Dasar-dasar Pembuatan Website Dengan CodeIgniter

Harry Yunanto

http://www.iorme.net

Bab 1 PENDAHULUAN

Dalam dunia pemrograman website ada banyak sekali bahasa pemrograman yang bisa digunakan. Salah satu bahasa pemrograman yang sangat terkenal dan banyak sekali digunakan oleh para pembuat website adalah PHP.

PHP (PHP: Hypertext Preprocessor) menjadi sebuah kekuatan yang menakjubkan di dunia internet. PHP sudah menjadi bahasa pemrograman untuk semua jenis website. Mulai dari website pribadi yang sederhana sampai website perdagangan online yang super rumit, banyak yang dibuat menggunakan PHP. Bisa dibilang PHP ikut berperan dalam berbagai revolusi yang terjadi di dunia website. Munculnya PHP menjadi tanda atau semangat baru yang disebut dengan website dinamis.

Dahulu sebuah website tidaklah interaktif seperti sekarang. Website hanya dibuat menggunakan HTML saja. Meskipun HTML juga bisa dibilang sebagai sebuah bahasa pemrograman tetapi tentu saja beda maksudnya dengan PHP. HTML digunakan untuk membuat tampilan dari sebuah halaman website. Bagaimana bentuk halaman, bagaimana warnanya, bagaimana font yang digunakan semuanya ditentukan dengan menggunakan HTML. Sedangkan PHP bisa dibilang sebagai mesin dari website. Apa yang akan dilakukan jika sebuah tombol submit diklik, bagaimana penyimpanan datanya dan lain-lain dikerjakan dengan menggunakan PHP. Adanya PHP merupakan salah satu pendorong website statis yang hanya dibuat menggunakan HTML menjadi hampir lenyap. PHP membuat pengunjung website bisa berinteraksi dengan pemilik website atau dengan pengunjung lainnya.

Salah satu keunggulan PHP dibanding bahasa pemrograman lainnya adalah PHP dapat didapatkan secara gratis, meskipun bukan berarti karena gratis kemampuannya pas-pasan. PHP sangat powerfull. Terbukti dengan banyaknya website yang dibangun menggunakan PHP. PHP juga terkenal lebih aman

daripada bahasa pemrograman website yang lain. PHP juga sudah mendukung OOP (Object Oriented Programming). Apalagi dengan PHP versi 5 yang disebut-sebut lebih matang dalam dukungannya duntuk OOP dibanding versi sebelumnya.

Semakin lama semakin lama semakin banyak orang yang menggunakan PHP, semakin banyak website yang dibangun dengan memanfaatkan PHP. Oleh karena itu semakin banyak kode-kode yang dibuat berbagai tujuan mulai dari kode sederhana sampai kode yang rumit. Dan kabar baiknya banyak sekali orang yang berhati baik d iluar sana yang rela membagi kode-kode yang sudah mereka buat kepada siapa saja. Maka banyaklah class-class atau fungsi-fungsi PHP yang tersedia luas di internet, class dan fungsi yang bisa kita dapatkan secara gratis untuk pembuatan website kita.

Banyak sekali situs yang menyediakan koleksi library kode atau fungsi/class dalam PHP misalnya saja PEAR (PHP Extension and Application Repository, http://pear.php.net) dan http://www.phpclasses.org. dengan adanya situs-situs semacam itu maka kita tidak perlu terlalu bersusah-susah lagi dalam membuat website menggunakan PHP, karena banyak kode-kode yang kita butuhkan bisa kita temukan di internet dan kita gunakan secara bebas.

Selain banyak yang sudah menyediakan fungsi/class PHP, banyak pula beredar CMS (Content Management System) untuk membuat website, seperti joomla (http://www.joomla.org) atau WordPress (http://www.wordpress.org). Penggunaan CMS juga sangat membantu dalam pembuatan website, dengan CMS maka untuk membuat sebuah website lengkap dengan berbagai fitur bisa dibuat dalam hitungan waktu beberapa menit saja. Hal ini tentu menyenangkan meskipun di satu sisi bisa disebut mesin pembunuh para programmer website. Bayangkan saja jika nantinya orang tidak mau lagi menyewa programmer website tetapi lebih memilih menggunakan CMS yang siap pakai. Meskipun begitu masih banyak hal-hal atau tugas-tugas dari sebuah website yang tidak

bisa dikerjakan menggunakan CMS, karena CMS dibuat untuk mengerjakan tugas-tugas yang lebih umum maka untuk tugas-tugas spesifik tentu saja programmer masih dibutuhkan.

Selain itu masih ada framework PHP, misalnya saja simphony (http://www.simphony-project.org) atau CodeIgniter (http://www.codeigniter.com). Sebuah framework sudah menyediakan berbagai macam fungsi/class yang kita butuhkan dalam pembuatan website. Framework juga menyediakan lingkungan pengembangan yang harus kita ikuti dalam pembuatan website menggunakan framework tersebut.

1.1 Apa Itu Framework?

Sebenarnya apa sih yang disebut dengan framework itu?? Framework bisa diartikan sebagai alat yang digunakan untuk membantu pekerjaan. Karena untuk pembuatan website maka framework disini dapat diartikan sebagai alat yang dapat digunakan untuk mempermudah pembuatan website.

Jika dengan CMS maka kita tinggal menjalankan saja tidak perlu lagi memikirkan untuk menulis kode program sendiri, tetapi tidak demikian dengan framework. Menggunakan framework kita masih harus menulis kode, bedanya kode-kode yang kita tulis harus menyesuaikan dengan lingkungan framework yang kita gunakan. Memang konsekuensinya kita harus belajar lagi lingkungan pengembangan berdasarkan framework yang kita gunakan, tetapi hal itu akan terbayar setelah kita menguasai dan bisa menggunakan framework tersebut.

Sebuah framework selain menyediakan lingkungan pengembangan sendiri-sendiri juga menyediakan berbagai macam fungsi siap pakai yang bisa kita gunakan dalam pembautan wenbsite. Sehingga tidak perlu kaget jika akan banyak kode atau fungsi yang terlihat tidak seperti biasanya, karena fungsi-fungsi tersebut merupakan fungsi bawaan framework dan bukan fungsi asli dari PHP. Fungsi tersebut terkadang merupakan pengembangan atau penyesuaian

fungsi asli PHP agar lebih mudah digunakan atau agar lebih sesuai dengan kebutuhan pengguna.

1.2 CodeIgniter

CodeIgniter merupakan salah satu dari sekian banyak framework PHP yang ada. CodeIgniter dikembangkan oleh Rick Ellis (http://www.ellislab.com).

Tujuan dari pembuatan framework CodeIgniter ini menurut user manualnya adalah untuk menghasilkan framework yang akan dapat digunakan untuk pengembangan proyek pembuatan website secara lebih cepat dibandingkan dengan pembuatan website dengan cara koding secara manual, dengan menyediakan banyak sekali pustaka yang dibutuhkan dalam pembuatan website, dengan antarmuka yang sederhana dan struktur logika untuk mengakses pustaka yang dibutuhkan. CodeIgniter membiarkan kita untuk memfokuskan diri pada pembuatan website dengan meminimalkan pembuatan kode untuk berbagai tujuan pembuatan website.

1.3 Kenapa CodeIgniter?

Ada banyak sekali framework PHP yang beredar di internet. Lantas kenap kita memilih CodeIgniter bukan yang lain?? Berikut alasan-alasannya:

1. Gratis.
   CodeIgniter dilisensikan dibawah lisensi Apache/BSD style open source license, ini berarti kita dapat menggunakannya sesuai dengan keinginan kita.
2. Berjalan di PHP versi 4 dan 5.
   Sekarang ini PHP sudah mencapai versi ke 5, meskipun begitu masih banyak orang yang tetap menggunakan PHP versi 4, oleh sebab itu CodeIgniter dikembangkan agar tetap kompatibel dengan PHP versi 4 dan dapat dijalankan pada PGP versi 5.
3. Ringan dan cepat.

Secara default CodeIgniter hanya berjalan dengan me load beberapa pustaka saja, dengan demikian hanya membutuhkan resource yang sedikit sehingga ringan dan cepat dijalankan. Pustaka-pustaka lain yang nantinya akan digunakan bisa di load sesuai dengan kebutuhan.

4. Menggunakan MVC.
   CodeIgniter menggunakan lingkungan pengembangan dengan metode Model View Controller (MVC) yang membedakan antara logika dan presentasi/tampilan, sehingga tugas bisa lebih mudah dipecah-pecah. Ada bagian yang khusus membuat tampilan dan bagian yang membuat core programnya.
5. Dokumentasi.
   Salah satu hal yang bisa dijadikan barometer apakah sebuah aplikasi benar-benar dikembangkan atau tidak bisa dilihat dari dokumentasinya. Dalam hal ini CodeIgniter sangat luar biasa, terdapat dokumentasi yang sangat lengkap tentang semua hal yang ada dalam CodeIgniter. Mulai dari langkah instalasi sampai dokumentasi fungsi-fungsi nya tersedia. Adanya dokumentasi sangat memudahkan bagi pemula dalam mempelajari lingkungan pengembangan website dengan CodeIgniter.
6. Pustaka yang lengkap.
   CodeIgniter dilengkapi dengan berbagai pustaka siap pakai untuk berbagai kebutuhan, misalnya saja koneksi database, email, session dan cookies, keamanan, manipulasi gambar dan banyak lagi.

1.4 Fitur-Fitur CodeIgniter

Berikut fitur-fitur yang didukung oleh CodeIgniter :

1. Sistem berbasis Model View Controller
2. Kompatibel dengan PHP versi 4.
3. Ringan dan Cepat.
4. Terdapat dukungan untuk berbagai basis data.
5. Mendukung Active Record Database.
6. Mendukung form dan validasi data masukan.

7. Keamanan dan XSS filtering.
8. Tersedia pengaturan session.
9. Tersedia class untuk mengirim email.
10. Tersedia class untuk manipulasi gambar (cropping,resizing, rotate dan lain-lain).
11. Tersedia class untuk upload file.
12. Tersedia class yang mendukung transfer via FTP.
13. Mendukung lokalisasi bahasa.
14. Tersedia class untuk melakukan pagination (membuat tampilan perhalaman).
15. Mendukung enkripsi data.
16. Mendukung benchmarking.
17. Mendukung caching.
18. Pencatatan error yang terjadi.
19. Tersedia class untuk membuat calendar.
20. Tersedia class untuk mengetahui user agent, misalnya tipe browser dan sistem operasi yang digunakan pengunjung.
21. Tersedia class untuk pembuatan template website.
22. Tersedia class untuk membuat trackback.
23. Tersedia pustaka untuk bekerja dengan XMP-RPC.
24. Menghasilkan clean URL.
25. URI routing yang felksibel.
26. Mendukung hooks, ekstensi class dan plugin.
27. Memiliki helper yang sangat banyak jumlahnya.

1.5 Model View Controller

Seperti sudah disebutkan di muka bahwa CodeIgniter menerapkan lingkungan pengembangan dengan metode MVC (Model View Controller). MVC memisahkan antara logika pembuatan kode dengan pembuatan template atau

tampilan website. Penggunaan MVC membuat pembuatan sebuah proyek website menjadi lebih terstruktur dan lebih sederhana.

Secara sederhana konsep MVC terdiri dari tiga bagian yaitu bagian Model, bagian View dan bagian Controller. Didalam website dinamis setidaknya terdiri dari 3 hal yang paling pokok, yaitu basis data, logika aplikasi dan cara menampilkan halaman wesite. 3 hal tersebut direpresentasikan dengan MVC yaitu model untuk basis data, view untuk cara menampilkan halaman website dan controller untuk logika aplikasi.

1. Model

   Merepresantiskan struktur data dari website yang bisa berupa basis data maupun data lain, misalnya dalam bentuk file teks atau file xml. Biasanya didalam model akan berisi class dan fungsi untuk mengambil, melakukan update dan menghapus data website. Karena sebuah website biasanya memnggunakan basis data dalam menyimpan data maka bagian Model biasanya akan berhubungan dengan perintah-perintah query SQL.

   Model bisa dibilang khusus digunakan untuk melakukan koneksi ke basis data oleh karena itu logika-logika pemrograman yang berada didalam model juga harus yang berhubungan dengan basis data. Misalnya saja pemilihan kondisi tetapi untuk memilih melakukan query yang mana.

2. View

   Merupakan informasi yang ditampilkan kepada pengunjung website. Sebisa mungkin didalam View tidak berisi logika-logika kode tetapi hanya berisi variabel-variabel yang berisi data yang siap ditampilkan. View bisa dibilang adalah halaman website yang dibuat menggunakan HTML dengan bantuan CSS atau JavaScript.

   Didalam view jangan pernah ada kode untuk melakukan koneksi ke basis data. View hanya dikhususkan untuk menampilkan data-data hasil dari model dan controller.

3. Controller

   Controller merupakan penghubung antara Model dan View. Didalam Controller inilah terdapat class dan fungsi-fungsi yang memproses permintaan dari View kedalam struktur data didalam Model.

   Controller juga tidak boleh berisi kode untuk mengakses basis data. Tugas controller adalah menyediakan berbagai variabel yang akan ditampilkan di view, memanggil model untuk melakukan akses ke basis data, menyediakan penanganan error, mengerjakan proses logika dari aplikasi serta melakukan validasi atau cek terhadap input.

Jadi secara singkat urutan dari sebuah request adalah sebagai berikut : user berhubungan dengan view, dimana didalam view inilah semua informasi ditampilkan. Saat user melakukan permintaan atau request, misal klik tombol maka request tersebut akan diproses oleh Controller. Apa yang harus dilakukan, data apa yang diinginkan, apakah ingin melihat data, atau memasukan data atau mungkin melakukan validasi data terlebih dahulu, semua diproses oleh Controller. Kemudian Controller akan meminta Model untuk menyelesaikan request, entah itu melakukan query atau apapun. Dari Model, data akan dikirim kembali untuk di proses lebih lanjut di dalam Controller dan baru dari Controller data akan ditampilkan di View.

