktur CodeIgniter

Pada Framework CodeIgniter dalam pembuatan website tahap awal yang harus diperhatikan adalah pada bagian Model View dan Controller yang terdapat pada menu folder application. Secara umum penggunaan framework CodeIgniter ini nantinya anda akan lebih banyak berkutat pada folder tertentu saja diantaranya folder config, controllers, models, views karena dari sinilah awal aplikasi itu dibuat.



Didalam folder application anda akan menemukan berbagai macam folder yang memiliku tugas masing – masing akan tetapi tetap berkaitan satu sama lain. Tugas dan kegunaan folder itu antara lain adalah sebagai berikut :

Folder Application

Folder **cache** : tempat menyimpan semua cache yang dibuat di caching library

Folder **config** : tempat penyimpanan semua file konfiguarasi yang ada pada aplikasi , mulai setting database, router dan autoload aplikasi.

Folder **controller** : tempat menyimpan semua file controller yakni file perintah pemanggilan / penghubung antara model dan view.

Folder **error** : tempat menyimpan template file error aplikasi.

Folder **helpers** : tempat menyimpan file helper yang bukan berasal dari CI.

Folder **hooks** ; tempat menyimpan hook yang berfungsi sebagai alur fungsi dari core CI.

Folder **language** : tempat menyimpan bahasa bahasa yang akan anda gunakan, biasanya digunakan untuk website multi bahasa.

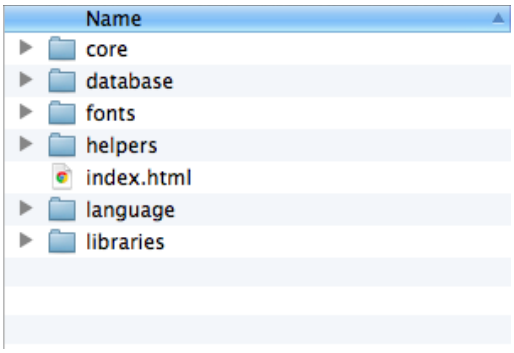
Folder **libraries** : tempat menyimpan library buatan kita sendiri.

Folder **logs** : tempat menyimpan semua log generated oleh CI.

Folder **models** : tempat menyimpan perintah model yakni perintah yang menghubungkan dengan database.

Folder **views** : tempat menyimpan file tampilan aplikasi, seperti html atau tamplate website. Tempat ini adalah tempat frontend (antarmuka) dari sebuah aplikasi.

Folder System



Folder **core** : menyimpan library inti dari codeIgniter.

Folder **database** : tempat menyimpan semua drive 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.

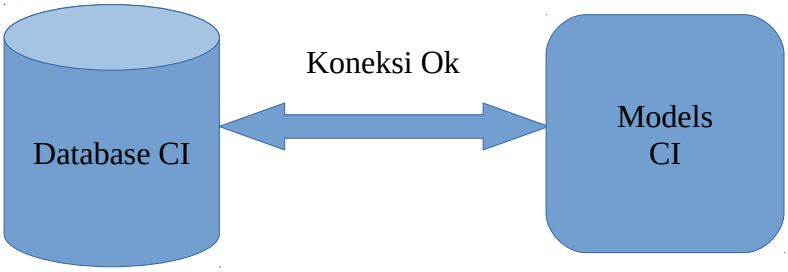
Folder **libraries** : tempat menyimpan semua library core CI.

Folder **user-guide** : folder folder petunjuk penggunaakn dalam framework codeIgniter yang berisi fitur fitur umum yang dimiliki.

CodeIgniter.

4.3. Mengenal Models

Models merupakan sebuah sistem MVC yang ada pada Framework CodeIgniter yang terletak di dalam folder application dengan nama models. Folder models yang ada pada folder **application/models** ini berguna untuk memanggil dan menghubungkan antara sistem dengan database yang sudah terkoneksi. Untuk bisa mengolah data anda harus mensetting terlebih dahulu koneksi antara aplikasi dengan database web server.



Bagian models menghubungkan database yang telah dibuat dengan bentuk Class dan fungtions dengan query sederhana creat update view dan delete.

Models membuat bentuk Class dan fungtions yang berguna untuk memanipulasi nama table database dengan nama baru yang nantinya akan dipanggil oleh controller bersadarkan Class dan functionnya.