Bab 2 INSTALASI

2.1 Kebutuhan Server

Seperti yang kita tahu PHP adalah bahasa pemrograman website yang berjalan disisi server, oleh karena itu untuk dapat menjalankan website yang dibuat menggunakan PHP, di komputer harus terinstall aplikasi web server yang mendukung PHP. Banyak sekali aplikasi web server yang ada beredar, salah satu web server yang sangat terkenal dan juga bersifat bebas adalah web server Apache, sebuah web server yang digunakan pada sebagian server yang ada di internet.

Sudah disebutkan di muka bahwa CodeIgniter kompatibel dengan PHP versi 4 maupun versi 5. Untuk dapat menjalankan CodeIgniter dengan baik diperlukan PHP setidaknya versi 4.3.2.

Untuk website yang ingin menggunakan basis data sebagai tempat penyimpanan datanya maka tidak perlu kuatir karena CodeIgniter juga mendukung berbagai jenis server basis data, yaitu MySQL, MySQLi, MS SQL, Postgre, Oracle, SQLite, and ODBC.

CodeIgniter juga bisa dijalankan di semua sistem operasi yang bisa menjalankan aplikasi-aplikasi diatas, baik Windows, Linux, BSD dan yang lainnya.

2.2 Instalasi Server

Jika anda pengguna Linux dan menggunakan distro-distro besar semacam Debian, Fedora Core, CentOS dan distro lainnya maka tidak perlu kuatir karena distro-distro tersebut biasanya sudah menyertakan semua aplikasi yang kita butuhkan untuk dapat menjalankan CodeIgniter, mulai dari web server Apache, PHP dan basis data MySQL atau postgreSQL.

Bagi pengguna Windows pun tidak perlu kuatir karena aplikasi-aplikasi tersebut juga tersedia untuk sistem operasi Windows.

Untuk memudahkan bisa dicoba aplikasi XAMPP (http://www.apachefriends.org). XAMPP sudah menyertakan webs server Apache, PHP dan basis data MySQL, sehingga kita tidak perlu pusing-pusing lagi dengan instalasi dan pengaturan yang harus dilakukan karena XAMPP sudah menyediakan semuanya. XAMPP juga tersedia untuk sistem operasi Windows dan Linux sehingga para pengguna Linux yang malas melakukan instalasi aplikasi yang terkadang membingungkan di linux seperti saya bisa mendapatkan lingkungan server untuk website secara cepat dan mudah.

Untuk pengguna Linux, paket instalasi XAMPP merupakan sebuah file archive tar.gz. Untuk proses instalasi, masuk ke Terminal kemudian jalankan perintah berikut :

shell # tar xzvf xampp-linux-1.6.5.a.tar.gz -C /opt/

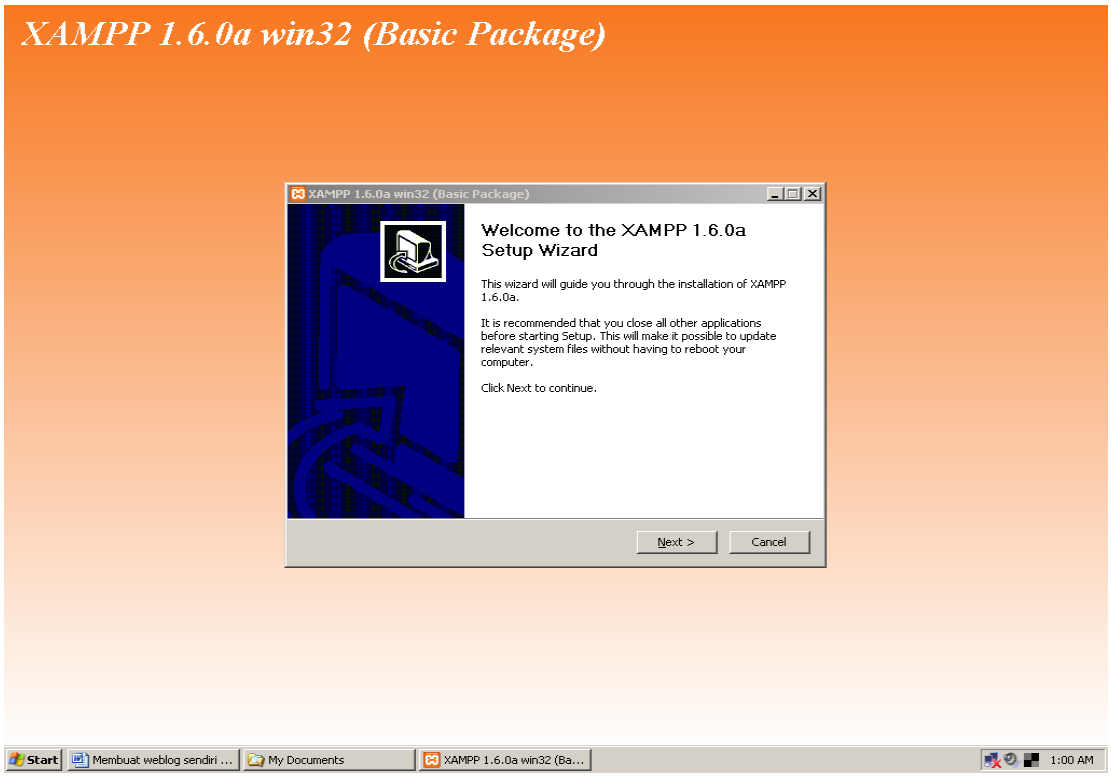
Kemudian untuk menjalankan XAMPP, jalankan perintag berikut :

shell # cd /opt/lampp/
shell # ./lampp start

Karena proses instalasi sebenarnya hanyalah proses ekstrak biasa maka selain dengan menggunakan perintah baris bisa juga dengan menggunakan aplikasi seperti file-roller atau ark.

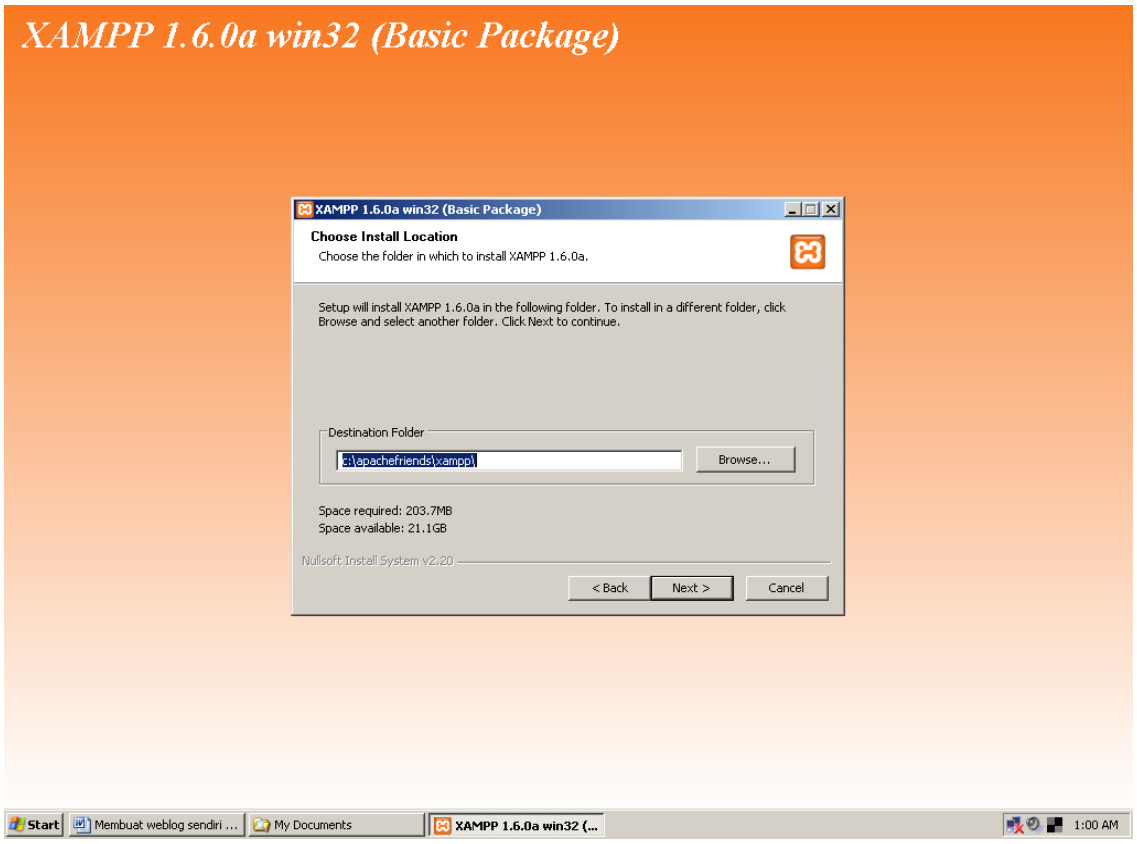
Berikut adalah langkah-langkah instalasi XAMPP for Windows, klik dua kali pada file installer XAMPP, maka akan muncul tampilan seperti berikut :

Klik next untuk melanjutkan instalasi



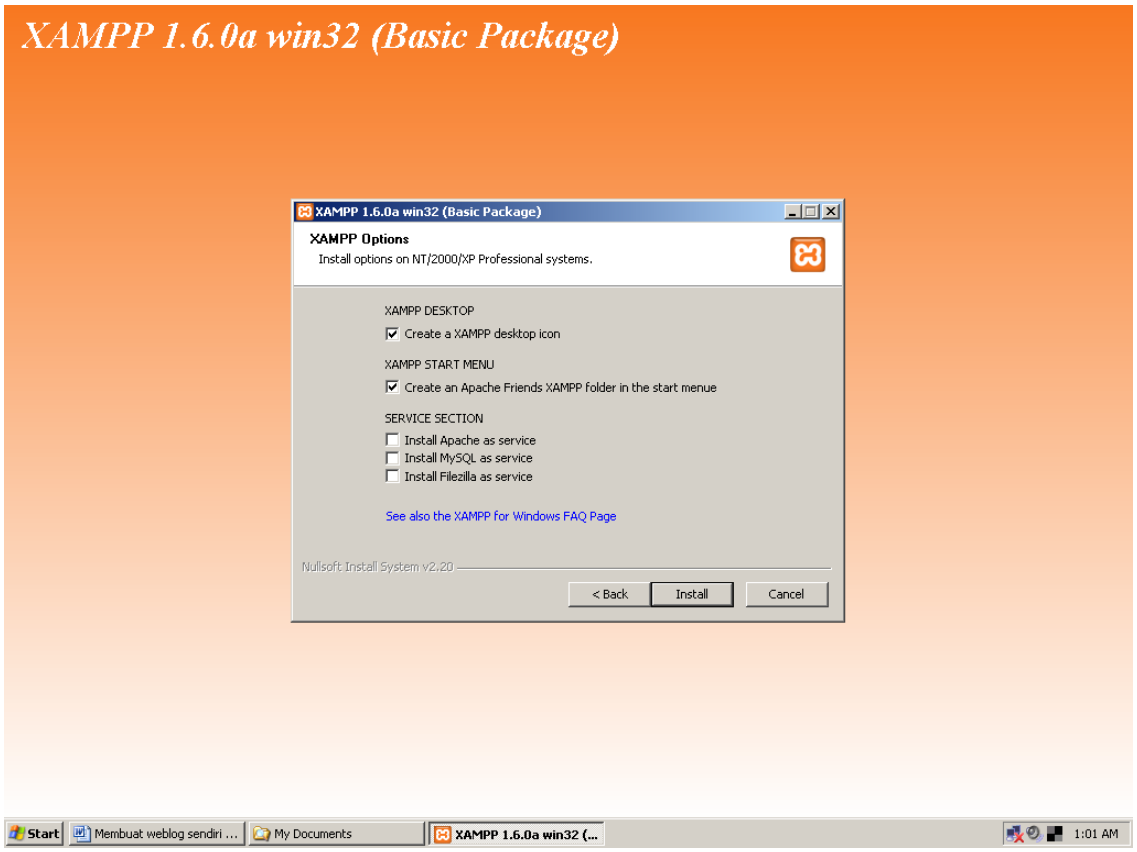
Gambar : Tampilan awal instalasi

Masukan folder tempat XAMPP akan dinstall. Defaultnya adalah C:\apachefriedns\xampp tetapi anda bisa saja memilih tempat lain. Kemudian klik next untuk melanjutkan.