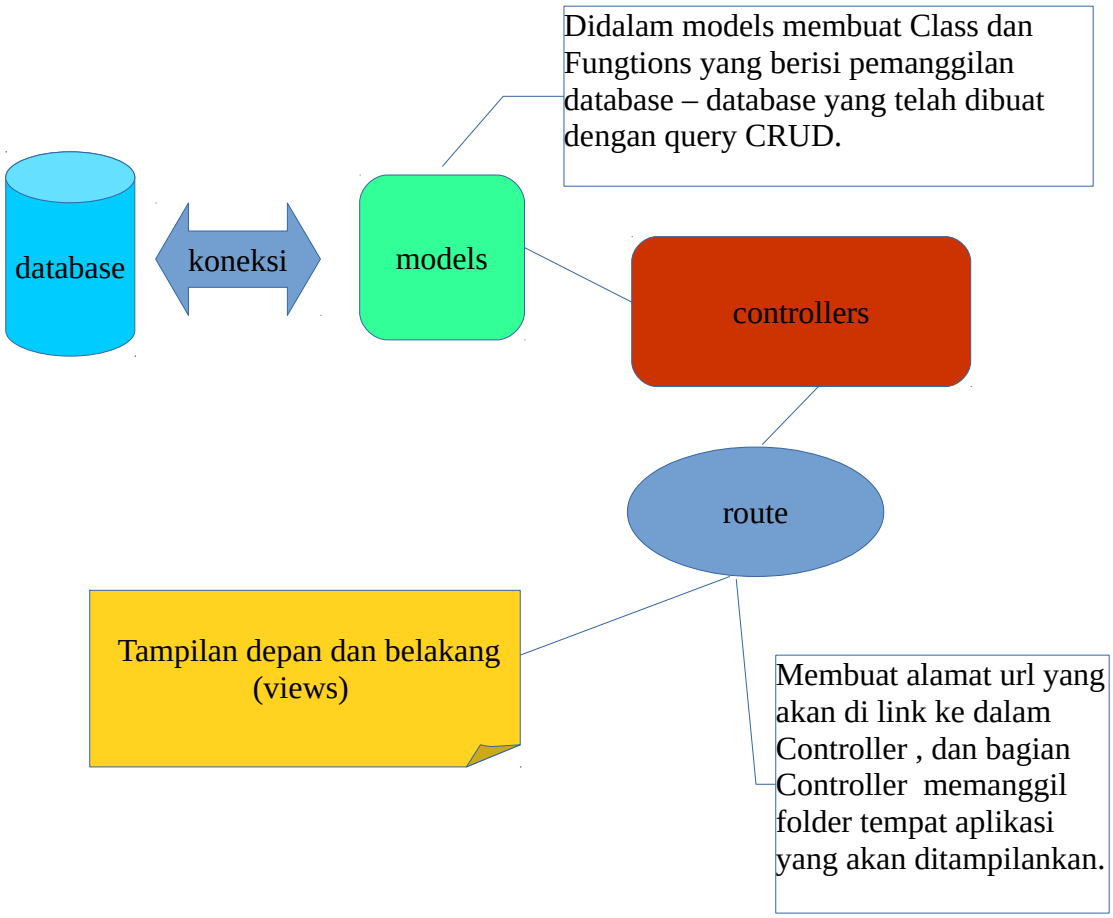
4.4. Mengenal Controller dan View

Controller disebut juga sebagai jantungnya Framework, karena disini semua alur dari database dan alur ke view dilakukan di controller. Perintah controller ini dilakukan pada folder application/controller yang memiliki nama file sama dengan nama kelasnya.