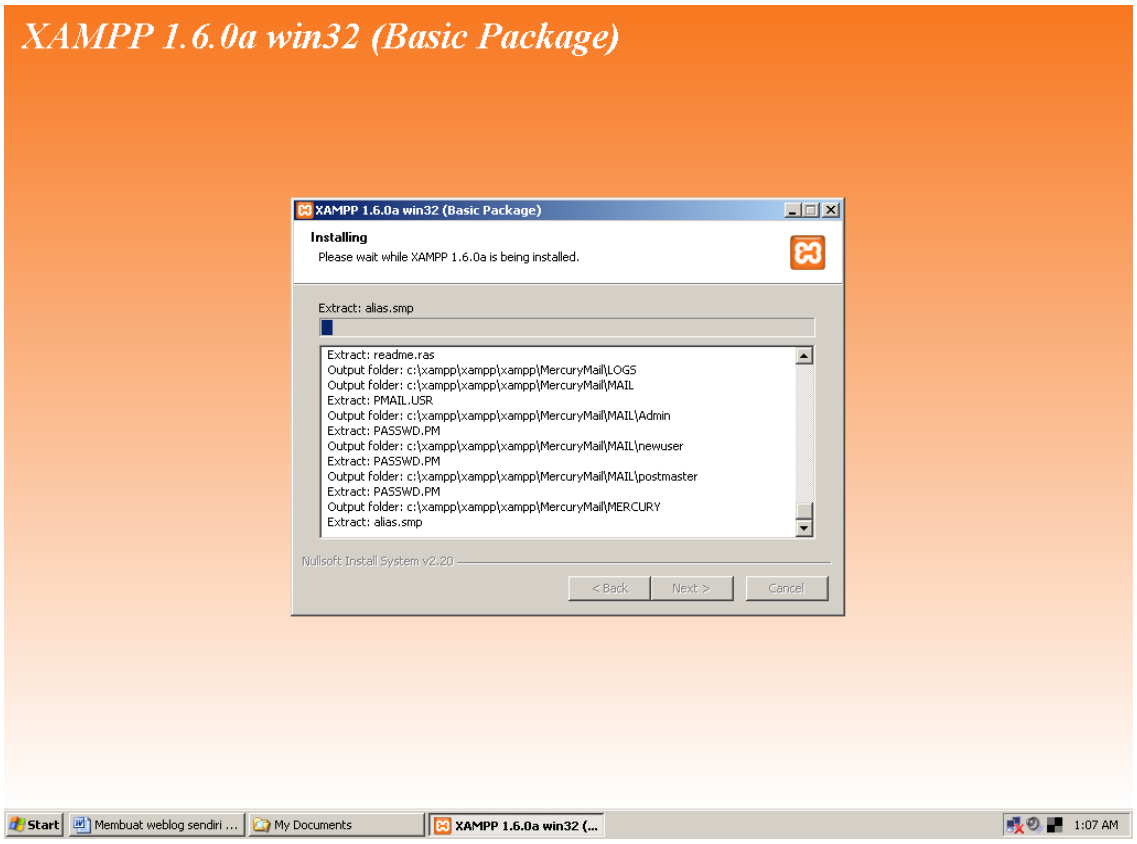
Gambar : memasukan folder tujuan instalasi

Jika ingin membuat shortcut di desktop silahkan check pada bagian Create XAMPP desktop icon. Jika ingin membuat menu di start check create an Apache Friends XAMPP folder in the start menu. Jika kita melakukan check pada bagian service section maka XAMPP akan diinstall sebagai service sehingga akan otomatis dijalankan saat komputer dinyalakan. Kemudian klik install.

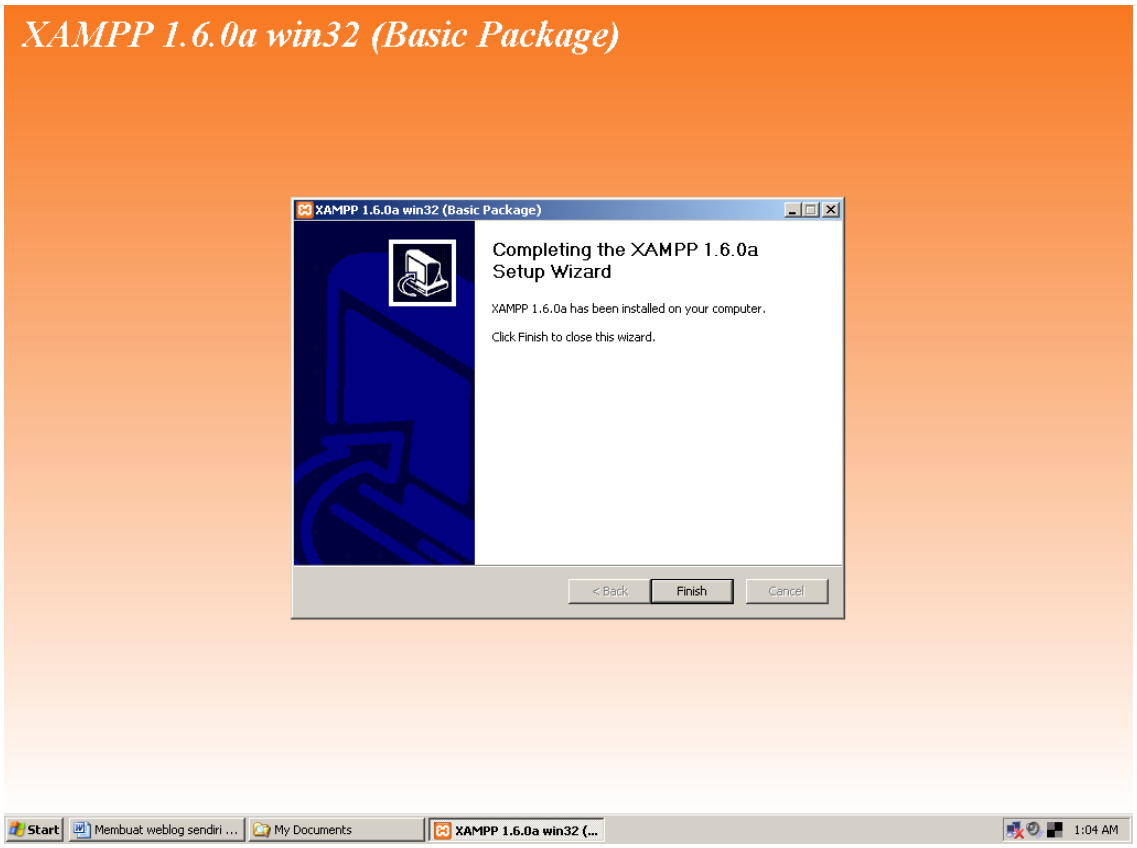


Gambar : Opsi instalasi

Proses instalasi sedang berlangsung, tunggu beberapa saat.

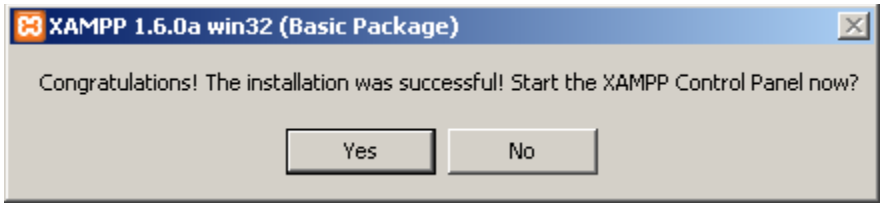


Gambar : Proses instalasi



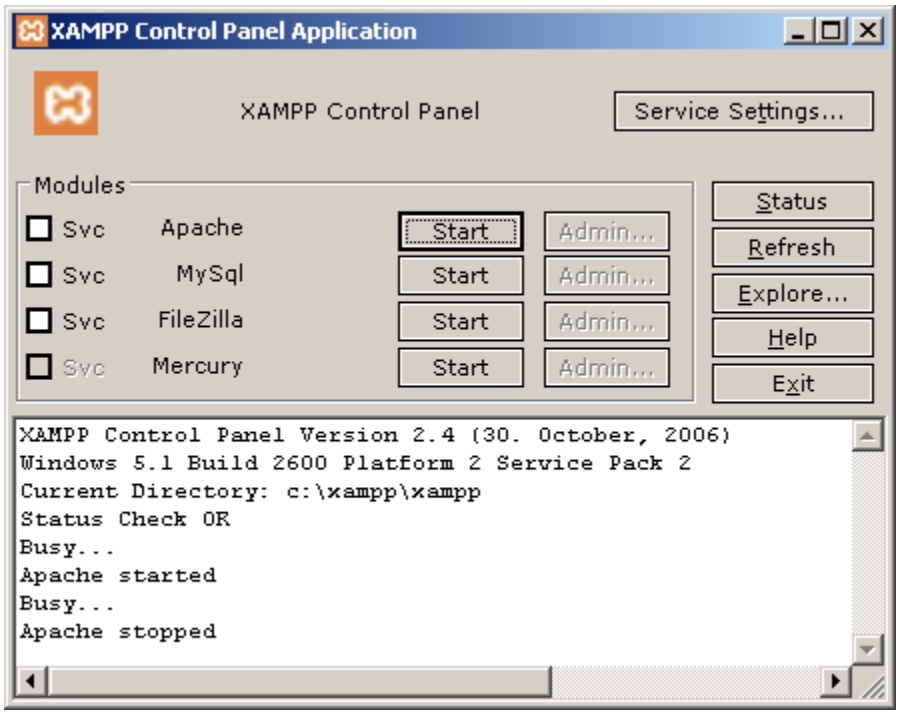
Gambar : Instalasi Selesai

Jika instalasi sudah selsai maka klik Finish. Maka akan muncul window berikut .



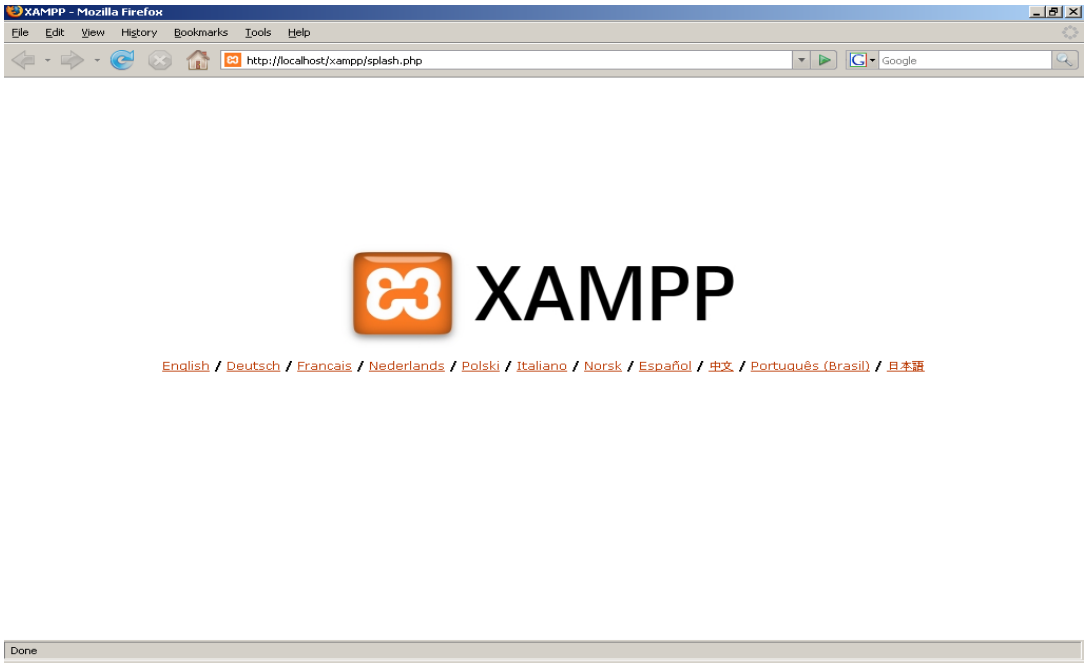
Klik Yes jika ingin menjalankan XAMPP Control Panel.

Klik tombol start pada bagian Apache dan MySQL. Maka XAMPP sudah siap digunakan.



Gambar : Control Panel XAMPP

Setelah itu kita bisa mengakses halaman website XAMPP di komputer kita dengan memasukan alamat http://localhost/ pada address bar web browser.



Gambar : XAMPP for Windows

2.3 Instalasi CodeIgniter

Meskipun namanya instalasi tetapi karena CodeIgniter adalah aplikasi berbasis website maka yang sebenarnya kita lakukan adalah meng-copy folder aplikasi CodeIgniter kedalam DocumentRoot dari web server yang sudah kita install sebelumnya. Bukan melakukan instalasi seperti pada aplikasi sistem.

Sebelum melakukan instalasi yang perlu dilakukan pertama kali adalah mendapatkan kode sumber dari CodeIgniter itu sendiri, jika tidak punya maka bagaimana mungkin bisa melakukan instalasi. CodeIgniter bisa di download http://www.codeigniter.com/download.php, versi terbaru sampai buku ini ditulis adalah versi 1.6.1. Untuk melakukan instalasi cukup ektraks file hasil download, yaitu file CodeIgniter_1.6.1.zip, kemudian letakan folder hasil ekstrak tadi di DocumentRoot web server, yaitu folder htdocs didalam direktori C:\\apachefriends\xampp bagi yang menggunakan XAMPP di Windows. Folder hasil ekstraks tersebut bisa dirubah namanya agar memudahkan kita, misal di rename menjadi ci (default hasil ekstrak adalah CodeIgniter_1.6.1). Didalam

folder tersebut ada 2 folder lagi yaitu system dan user_guide, silahkan saja untuk memindahkan folder user_guide ke tempat lain karena inti aplikasi ada di folder system dan folder user_guide berisi dokumentasi dari CodeIgniter.

Didalam folder system masih terdapat beberapa folder lain, yang akan sering kita akses adalah folder application karena di folder inilah script-script kita akan disimpan. Beberapa folder yang ada di dalam direktori system adalah :

1. application, di folder inilah kode-kode yang kita buat nantinya akan disimpan didalam folder yang sesuai. Model disimpan di folder models, Controller di folder controller dan View di folder views. Folder-folder yang terdapat di dalam direktori application adalah :
    a. models untuk menyimpan model yang kita buat.
    b. controller untuk menyimpan controller.
    c. views untuk menyimpan view tampilan website.
    d. config untuk menyimpan konfigurasi website yang akan kita buat. Mulai dari konfigurasi dasat, basis data, routing dan lain-lain.
    e. error berisi file-file yang akan ditampilkan jika ada error pada script yang kita buat
    f. libraries untuk menyimpan pustaka yang kita tambahkan atau pustaka buatan kita sendiri.
    g. hooks untuk meyimpan hook yang kita buat.
2. cache, untuk meyimpan caching dari website.
3. codeigniter, berisi file-file yang akan me-load inti dari framework.
4. database, berisi class-class yang akan digunakan untuk bekerja dengan basis data, termasuk didalamnya driver-driver untuk beberapa server basis data yang didukung oleh CodeIgniter.
5. fonts, digunakan untuk menyimpan font yang nanti akan kita gunakan di dalam website.
6. helpers, berisi helper.
7. language, digunakan untuk menyimpan file-file dukungan bahasa.

8. libraries, berisi pustaka-pustaka yang disediakan untuk digunakan untuk pembuatan website.
9. logs, berisi file-file catatan yang mencatat log dari website kita.
10. plugins, untuk menyimpan plugin.
11. scafollding, berisi file-file untuk keperluan scafollding.

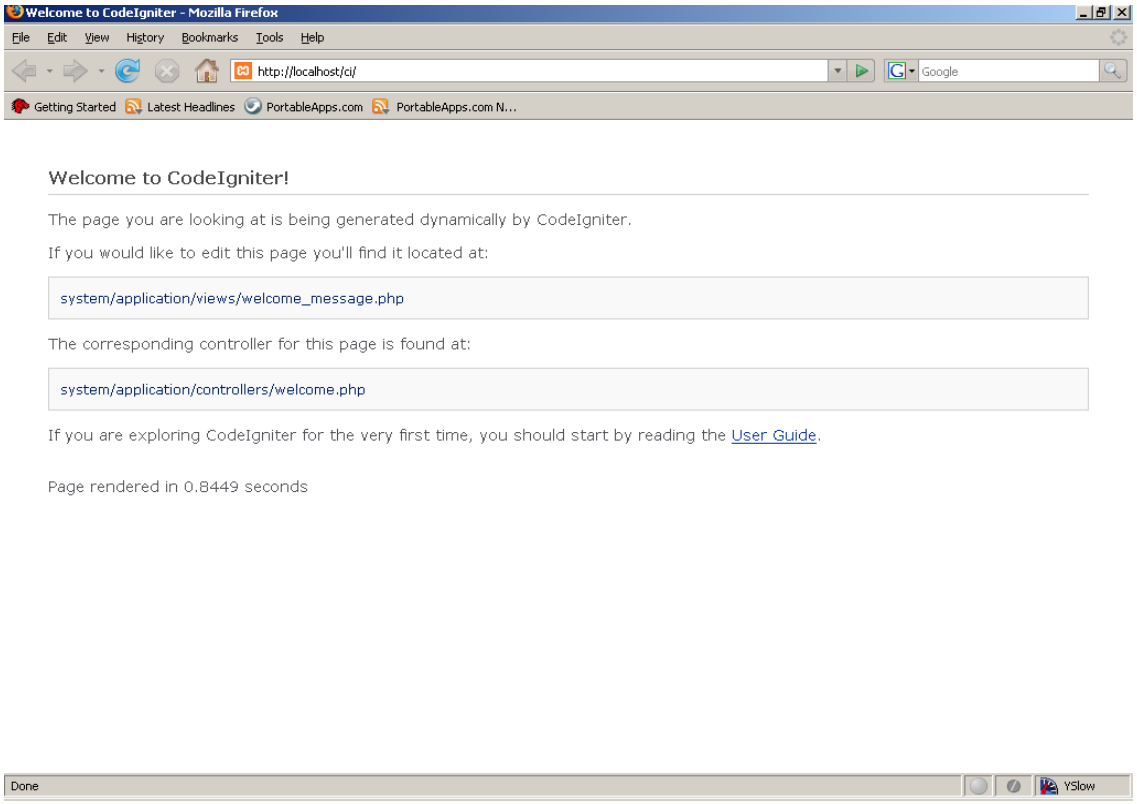
Setelah itu, misal foldernya bernama ci maka website bisa diakses lewat http://localhost/ci, tapi sebelum itu jangan lupa untuk melakukan sedikit modifikasi pada bagian konfigurasi CodeIgniter. Buka file system/application/config/config.php. Di file inilah konfigurasi dasar CodeIgniter disimpan. Yang perlu dirubah untuk instalasi awal ini adalah pada bagian base URL Setelah nanti kita siap membuat sebuah website yang sebenarnya maka akan banyak pengaturan yang harus dilakukan.

$config['base_url'] = "www.your-site.com";

Pada bagian www.your-site.com ganti dengan url anda. Karena saya hanya mencoba di localhost dengan nama folder ci, maka base URL nya saya ganti menjadi :

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

Base URL adalah URL default dari website yang kita buat, secara default URL tersebut akan selalu digunakan untuk pembuatan link di halaman website, tentu saja untuk link-link internal bukan link eksternal ke website lain. Setelah itu silahkan dibuka alamat http://localhost/ci.



Gambar : Tampilan awal setelah CodeIgniter berhasil di install.

Instalasi sudah selesai dan sekarang kita siap membuat website menggunakan framework CodeIgniter.

2.4 Konfigurasi Basis Data

File konfigurasi untuk basis data terdapat didalam direktori system/application/config dengan nama file database.php. Disinilah kita dapat memasukan konfigurasi basis data sesuai dengan aplikasi basis data yang kita miliki, misalnya saja hostname temapt server basis data berada, nama basis data yang akan digunakan, nama user yang digunakan untuk mengakses basis data beserta passwordnya.

$active_group = "default";
$active_record = TRUE;

$db['default']['hostname'] = "localhost";

$db['default']['username'] =”[“nama_user]”;

$db['default']['password'] = "[password]";

$db['default']['database'] = "[nama_database]";

$db['default']['dbdriver'] = "[nama_basisdata]";

$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";

Keterangan:

$active_group=”default” , digunakan untuk memilih group koneksi ke basis data yang mana yang akan digunakan.

$active_record = TRUE, apakah akan me load active record class atau tidak.

$db['default']['hostname'] = "localhost", adalah nama host server basis data.

$db['default']['username'] = "[nama_user]", adalah nama user yang digunakan untuk mengakses basis data.

$db['default']['password'] = "[password]", adalah password user yang digunakan untuk mengakses basis data.

$db['default']['database'] = "[nama_basisdata]", adalah nama basis data yang digunakan untuk menyimpan data.

$db['default']['dbdriver'] = "[driver_basisdata]", adalah nama server basisdata yang digunakan, contoh diatas adalah mysql.

$db['default']['dbprefix'] = "", nama prefix tabel, misalnya saja jika semua tabel berawalan "ci_" maka masukan value "ci_".

$db['default']['pconnect'] = TRUE , apakah akan menggunakan koneksi persistent ataukah tidak.

$db['default']['db_debug'] = TRUE, apakah akan menampilkan error ataukah tidak.
$db['default']['cache_on'] = FALSE, apakah akan menggunakan caching atau tidak.
$db['default']['cachedir'] = "", jika menggunakan caching maka tentukan direktori caching yang akan digunakan.
$db['default']['char_set'] = "utf8", character set yang akan digunakan untuk berkomunikasi dengan basis data.
$db['default']['dbcollat'] = "utf8_general_ci", character collation yang akan digunakan untuk berkomunikasi dengan basis data.

Jika misalnya ingin membuat group lain maka tinggal ganti default pada array dengan nama group baru tersebut, kemudian rubah nilai variabel $active_group, misalnya saja kita membuat group baru dengan nama blog maka konfigurasinya akan menjadi :

```
$db['blog']['hostname'] = "localhost";
$db['blog']['username'] =”[“nama_user]”;
$db['blog']['password'] = "[password]";
$db['blog']['database'] = "[nama_database]";
$db['blog']['dbdriver'] = "[nama_basisdata]";
$db['blog']['dbprefix'] = "";
$db['blog']['pconnect'] = TRUE;
$db['blog']['db_debug'] = TRUE;
$db['blog']['cache_on'] = FALSE;
$db['blog']['cachedir'] = "";
$db['blog']['char_set'] = "utf8";
$db['blog']['dbcollat'] = "utf8_general_ci";
```

2.5 Alamat URL dalam CodeIgniter

CodeIgniter menghasilkan clean URL yang mudah dikenali oleh search engine dan manusia. Sebagai contoh :

www.nama-website.com/index.php/blog/post/

Dengan blog sebagai nama controller dan post adalah nama fungsi didalam controller blog. URL dalam CodeIgniter dibagi-bagi kedalam segment-segment dengan tanda slash (/) sebagai tandanya. Dalam contoh diatas blog adalah segment pertama dan post adalah segment kedua dan seterusnya.

2.6 Konfigurasi Routing

Konfigurasi routing digunakan untuk memetakan permintaan atau request kedalam class controller didalam website yang kita buat. Misalnya saja jika kita membuka alamat http://www.nama-website.com, permintaan tersebut tidak menyertakan nama controller apa yang ingin dibuka tetapi kita bisa secara default mengarahkannya agar secara otomatis akan membuka controller sesuai yang kita definisikan.

Untuk melakukan konfigurasi routing buka file konfigurasinya di direktori system/application/config dengan nama file adalah routes.php. Settingan utama yang ada adalah sebagai berikut :

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

Artinya secara default semua permintaan yang tidak menyertakan nama controllernya akan diarahkan untuk membuka controller “welcome”. Sehingga saat alamat http://www.nama-website.com dibuka secara otomatis akan membuka http://www.nama-website.com/index.php/welcome.

$route['scaffolding_trigger'] = "", digunakan jika ingin menggunakan fitur scaffolding. Kita diijinkan untuk membuat sebuah kata rahasia untuk keperluan mengakses basis data. Fitur ini nanti akan dibahas tersendiri pada bab selanjutnya.

Beberapa contoh konfigurasi routing :

$route['default_controller'] = "blog";
Secara default semua permintaan tanpa menyertakan alamat controller akan diarahkan untuk membuka controller “blog”.

$route['blog'] = "blog/hasil";
Jika membuka controller blog maka akan secara default mengarah ke fungsi “hasil”.

Kita juga bisa memanfaatkan regular expression dalam melakukan konfigurasi routing ini.

$route['blog/:any'] = "blog/hasil";
Kata apapun yang diletakan setelah blog/ baik nama fungsinya ada atau tidak maka akan diarahkan untuk membuka fungsi “hasil”.

$route['blog/([a-z]+)/(\d+)'] = "$1/id_$2";
Misalnya kita membuka alamat http://www.nama-website/blog/post/2 maka akan menghasilkan http://www.nama-website/blog/post/id_2.

Bab 3 PENGENALAN CODEIGNITER

Untuk dapat menggunakan CodeIgniter, tentu kita harus mempelajari dahulu bagaimana membuat sebuah aplikasi berbasis website menggunakan CodeIgniter. Karena bekerja dengan sebuah framework tentu tidak sama dengan bekerja dengan membuat kode secara manual dari nol. Menggunakan framework berarti kita harus mau mempelajari bagaimana melakukan pengembangan website di lingkungan framework tersebut. Selain itu lingkungan pengembangan satu framework dengan yang lain juga berbeda, meskipun kita sudah biasa menggunakan symphoni, misalnya, jika ingin menggunakan CodeIgniter kita tetap saja harus mempelajari lingkungan pengembangan di CodeIgniter.

CodeIgniter juga memiliki banyak fitur-fitur yang harus dipelajari terlebih dahulu agar bisa memanfaatkannya untuk keperluan pengembangan website. Bab ini akan membahas cara-cara membuat halaman website serderhana dengan CodeIgniter menggunakan MVC serta pengenalan bagian-bagian dari CodeIgniter seperti library, helpers dan plugin serta beberapa fitur dari CodeIgniter.

Seperti yang sudah disebutkan di muka bahwa CodeIgniter menggunakan lingkungan pengembangan dengan MVC (Model View Controller), sehingga kita harus menggunakan MVC dalam pembuatan website kita.

3.1 Hello World

Dalam CodeIgniter, controller adalah sebuah class yang diberi nama dengan sama dengan nama filenya dan berhubungan dengan alamat URI dari website. Misalnya saja kita membuat sebuah controller dengan nama test, maka controller test tersebut bisa kita panggil dengan alamat :

http://www.nama-website.com/index.php/test/

Karena nama controller harus sama dengan nama file controller yang dibuat sehingga pada alamat diatas CodeIgniter akan mencari file dengan nama test.php didalam direktori controller dan jika ditemukan controller tersebut akan dijalankan.

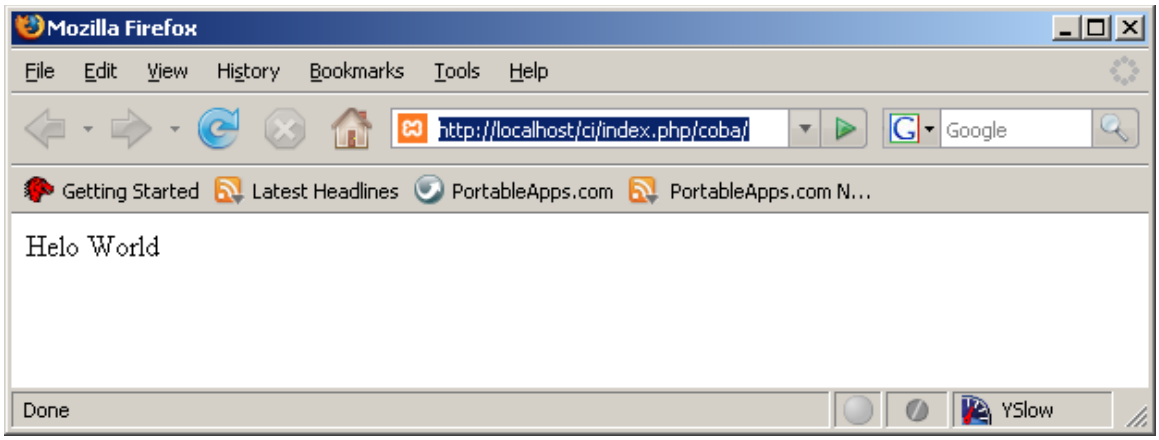
Nama class didalam controller juga harus sama dengan nama filenya. Bedanya nama class harus dimulai dengan huruf besar. Sehingga jika kita memiliki sebuah controller dengan nama test maka nama class-nya adalah Test. Jika namanya tidak sesuai maka CodeIgniter tidak akan dapat menjalankan controller yang dimaksud.