Hubungan antara Controller dan View seperti mesin dengan body tampilannya, andaikan bila digambarkan dalam sebuah motor, bagian Controller adalah enginenya sedangkan view adalah tampilan motornya. Bila dalam suatu sistem apikasi web dengan framework CI yang menggunakan model MVC bisa digambarkan bila pada bagian bahan bakarnya misalnya database sedangkan bagian model digambarkan sebagai selang penghubung database dengan engine dengan alur yang kita buat didalam

application/config/routes.php

Bila digambarkan sebagai berikut :



Sistem model MVC yang diterapkan dalam framework CodeIgniter ini yang memiliki alur proses yang digambarkan pada rangkaian tugas diatas. CodeIgniter yang terhubung dengan web server harus terkoneksi dengan tepat, anda harus melakukan setting koneksi database dibagian folder **config/database.php.**

Tugas Controller adalah menghubungkan antara models dan views dengan menampung Class dan Fungtions yang sudah dibuat model untuk dilalukan proses output/input pada bagian views untuk membuat frontend (tampilan depan) dan backend (dasboard)

dengan bantuan URL yang dibuat didalam **config/routes.php** , untuk membuat manipulasi folder tempat php dengan link url.

4.5. Konfigurasi CodeIgniter

Setting Database

Bila anda ingin mensetting konfigurasi koneksi database anda tinggal masuk pada menu **config/database.php** . Didalam file database.php anda tinggal menginputkan username , password dan nama database.

```
$active_group = 'default';
$active_record = TRUE;

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

Setting URL

Dalam setting url anda bisa melakukannya pada menu **config/routes.php** , dalam mensetting routes.php ini berhubungan dengan folder controllers

```
// Route Untuk Halaman Administrator
$route['admin/dashboard']          = 'c_admin';
$route['admin/users']                 = 'c_admin/users';
$route['admin/users/new']          = 'c_admin/add_user';
$route['admin/users/edit/(:any)']   = 'c_admin/editsuer/$1';
$route['admin/users/save']      = 'c_admin/saveuser';
```

$route['admin/dashboard'] = 'c_admin';

* maksud code diatas adalah url dengan alamat admin/dashboard akan memanggil link pada bagian contollers dengan nama Class c_admin.

$route['admin/users'] ='c_admin/users';

- maksud code diatas adalah url dengan alamat admin/users akan memanggil link pada bagian controllers dengan nama Class c_admin dan Method / functions dengan mana users.

Auto Load

Pengaturan packages, libraries, file helper, file custom, config, file language dan models yang nantinya akan di load secara default yang dapan anda ubah pada **application/config/autoload.php**

Mengatur Konfigurasi

Mengatur konfigurasi anda dapat melakukannya di dalama file **application/config/config.php** bertujuan untuk mengatur beberapa konfigurasi utama dalam konfigurasi yang kita buat.

Contoh :

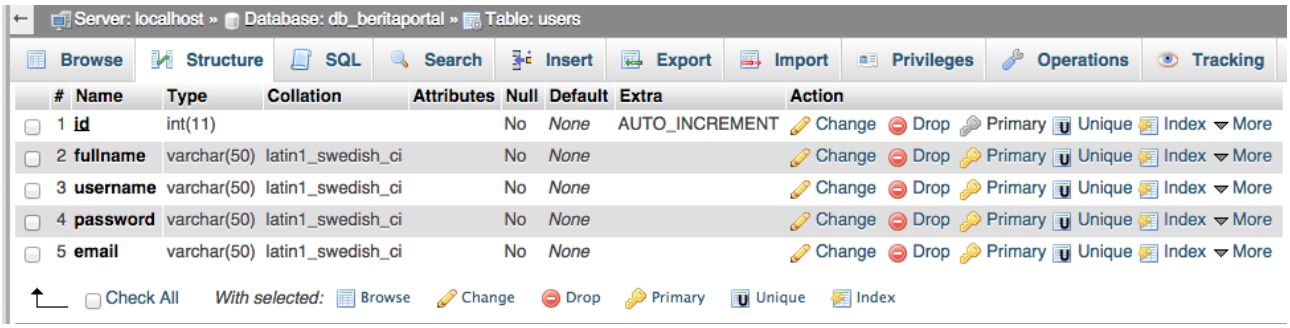
- mengatur base site url yang kita buat

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

/*
```

Membuat form login

Pertama dalam pempuatan form ini anda harus membuat databasenya terlebih dahulu, pembuatan database sederhana akan dicontohkan dengan menggunakan database mysql.