Sesuai dengan kebiasaan belajar pemrograman maka kita akan coba memulai dengan menampilkan tulisan “Hello World” menggunakan CodeIgniter, perlu diingat base URL yang kita gunakan adalah sama dengan yang digunakan dalam instalasi yaitu http://localhost/ci/.

Pertama kita buat file untuk controller, misalnya saja kita beri nama coba, oleh karena itu maka nama class-nya adalah class Coba.

```
<?php
 class Coba extends Controller {
     function index() {
      echo "Helo World";
    }
 }
?>
```

Simpan kedalam direktori controller kemudian buka alamat http://localhost/ci/index.php/coba/ di web browser.



Gambar : Helo World

class Coba extends Controller {

Kode diatas berarti kita membuat sebuah class dengan nama Coba yang merupakan sebuah controller. Fungsi index() adalah fungsi yang secara default akan dijalankan saat sebuah controller dijalankan. Kita bsia menambahkan fungsi-fungsi lain jika diperlukan kedalam controller berapapun banyaknya.

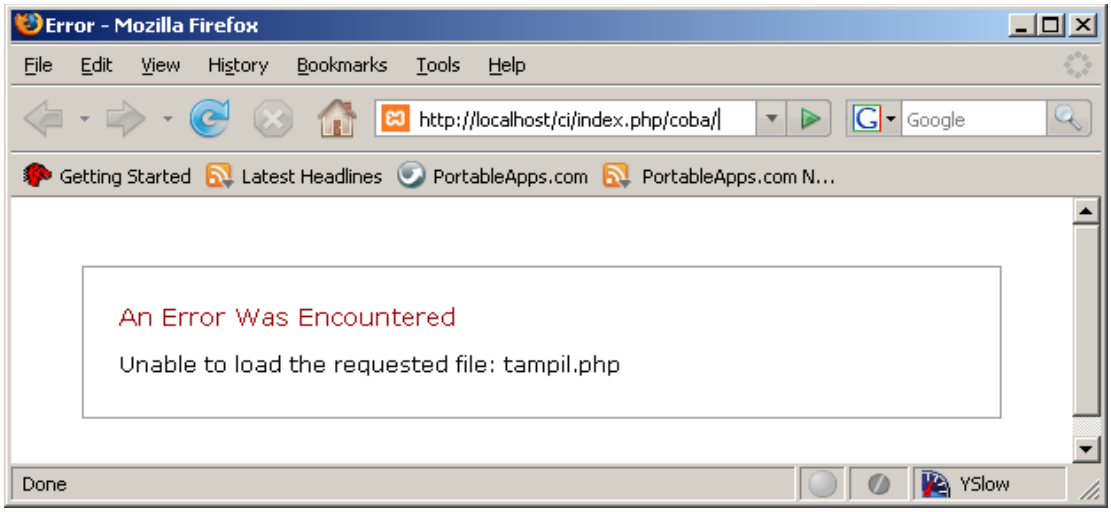
Contoh diatas belum memperlihatkan bagaimana menggunakan MVC karena kita baru menggunakan controller saja. Berikut kode untuk menampilkan “Hello World” tetapi sudah menggunakan view.

Sedikit lakukan perubahan pada file coba.php sebelumnya sehingga menjadi seperti berikut :

```
<?php
 class Coba extends Controller {
     function index() {
      $data['teks'] = "Helo World";
      $this->load->view('tampil',$data);
```

```
    }
}
?>
```

Jika kita kembali membuka alamat http://localhost/ci/ maka akan keluar error :



Gambar : Error karena view belum ada.

Error tersebut muncul karena fungsi index() kita rubah menjadi seperti berikut :

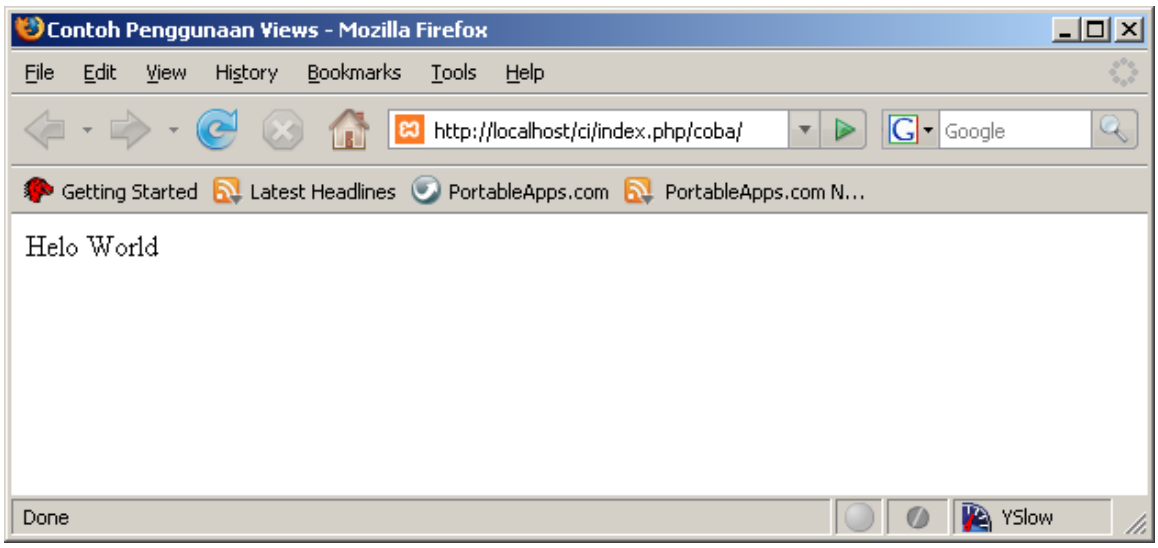
```
$data['teks'] = "Helo World";
$this->load->view('tampil',$data);
```

Maksudnya string “Hello World” disimpan kedalam array $data[“teks”] kemudian kita untuk menampilkannya kita memanggil sebuah view dengan nama tampil. Karena itu agar tulisan “Helo World” dapat ditampilkan, terlebih dahulu kita buat sebuah file dengan nama tampil.php didalam direktori views dengan isi sebagai berikut :

```
<html>
  <head>
      <title>Contoh Penggunaan Views</title>
   </head>
 <body>
    <?= $teks; ?>
 </body>
</html>
```

Didalam view inilah isi dari array $data[“teks”] ditampilkan dalam baris :

```
<?= $teks; ?>
```



Gambar : Contoh penggunaan view

Seperti halnya halaman website biasa, sebuah view tentu saja akan dibuat menggunakan HTML, kemudian tampilannya bisa dipercantik dengan memanfaatkan CSS dan JavaScript.

Karena models diperuntukan untuk berhubungan dengan struktur data website yang nantinya akan disimpan didalam basis data maka untuk dapat

menggunakan models kita harus terlebih dahulu melakukan konfigurasi dan membuat basis data yang akan dibahas pada bab selanjutnya.

Namun secara sederhananya dengan menggunakan models, maka models akan berisi fungsi-fungsi yang berisi akses ke basis data entah melihat data, melakukan update, menghapus atau yang lain. Kemudian models akan mengembalikan nilai yang akan diterima dan diproses oleh controller untuk selanjutnya ditampilkan di dalam views.

Sebagai contoh sederhana kita akan mengembalikan sebuah nilai variabel dengan models, kemudian nilai tersebut diterima oleh controller dan kemudian controller akan memanggil views untuk menampilkan hasilnya.

Untuk membuat models sama seperti halnya membuat controller, nama class dalam model harus sama dengan nama filenya tetapi dengan huruf pertama adalah huruf besar. Contohnya kita akan membuat sebuah models dengan nama model_coba, sehingga nama class-nya adalah Model_coba dan nama file-nya adalah model_coba.php.

Buat sebuah file dengan nama model_coba.php dan simpan kedalam direktori models dengan isi sebagai berikut :

```
<?php
Class Model_coba extends Model {
 function helo() {
  $teks = "Helo World";

  return $teks;
 }
}
?>
```

Kemudian lakukan perubahan pada file coba.php didalam direktori controller sehingga menjadi sebagai berikut :

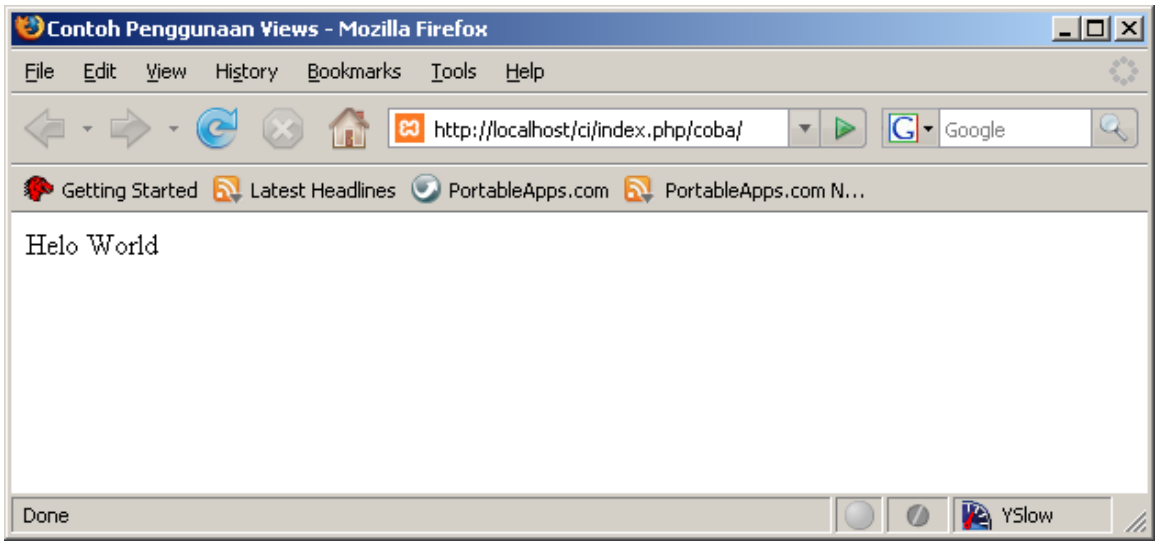
```
<?php
 class Coba extends Controller {
  function index() {
     $this->load->model('Model_coba','',TRUE);
     $data['teks'] = $this->Model_coba->helo();
    $this->load->view('tampil',$data);
  }
}
?>
```

Models yang sudah kita buat dipanggil oleh controller coba dengan *$this->load->model('Model_coba','',TRUE);*. Kemudian kita jalankan fungsi helo() didalam Model_coba dengan *$data['teks'] = $this->Model_coba->helo();*. Setelah itu panggil views dengan *$this->load->view('tampil',$data);*.

Untuk views yang digunakan tetap sama seperti sebelumnya :

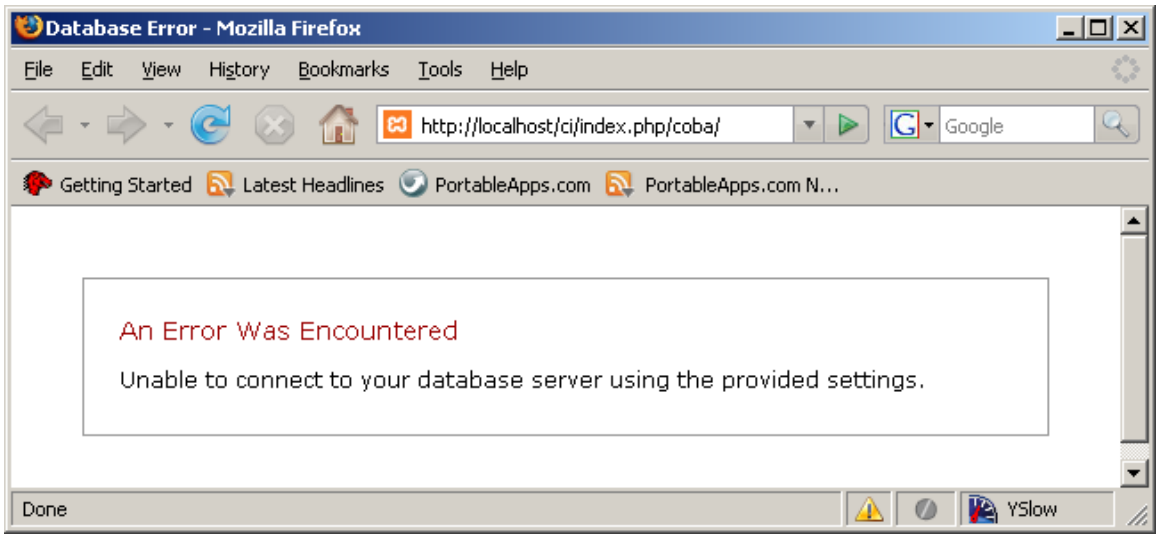
```
<html>
<head>
<title>Contoh Penggunaan Views</title>
<head>
<body>
<?= $teks; ?>
</body>
</html>
```

Kemudian buka alamat http://localhost/ci/.



Gambar : Hasil dari penggunaan models Model_coba

Bisa dilihat meskipun hasilnya sama saja, tetapi sebenarnya ada perbedaan besar daripada yang sebelumnya yaitu dengan ditambahkannya penggunaan models. Jika saat menjalankan yang anda dapati adalah error seperti berikut :



Gambar : Error saat menjankan controller

Error tersebut menandakan bahwa untuk dapat menggunakan model maka kita harus sudah terkoneksi ke server basis data, bergubung kita belum melakukan konfigurasi untuk basis data yang akan digunakan maka terjadilah error. Jika

tetap ingin menjalankan controller coba yang tadi kita buat maka terlebih dahulu lakukan konfigurasi basis data seperti pada bab sebelumnya.

3.2 Library

Library atau pustaka merupakan kumpulan class dan fungsi yang disediakan CodeIgniter. Lewat library inilah CodeIgniter menyediakan fungsi-fungsi yang bisa digunakan dalam pembuatan website. Misalnya saja library untuk akses ke basis data, library untuk mengirim email, library untuk validasi input dan lain-lain.

Library diletakan didalam direktori system/libraries atau bisa juga didalam system/application/libraries.

Untuk dapat menggunakan sebuah library maka library tersebut harus di load lebih dahulu didalam controller yang akan menggunakannya.

$this->load->library('[nama_library]');

Misalnya saja jika ingin mengunakan library validation maka library tersebut harus di-load terlebih dahulu.

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

Kita juga bisa me-load beberapa library sekaligus dengan memasukannya kedalam array.

$this->load->library(array('[library1]','library2'));

Misalnya kita ingin me-load library validation dan date maka.

$this->load->library(array('validation','date'));

Selain di-load secara manual di dalam controller, library yang ingin digunakan juga bisa di-load secara otomatis sehingga bisa digunakan pada semua controller dengan menambahkannya pada array pada file autoload.php didalam direktori system/application/config/ pada bagian :

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

Contoh untuk meload library validation dan date secara otomatis.

$autoload['libraries'] = array('validation','date');

3.3 Helpers

Helpers adalah kumpulan fungsi-fungsi dalam berbagai kategori yang seperti halnya library juga kita gunakan untuk pembuatan website. Misalnya saja URL helpers untuk berkerja dengan URL, Form helpers untuk bekerja dengan form HTML dan masih banyak lagi.

Helpers dalam CodeIgniter tidak seperti library yaitu tidak ditulis dalam OOP tetapi menggunakan struktur pemrograman biasa. Mungkin saja hal itu untuk memudahkan saja karena sebuah fungsi didalam helpers akan berguna hanya untuk menyelesaikan satu masalah. Misalnya saja fungsi anchor() digunakan untuk membuat hyperlink.

Helpers secara default tersimpan didalam direktori system/helpers meskipun begitu kita juga bisa menyimpannya didalam direktori system/application/helpers. Nama file helper memiliki standar [nama_helper]_helper.php sehingga helpers form akan memiliki file dengan nama form_helper.php.

Untuk dapat menggunakan helper maka terlebih dahulu kita me-loadnya didalam controller yang akan menggunakannya. Setelah diload helpers tersebut akan

bisa digunakan dalam controller tersebut dan dalam views yang dipanggil dari controller bersangkutan.

$this->load->helper('[nama_helper]');

Sehingga *$this->load->helper('form')* akam meload form helpers.

Kita juga bisa meload beberapa helper sekaligus dengan memasukan helpers-helpers yang akan diload kedalam array.

$this->load->helper(array('[helper1]','[helper2]'));

Misalnya saja untuk meload form dan url helpers maka perintahnya adalah :

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

Selain di load didalam controller, yang menyebabkan helpers tersebut hanya bisa digunakan didalam controller tersebut, kita juga bisa melakukan load otomatis sehingga helpers yang di load otomatis akan dapat digunakan dalam semua controller. Caranya buka file system/application/config/autoload.php kemudian masukan helpers yang akan di load dalam array di baris :

$autoload['helper'] = array();

Misalny jika ingin me-load helpers form dan url secara otomatis maka baris diatas akan menjadi :

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

3.4 Plugin

Plugin, seperti halnya helpers dan library juga berisi fungsi yang dapat digunakan untuk membantu pekerjaan pembuatan website. Bedanya satu plugin hanya untuk menyelesaikan satu buah tugas saja. Tidak seperti didalam helpers yang bisa memiliki beberapa fungsi untuk tugas yang berbeda meskipun masih berhubungan.

Selain itu jika helpers dan library termasuk kedalam inti sistem dari CodeIgniter, maka plugin merupakan tambahan yang dibuat oleh para pengembang di luar CodeIgniter dan biasanya didistribusikan sendiri.

Plugin disimpan didalam direktori system/plugin dengan standar nama file [nama_plugin]_pi.php sehingga jika ada plugin login maka nama filenya adalah login_pi.php. Sampai CodeIgniter versi 1.6.1 CodeIgniter hanya menyertakan dua buah plugin yaitu captcha dan calendar.

Untuk dapat menggunakan plugin terlebih dahulu plugin tersebut harus diload terlebih dahulu didalam controller yang akan menggunakan plugin tersebut.

$this->load->plugin('[nama_plugin]');

Misalnya kita ingin me-load plugin captcha maka perintahnya adalah :

$this->load->plugin('captcha');

Setelah itu maka plugin captcha telah siap digunakan didalam controller dan views yang dipanggil dari controller yang memangggil plugin captcha tersebut.

Plugin juga dapat di-load secara otomatis dengan menambahkannya kedalam file system/application/config/autoload.php pada bagian :

$autoload['plugin'] = array();

Jika ingin me-load plugin captcha secara otomatis maka masukan kedalam array

$autoload['plugin'] = array('captcha');

3.5 Scaffolding

Pada saat melakukan konfigurasi routing pada file system/application/config/routes.php kita menemukan sebuah istilah yang disebut scaffolding. Fitur scaffolding pada CodeIgniter menyediakan cara yang mudah dan cepat untuk melakukan akses ke basis data, misalnya untuk input, edit atau delete data tanpa perlu menggunakan aplikasi lain.

Dengan memasukan sebuah kata kedalam konfigurasi *$route['scaffolding_trigger']* kita dapat mengakses tabel dalam basis data hanya dengan meyertakan kata tersebut saat membuka alamat website. Tentunya dengan asumsi bahwa semua konfigurasi untuk melakukan akses ke server basis data sudah benar. Fitur ini sangat berguna dalam masa pengembangan dan sebaiknya tidak lagi digunakan jika website sudah kita upload ke internet.

Secara sederhana skenario penggunaan scaffolding adalah sebagai berikut: dalam membuat website dengan basis data tentu saja kita harus membuat tabel kemudian melakukan pengisian data, edit atau delete data. Hal tersebut bisa dilakukan dengan perintah-perintah SQL atau menggunakan aplikasi semacam phpMyAdmin atau MySQL-front bagi yang menggunakan MySQL. Fitur scaffolding menyediakan sebuah interface web browser untuk melakukan hal tersebut. Yang perlu dilakukan adalah buat tabelnya, kemudian tentukan kata yang akan kita gunakan untuk dapat membuka tabel yang kita buat.

Contoh, saya memiliki sebuah tabel dengan nama siswa dengan 3 field yaitu id, nama, alamat dan kelas. Untuk dapat menggunakan fitur scaffolding pada tabel tersebut pertama tentukan kata yang ingin digunakan dengan menambahkannya pada file routes.php dalam direktori system/application/config pada bagian :

```
$route['scaffolding_trigger'] = "";
```

Misalnya kita ingin menggunakan kata “aad”, maka masukan kata tersebut :

```
$route['scaffolding_trigger'] = "aad";
```

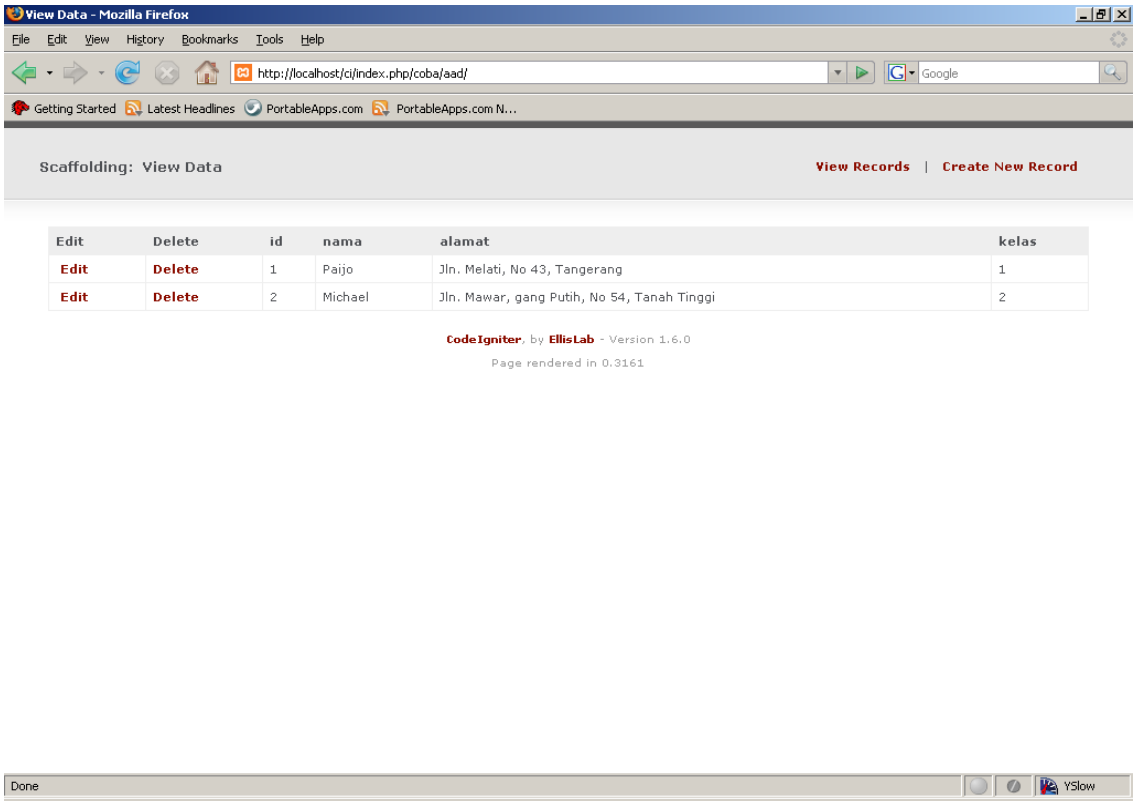
Kemudian load scaffolding didalam fungsi konstruktor controller, karena kita sudah membuat sebuah controller dengan nama Coba maka edit file coba.php dalam direktori controllers sehingga menjadi seperti berikut:

```
<?php
 class Coba extends Controller {
   function Coba () {
     parent::Controller();
     $this->load->scaffolding('siswa');
   }
  function index() {
      / /isi dari fungsi index
  }
}
?>
```

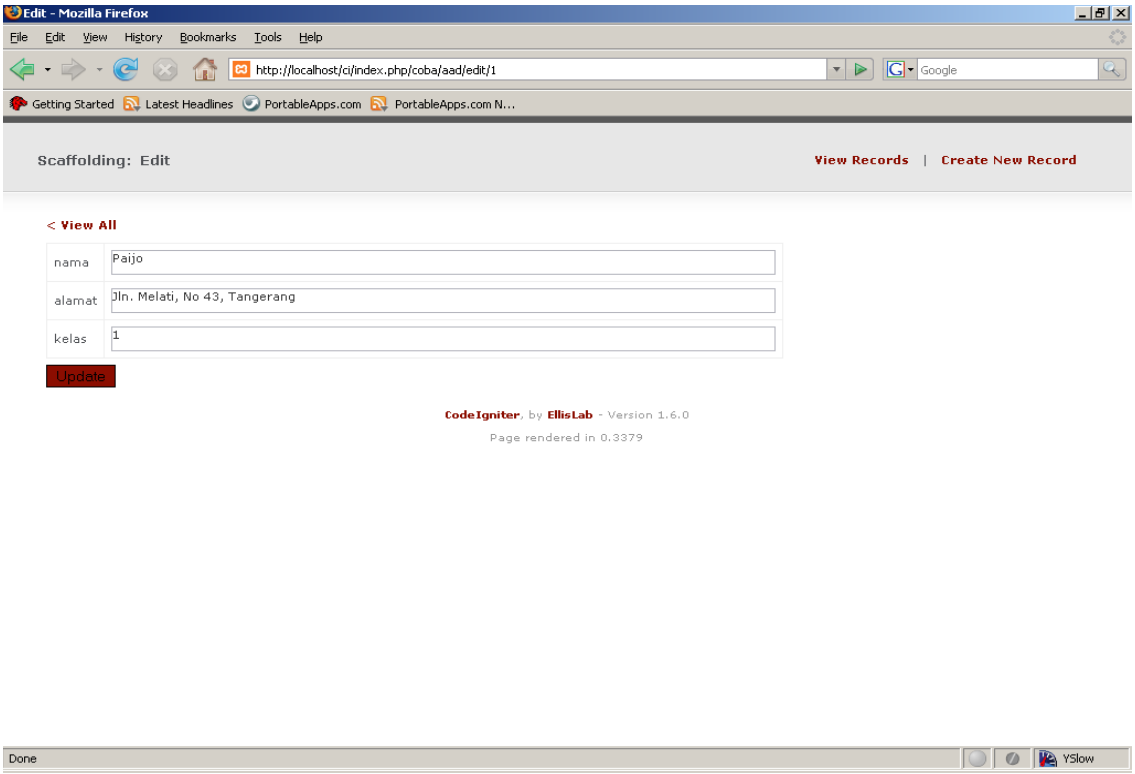
Baris :

```
$this->load->scaffolding('siswa');
```