Database db_beritaportal dengan Table users(id,fullname,username,password,email). Kemudian koneksi database harus kita setting terlebih dahulu agar terhubung dengan aplikasi, untuk koneksi database dilakukan didalam folder

application/config/database.php

Setting koneksi database

```
$active_group = 'default';
$active_record = TRUE;

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

Membuat models

Setelah database dan koneksi selesai dihubungkan , kemudian kita akan membuat Class dan Fungtions yang ada pada folder **application/models** dengan membuat file baru dengan nama m_panel.php dengan isi sebagai berikut :

```
1  <?php
2
3  class M_panel extends CI_Model {
4      function __construct() {
5          parent::__construct();
6      }
7
8      function login($where = ''){
9          return $this->db->query("select * from users $where;");
10     }
11 }
12
13 ?>
```

coding diatas menjelaskan Class yang dibuat dengan nama M_panel yang memiliki function dengan nama login. Function login memanggil database user dengan perintah select

Membuat Controller

Setelah membuat models ,selanjutnya membuat sebuah Controller dengan nama C_panel.php dengan membuat perintah perintah perulangan dalam pembuatannya. Langsung saja ke codingnya sebagai berikut :

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

class C_panel extends CI_Controller {

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

        $this->load->model('M_panel');
        $this->load->library('form_validation');
    }

    function index(){
        $data   =   array(
            'title' => ' .:: Login Administrator ::. ',
            'eror'  => '',
        );
        $this->load->view('panel/login', $data);
    }


    function logout(){
        $this->session->sess_destroy();
        redirect('');
    }
}
```

function_construct (){ : Bertugas melakukan set nilai default atau bisa dikatakan menjalankan proses default ketika dijalankan dengan berisikan

$this->load->model('M_panel');

$this->load->library('form_validation');

Dalam secara default perintah itu menjalankan model yang telah dibuat yaitu M_panel dan library stardart dari folder system yakni form_validation.

function index(){ : Menjalankan pemanggilan perintah awal yang dilakukan dengan

memanggil index dan perintah Method

$this->load->view('panel/login',$data);

perintah ini memanggil bagian view untuk menampilkan halaman yang terdapat pada folder panel/login.php yang nantikan akan terhubungkan dengan data array dengan nama $data.

function logout(){ : membuat perintah logout yang bertugas memutuskan dengan library session.

$this->session->sess_destroy();

Perintah ini mengambil dari library session dengan perintah sess_destroy(); atau bisa dikatakan merusak atau memutuskan hubungan.

Autentikasi

Perintah autentifikasi adalah perintah perulangan yang diletakkan pada form untuk mempermudah dalam keamanan data supaya tidak sembarangan orang bisa masuk dengan mudahnya. Ini bertujuan memberikan error apabila anda salah dalam menginputkan nilai data pada sebuah form. Biasanya form yang belum anda masukkan dengan benar akan memberitahukan anda bahwa masukkan anda salah

```
20      function auth()
21      {
22          if($_POST)
23          {
24              $this->form_validation->set_rules('username','Username'.'required|trim|xss_clean');
25              $this->form_validation->set_rules('password','Password'.'required|trim|xss_clean');
26
27              if($this->form_validation->run() == FALSE)
28              {
29                  redirect('');
30              }
31
32              $username   = $this->input->post('username');
33              $password   = $this->encrypt->sha1($this->input->post('password'));
34
35              $auth       = $this->M_panel->login("where username = '$username' and password= '$password'")->result_array();
36              if($auth != NULL) // jika
37              {
38                  $data = array(
39                      'username'      => $auth[0]['username'],
40                      'email'         => $auth[0]['email'],
41                      'fullname'      => $auth[0]['fullname'],
42                  );
43                  $this->session->set_userdata('login',$data);
44                  redirect('panel/home');
45              }
46              else
47              {
48                  $data = array(
49                      'title'         => '.:: Eror Login :  Login Adminisrator Koroku CMS ::.',
50                      'eror'          => '
51                          <div class="box-body">
52                              <div class="alert alert-danger alert-dismissable">
53                                  <i class="fa fa-ban"></i>
54                                  <button type="button" class="close" data-dismiss="alert" aria-hidden="true">×</button>
55                                  <b>Kesalahan!</b>Periksa Kembali Username / Password Anda.
56                              </div>
57                          </div>
58                      ',
59                  );
60                  $this->load->view('panel/login',$data);
61              }
62
63          }
64          else
65          {
66              echo "Page not found";
67          }
68      }
```

function auth(){ : dalam perintah ini terdapat beberapa perulangan yang menggunakan If Else .