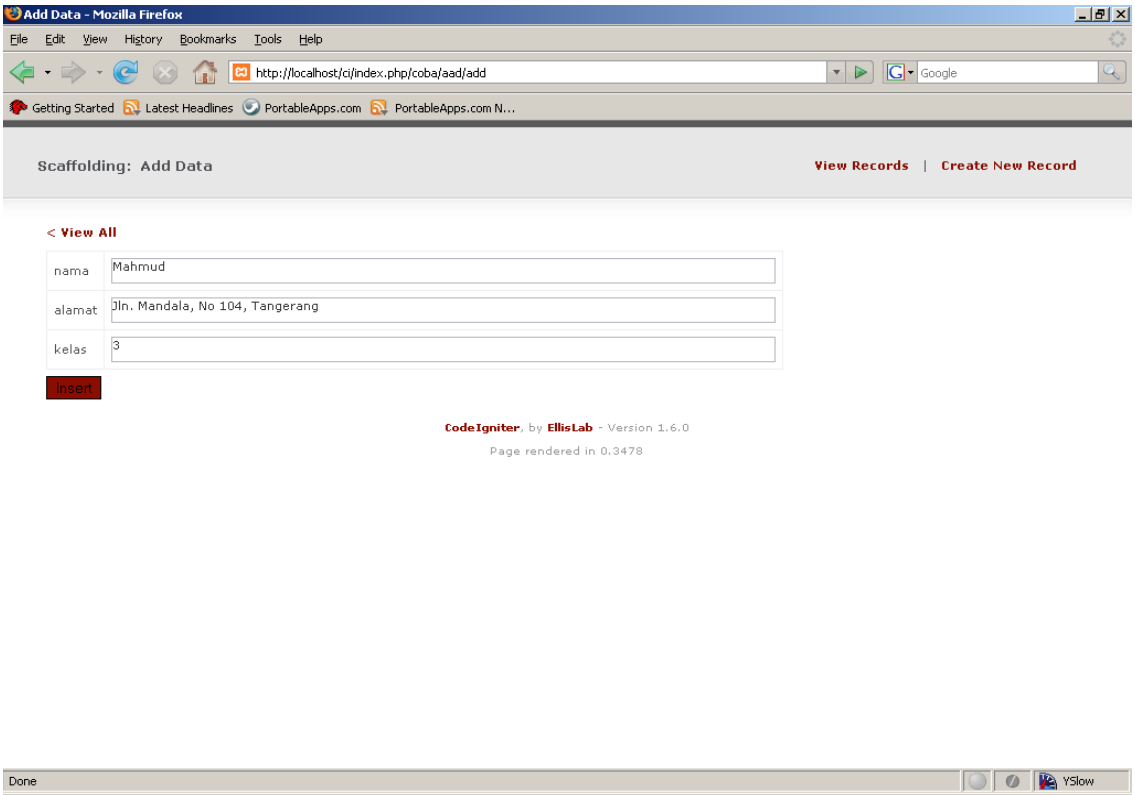
Maksudnya ktia me-load scaffolding untuk membuka tabel siswa. Untuk dapat mengaksesnya, buka alamat http://localhost/ci/index.php/coba/aad.



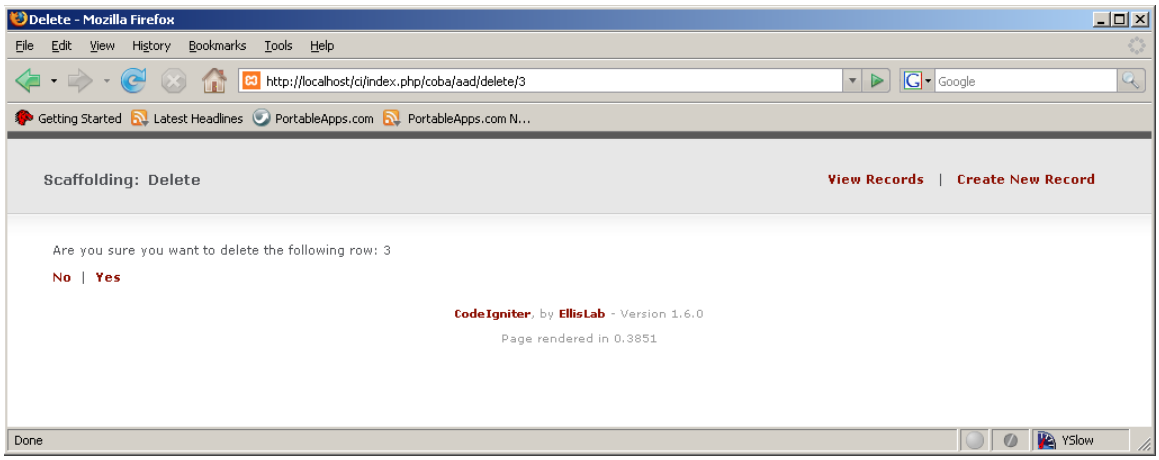
Gambar : Membuka tabel siswai menggunakan fitur scaffolding



Gambar : Halaman Edit Data



Gambar : Halaman Insert Data



Gambar : Halaman persetujuan hapus data

Seperti terlihat pada gambar maka kita bisa dengan mudah dan cepat melakukan operasi pada tabel siswa tanpa perlu membuka aplikasi lain.

3.6 Penanganan Error

Error atau kesalahan seringkali terjadi, mulai dari saat pengembangan sampai saat website sudah jadi dan di upload pun error masih sering ditemukan. Saat pengembangan pesan error sangat penting untuk mengetahui dimana kesalahan terjadi.

Secara default CodeIgniter menampilkan semua error yang terjadi. Konfigurasinya terdapat pada file index.php pada bagian :

error_reporting(E_ALL);

Fungsi tersebut membuat PHP akan menampilkan semua error yang terjadi.Jika tidak ingin semua error tampil kita bisa mengganti nilai E_ALL menjadi nilai lain agar PHP tidak menampilkan semua error yang terjadi. Karena pesan error bisa menjadi masalah. Orang lain bisa menggunakan pesan error untuk mendapatkan informasi tentang kelemahan yang ada pada website.

CodeIgniter menyediakan fungsi-fungsi yang berkaitan dengan error. Fungsi-fungsi tersebut tidak perlu di load terlebih dahulu karena sudah termasuk kedalam inti sistem CodeIgniter yang di load secara otomatis.

Tampilan untuk pesan error tersimpan pada direktori system/application/error/. Didalam direktori tersebut terdapat beberapa file error untuk berbagai keperluan.

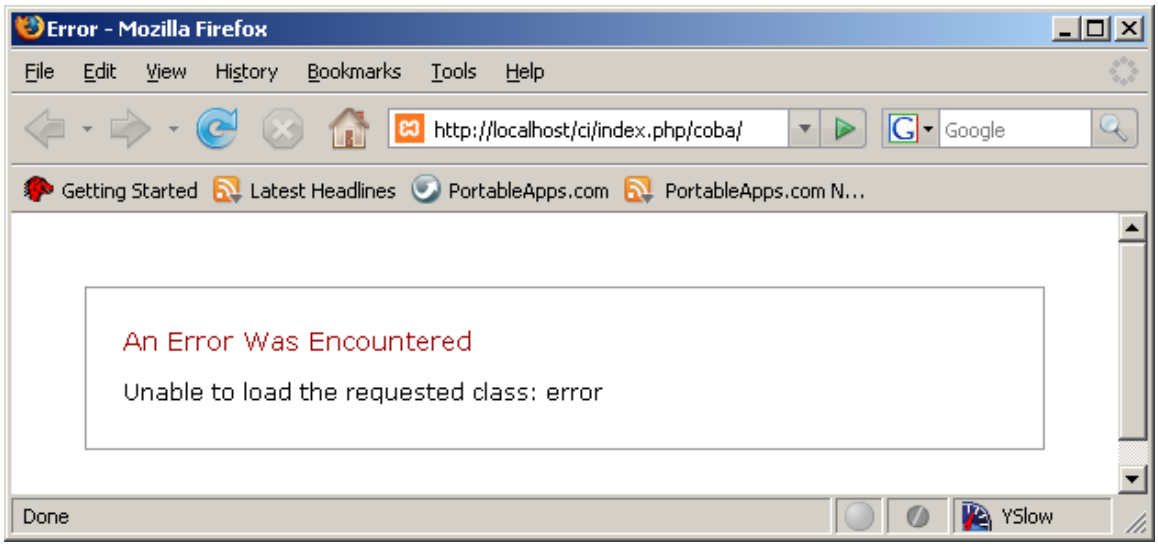
1. file error_404.php untuk menampilan error jika ada permintaan halaman yang tidak ditemukan.
2. error_db.php untuk menampilkan error yang berhubungan dengan gagalnya operasi pada basis data.
3. error_general.php untuk menampilkan error secara umum.
4. error_php.php untuk menampilkan error yang berhubungan dengan eksekusi php.

File-file tersebut hanyalah template untuk tampilan saja, sedangkan pesan mengenai error akan disesuaikan dengan error yang terjadi sesuai dengan apa yang sedang di eksekusi.

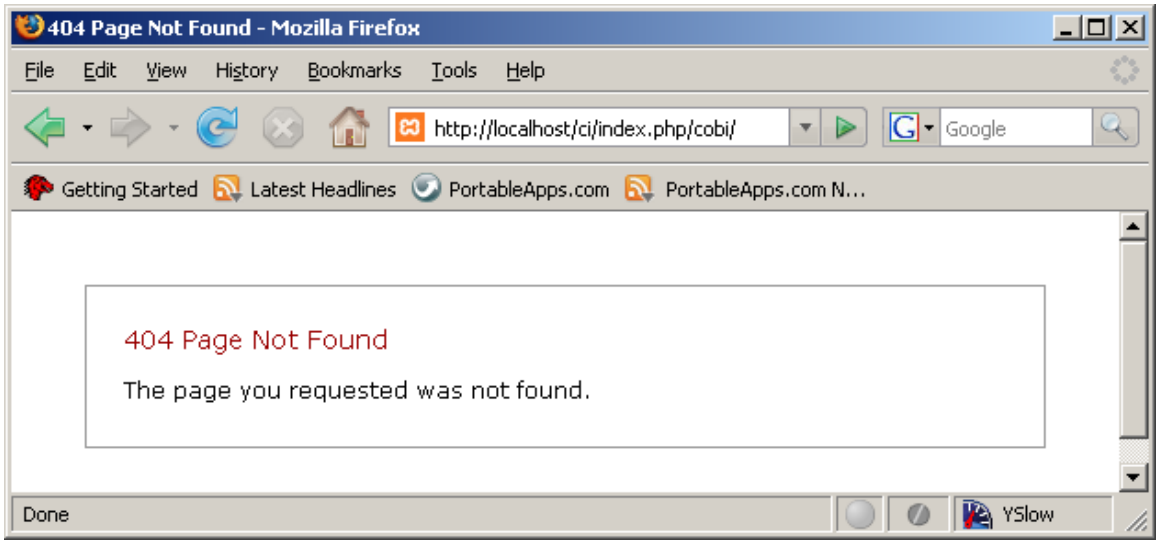
Kita dapat memanfaatkan berbagai fungsi yang berkaitan dengan error. Fungsi-fungsi tersebut diantaranya:

1. show_error('message')
   Fungsi ini akan menampilkan error berdasarkan template pada file error_general.php.
2. show_404('page')
   Fungsi ini akan menampilkan error berdasarkan template pada file error_404.php.

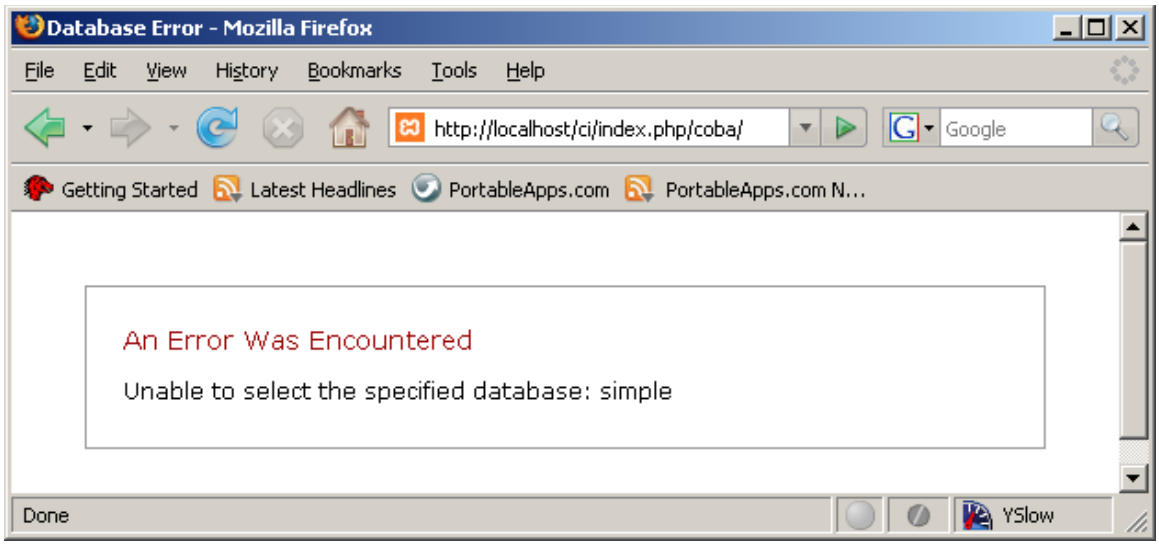
Sedangkan template pada file error_db.php dan error_php.php akan digunakan secara otomatis jika menemukan error pada operasi basis data dan eksekusi php.