Perintah **If($_POST)** mengartikan bahwa jika form inputnya telah terisi data maka akan diarahkan kepada validation dengan rule **username** dan **password** yang harus sama dengan **name** yang diberikan kepada form input .

```
<div class="form-group">
    <input type="text" name="username" class="form-control" placeholder="Username" autofocus required=""/>
</div>
<div class="form-group">
    <input type="password" name="password" class="form-control" placeholder="Password" required="" />
</div>
```

Gambar diatas adalah form yang terletak pada folder view sebagai tampilan menu form.

required : maksud di atas adalah wajib diisi

Apabila validasi form yang dijalankan adalah **False** atau salah dia akan melakukan redirect(''); yang artinya dikembalikan. (**$this->form_validation->run()== FALSE**)

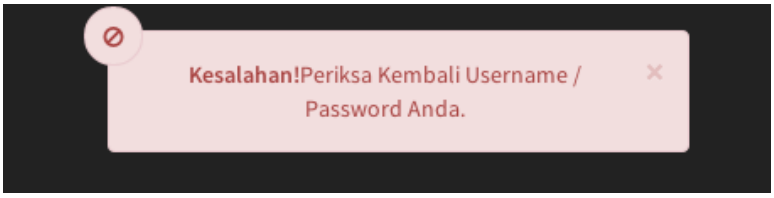
```
$username   = $this->input->post('username');
$password   = $this->encrypt->sha1($this->input->post('password'));

$auth       = $this->M_panel->login("where username = '$username' and password= '$password'")->result_array();
```

Pada perintah ini $**username** dan $**password** akan diarahkan kepada ke object $**auth** dengan mencocokan database yang telah dibuat tadi pada models dengan Class **M_panel** dan function **login**.

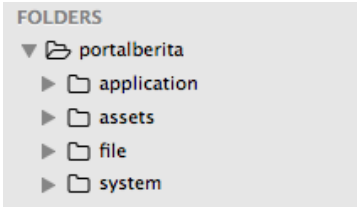
If ($auth != NULL) // jika auth ini tidak kosong maka proses akan dijalankan session untuk masuk ke login kemudian di redirect ke dalam url **panel/home**

dan else nya jika data yang diinputkan tidak sesuai dengan database makan akan muncul perintah



Membuat halaman view

Membuat tampilan aplikasi diletakkan didalam application/view. Kita buat contoh didalam view ada folder panel yang berisi form untuk login dengan isi file login.php . CSS dan file-file tampilanya kita letakkan pada folder **assets**. Folder asset kita buat diluar dan sejajar dengan folder application agar terlihat file yang terstruktur.



Kita buka folder aplikasi dan meletakkan file html didalam file **login.php**

```
<!DOCTYPE html>
<html class="bg-black">
    <head>
        <meta charset="UTF-8">
        <title><?php echo  $title; ?></title>
        <meta name="author" content="AIRPUTIH">
        <meta content='width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no' name='viewport'>
        <link href="<?php echo base_url();?>assets/admin/css/bootstrap.min.css" rel="stylesheet" type="text/css" />
        <link href="<?php echo base_url();?>assets/admin/css/font-awesome.min.css" rel="stylesheet" type="text/css" />
        <link href="<?php echo base_url();?>assets/admin/css/AdminLTE.css" rel="stylesheet" type="text/css" />

    </head>
    <body class="bg-black">

        <div class="form-box" id="login-box">
            <div class="header">Sign In</div>
            <form action="<?php echo base_url();?>panel/auth" method="POST">
                <div class="body bg-gray">
                    <div class="form-group">
                        <input type="text" name="username" class="form-control" placeholder="Username" autofocus required=""/>
                    </div>
                    <div class="form-group">
                        <input type="password" name="password" class="form-control" placeholder="Password" required="" />
                    </div>
                </div>
                <div class="footer">
                    <button type="submit" class="btn bg-olive btn-block">Login</button>
                </div>
            </form>

            <br />
            <br />

        <center>
            <div class="alert alert-info alert-dismissable">
                                    <i class="fa fa-info"></i>
                                    <b>This Admin Panel Developed By <a href="" target="_blank">Tutorial CI</a></b>
                                </div>
            <?php echo $eror; ?>
        </center>
        </div>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/2.0.2/jquery.min.js"></script>
        <script src="<?php echo base_url();?>assets/admin/js/bootstrap.min.js" type="text/javascript"></script>

    </body>
</html>
```

Pertama dalam melakukan oleh template adalah agar koneksi dengan assets yang teridi dari css dan jd terbaca oleh framework CI adalah sebagai berikut :

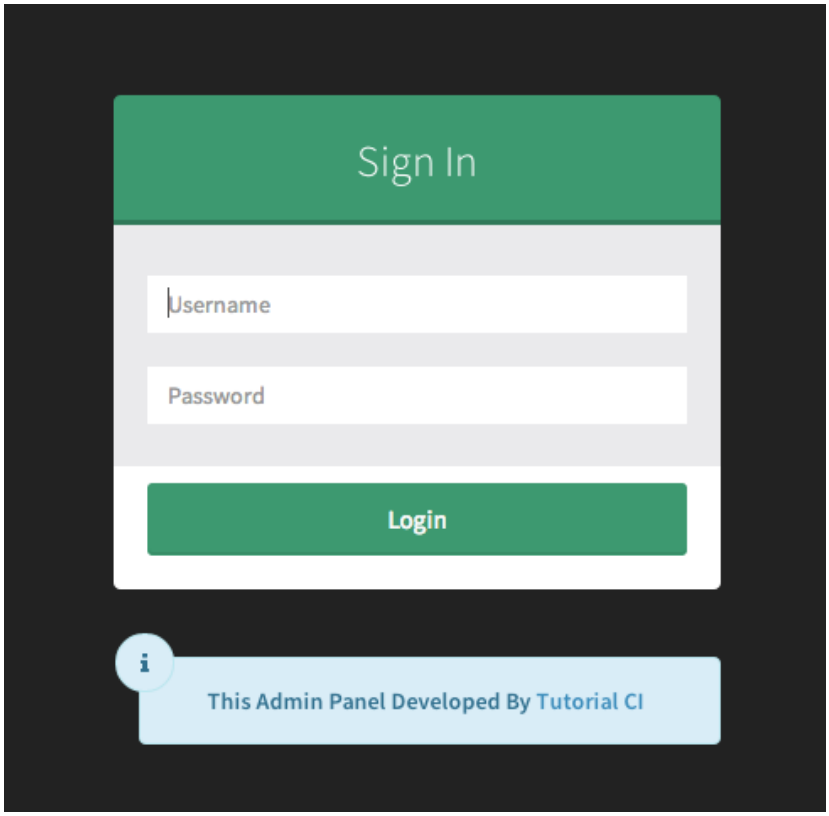
```
      <link href="<?php echo base_url();?>assets/admin/css/bootstrap.min.css"
rel="stylesheet" type="text/css" />
            <link href="<?php echo base_url();?>assets/admin/css/font-
awesome.min.css" rel="stylesheet" type="text/css" />
            <link href="<?php echo base_url();?
>assets/admin/css/AdminLTE.css" rel="stylesheet" type="text/css" />
```

yang awalnya adalah

```
      <link href="admin/css/bootstrap.min.css" rel="stylesheet"
type="text/css" />
```

```
            <link href="admin/css/font-awesome.min.css" rel="stylesheet"
type="text/css" />
            <link href="admin/css/AdminLTE.css" rel="stylesheet"
type="text/css" />
```

Hasilnya adalah sebagai berikut :



Tahap selanjutnya adalah kita membuat rule nya , rule tersebut kita buat di application/config/ routes.php. Rule tersebut dibuat untuk membuat url / arah alamat tujuan aplikasi ,tujuan tersebut memberikan arah kemana perintah URL tersebut akan di implementasikan oleh Controller.

```
41  $route['default_controller']          = "welcome";
42  $route['404_override']                = '';
43  $route['panel']                       = 'C_panel';
44  $route['panel/auth']                  = 'C_panel/auth';
45  $route['panel/logout']                = 'C_panel/logout';
46  $route['panel/home']                  = 'C_admin';
```

Aplikasi website yang dibuat tadi dengan nama perintah **localhost/portalberita/panel**

yang akan menuju pada Controller dengan nama perintah C_panel.php dengan Class C_panel yang akan menampilkan function index () yang memanggil view ('panel/login')

```
function index(){
            $data  =      array(
                  'title'       => ' .:: Login Administrator ::. ',
                  'eror' => '',
            );
            $this->load->view('panel/login', $data);
      }
```

$route['panel/auth'] = 'C_panel/auth';

Perintah ini adalah alamat autentifikasi form login yang dilakukan apabila form yang diinputkan salah maka alamat ini akan muncul.

C_panel/auth : mengartikan perintah ini ada pada **Controller** dengan nama **C_panel.php** dengan Class **C_panel** dan pada function dengan mana **auth**

$route['panel/logout'] = 'C_panel/logout';

Perintah ini adalah perintah yang dilakukan pada saat kita melakukan logout.

```
function logout(){
            $this->session->sess_destroy();
            redirect('');
      }
```

$route['panel/home'] = 'C_admin';

Perintah ini jalan pada saat login dengan form bisa masuk dengan benar, maka akan masuk pada perintah home yang perintah home ini menuju pada Controller dengan C_admin.php Class C_admin.

```
36              if($auth != NULL) // jika
37              {
38                  $data = array(
39                      'username'      => $auth[0]['username'],
40                      'email'         => $auth[0]['email'],
41                      'fullname'      => $auth[0]['fullname'],
42                  );
43                  $this->session->set_userdata('login',$data);
44                  redirect('panel/home');
45              }
46              else
47              {
```

Controller C_admin.

Pada Controller ini yang dijalankan adalah function index dari Class C_admin yang ada pada

```
1   <?php  if ( ! defined('BASEPATH')) exit('No direct script access allowed');
2
3   class C_admin extends CI_Controller {
4       function __construct(){
5           parent::__construct();
6           $this->load->library('gravatar');
7           $this->load->model('m_admin');
8
9           if($this->session->userdata('login') != TRUE)
10          {
11              redirect('panel');
12          }
13      }
14      function index(){
15
16          $data = array(
17              'title'         => 'Welcome Administrator',
18
19
20          );
21          $this->load->view('panel/head',$data);
22          $this->load->view('panel/home');
23          $this->load->view('panel/footer');
24      }
25
```

C_admin.php didalam Controller. Pada bagian ini mengarahkan tampilan hasil login kepada panel/home yang isi tampilannya ada pada menu folder view/panel.

Jika dalam input proses sukses maka akan masuk kedalam halaman baru.

Perintah : **$route['panel/home'] = 'C_admin' ;**

misalnya : www.portalberita.com/panel anda akan menuju halama login. Dan bila

sukses makan anda akan menuju secara ke halaman Home yang dengan alamat www.portalberita.com/panel/home

Maka dari perintah route diatas bila sudah masuk **panel/home** maka akan menampilkan view (tampilan) menuju :

```
$this->load->view('panel/head',$data) ;
$this->load->view('panel/home') ;
$this->load->view('panel/footer') ;
```

jadi anda harus membuat 3 tampilan html untuk bagian atas , tengah dan bawah dengan alamat :

- view/panel/head.php
- view/panel/home.php
- view/panel/footer.php

ini dibuat untuk mempermudah anda dalam membuat halaman baru semisal halaman contact jadi anda tinggal membuat Controller mirip diatas dengan perintah menampilkannya diubah.

```
$this->load->view('panel/head',$data) ;
$this->load->view('panel/contact') ;
$this->load->view('panel/footer') ;
```

Maka dengan begini anda tinggal membuat halaman Contact saja.

Catatan : Untuk belajar Framework CodeIgniter ini memang kelihatannya akan terlihat sulit akan tetapi seperti anda belajar bersepeda anda akan jatuh bangun untuk bisa. CI setidaknya seperti itu bila anda terbiasa maka akan menjadi biasa . CI menggunakan konsep model MVC jadi anda tinggal mempelajari pada bagian Model-View-Controller saja untuk bisa mengelola Framework ini.

Perkumpulan AirPutih

Jl. Warga Raya No. 30 RT/RW 003/003

Pejaten Barat - Pasar Minggu

Jakarta Selatan 12510

2014