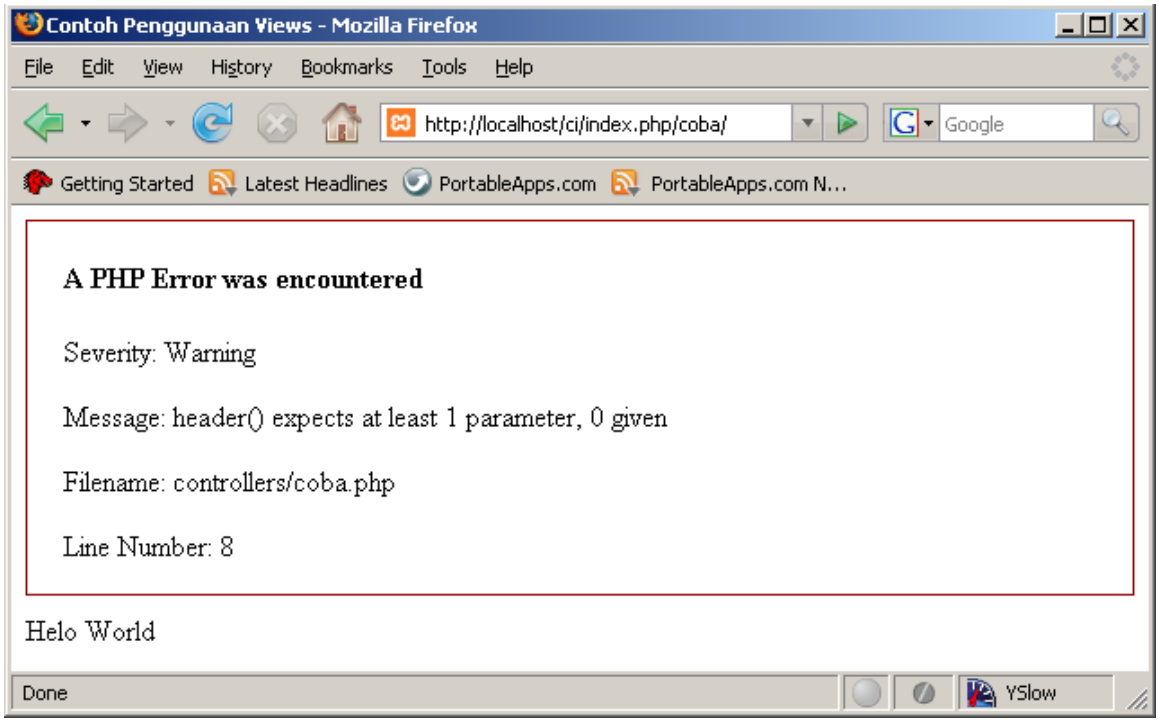
Gambar : Error karena library dengan nama error tidak ditemukan



Gambar : Error karena controller cobi tidak ditemukan



Gambar : Error karena kesalahan konfigurasi basis data (database dengan nama simple tidak ada)



Gambar : Error karena kesalahan dalam memberikan argumen pada fungsi header()

Selain hanya menampilkan pesan error, kita juga bisa me-log atau mencatat error yang terjadi kedalam file log didalam direktori system/log/ menggunakan fungsi

log_message('[level]','[pesan_error]');

Dimana [level] adalah tipe error yang disimpan sedangkan [pesan_error] adalah pesan yang dicatat. Ada tiga tipe error yang bisa digunakan, yaitu :

a. error, berupa error yang biasa terjadi.
b. debug, pesan yang dicatat bisa digunakan untuk melakukan debug terhadap aplikasi website yang kita buat.
c. info, pesan yang dicatat berupa informasi tentang proses saat eksekusi dijalankan.

Untuk dapat melakukan pencatatan error maka perlu dilakukan konfigurasi pada file config.php pada direktori system/application/config/ pada bagian :

$config['log_threshold'] = 0;

Nilai-nilai yang bisa digunakan :
0 : fasilitas pencatatan error dimatikan, tidak akan ada pesan error yang dicatat.
1 : mencatat error biasa, termasuk error pada eksekusi PHP.
2 : mencatat pesan debug.
3 : mencatat informasi tentang eksekusi.
4 : mencatat semua pesan yang ada.

Contoh isi file log dengan nilai 3 (informasi) :

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

DEBUG - 2008-01-31 14:39:39 --> Config Class Initialized
DEBUG - 2008-01-31 14:39:39 --> Hooks Class Initialized

DEBUG - 2008-01-31 14:39:39 --> Router Class Initialized
DEBUG - 2008-01-31 14:39:39 --> Output Class Initialized
DEBUG - 2008-01-31 14:39:39 --> Final output sent to browser
DEBUG - 2008-01-31 14:39:39 --> Total execution time: 0.1356
DEBUG - 2008-01-31 14:39:39 --> Cache file is current. Sending it to browser.

3.7 Cache

CodeIgniter menyediakan fasilitas caching untuk menaikan performa saat website diakses. Dengan fasilitas caching ini maka semua halaman yang diakses akan disimpan kedalam cache sehingga saat ada user lain yang mengakses maka akan diberikan halaman hasil caching. Hal ini dapat mempercepat proses loading halaman website dan tentu saja akan mengurangi beban kerja dari web server.

Fasilitas caching termasuk kedalam class library output, dan untuk melakukan caching pada setiap halaman website bisa dengan meletakan fungsi :

$this->output->cache([angka]);

Dimana [angka] adalah jumlah menit sebelum halaman tersebut akan kadaluarsa di refresh ulang. Saat sebuah halaman di akses untuk pertama kali maka cache dari halaman tersebut akan disimpan didalam direktori system/cache/. Pastikan direktori tersebut bisa ditulisi oleh web server. Jika ada pengunjung mengakses halaman yang sama maka akan diambilkan halaman hasil cache tersebut. Jika halaman tersebut sudah expired (umurnya sudah melebihi batas angka yang diberikan) maka hasil cache akan dihapus dan diganti dengan yang baru.

Contoh hasil cache dari halaman Contoh Penggunaan views :

1201765621TS---><html>
<head>
<title>Contoh Penggunaan Views</title>
<head>
<body>
Helo World</body>
</html>

Angka *1201765621TS* menunjukan waktu halaman cache tersebut disimpan dalam bentuk timestamp.

3.8 Keamanan

Masalah keamanan adalah maslah klasik yang selalu menghantui semua pengguna dan pengembang website. Meskipun CodeIgniter sudah menyediakan beberapa kemampuan untuk melakukan pengamanan tetapi tetap saja aman atau tidaknya suatu website lebih ditentukan oleh webmasternya.

CodeIgniter menyediakan batasan karakter apa saja yang bisa digunakan dalam pembuatan alamat URI yaitu berupa huruf dan angka atau jika menggunakan karakter lain hanya tilde (~), period (.), colon (:), underscore (_) dan dash (-) yang diperbolehkan. CodeIgniter juga tidak memperbolehkan penggunaan metode GET pada input form dan secara default semua input form menggunakan metode POST.

Saat sebuah controller di load maka CodeIgniter akan menghilangkan semua inputan yang menggunakan metode GET yang ditemukan,melakukan filter terhapap input dengan metode POST jika ditemukan, melakukan filter pada COOKIES jika ditemukan, menyediakan XSS filtering serta meyamakan karakter baris baru dengan \n.

Cross Scripting Hack adalah salah satu jenis cracking website dengan memasukan kode-kode jahat kedalam website kita. CodeIgniter memiliki fitur XSS filtering yang akan menyaring semua input kemungkinan adanya kode-kode jahat.

Fungsi untuk melakukan XSS filtering adalah :

$this->input->xss_clean();

Contoh penggunaanya :

$data_bersih = $this->input->xss_clean($data);

Perintah diatas akan menyaring variabel $data dengan XSS filtering dan menyimpan hasilnya kedalam variabel $data_bersih.

Jika ingin me-load XSS filtering secara otomatis setiap website dijalankan maka lakukan edit pada file system/application/config/config.php pada bagian :

$config['global_xss_filtering'] = FALSE;

Ganti kata FALSE dengan TRUE.

CodeIgniter datang dengan tiga fungsi helpers untuk bekerja dengan input metode POST, COOKIES dan SERVER. Kelebihan menggunakan fungsi yang disediakan oleh CodeIgniter daripada secara manual, misalnya dengan $_POST[‘var’] adalah fungsi tersebut secara default sudah melakukan cek apakah data masukan tersebut ada atau tidak. Sehingga kita tidak perlu membuat semacam logika pengecekan sendiri.

Misalnya saja, sebagai contoh, kita memiliki sebuah form dengan metode POST, didalam form terdapat field pengisian data dengan nama var. Secara manual pada saat input dimasukan maka saya biasa membuat logika kondisi dengan if untuk melakukan cek apakah variabel var ada atau tidak.

```
if ( ! isset($_POST['var'])) {
    $something = FALSE;
} else {
    $something = $_POST['var'];
}
```

Dengan fungsi yang sudah disediakan oleh CodeIgniter maka cukup dengan satu baris kode :

```
$this->input->post('var');
```

Kode tersebut sudah mengerjakan apa yang kita butuhkan. Kita bahkan bisa melakukan XSS filtering secara langsung tanpa perlu menambah baris kode. Cukup beri nilai TRUE pada variabel kedua, sehingga baris kode tersebut menjadi :

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

Seperti yang sudah disebutkan sebelumnya, CodeIgniter juga menyediakan fungsi untuk mengambil COOKIES dan informasi dalam variabel SERVER. Untuk mengambil COOKIES kita bisa menggunakan :

```
$this->input->cookie('nama_cookie');
```

Dan untuk mengambil variabel SERVER kita bisa menggunakan :

$this->input->server('var');

Baik $this->input->cookie() dan $this->input->server() semuanya bisa menggunakan XSS filtering dengan cara yang sama dengan $this->input->post().

CodeIgniter juga memiliki fungsi untuk menampilkan ip address dari pengunjung, hal ini berfungsi jika kita ingin menyimpan data ip pengunjung yang mengunjungi website kita misalnya. Fungsi tersebut adalah :

$this->input->ip_address();

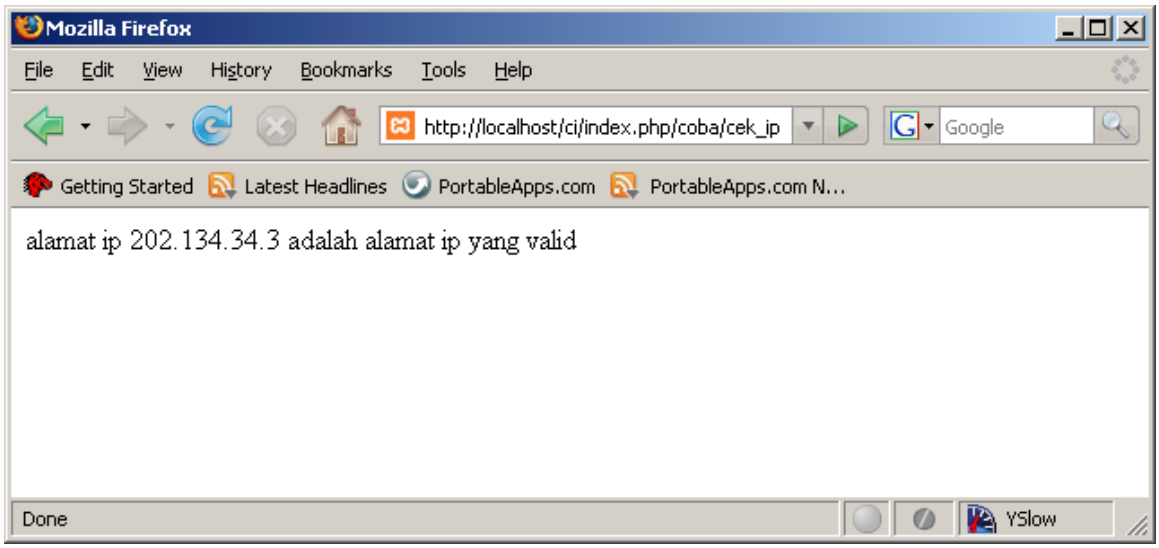
CodeIgniter juga memiliki fungsi untuk melakukan cek apakah alamat suatu ip valid atau tidak.

$this->input->valid_ip([alamat_ip]);

Untuk contoh penggunaan buat sebuah fungsi baru dengan nama cek_ip pada controller coba dengan isi sebagai berikut :

```
function cek_ip() {
  $ip = '202.134.34.3;
   if($this->input->valid_ip($ip)) {
    echo "alamat ip ".$ip." adalah alamat ip yang valid";
   } else {
    echo "alamat ip ".$ip." adalah alamat ip yang tidak valid";
   }
```

Kemudian buka alamat http://localhost/ci/index.php/coba/cek_ip



Gambar : Hasil fungsi cek_ip

3.9 Membuat Beberapa Website Dengan Satu CodeIgniter

Secara default aplikasi yang kita buat disimpan didalam direktori controllers, views dan models didalam direktori application. Sebenarnya kita bisa saja mengganti nama folder application tersebut dengan lain, syaratnya kita harus melakukan perubahan variabel $application_folder pada file index.php. Misalnya saja kita mengganti nama direktori application menjadi aplikasi maka nilai variabel $application_folder harus diganti menjadi “aplikasi”.

$application_folder =“aplikasi”;

Kita juga bisa mengganti nama direktori system dengan nama lain, kemudian rubah juga nilai variabel $system_folder pada file index.php. Misalnya saja direktori system kita ganti menjadi website maka nilai variabel $system_folder harus kita ganti menjadi “website”.

$system_folder = “website”;

CodeIgniter juga mengijinkan kita membuat beberapa website hanya dengan satu instalasi sistem inti CodeIgniter. Syaratnya setiap website harus memiliki index.php sendiri dan memiliki direktori application sendiri. Karena file index.php hanya diijinkan ada satu pada satu direktori maka untuk masing-masing website yang akan dibangun file index.php dapat diganti menjadi nama file lain.

Contoh :
Buka file index.php kemudian save as menjadi index1.php dan index2.php. Pada file index1.php rubah nilai variabel $application_folder menjadi aplikasi1.

$applicaition_folder = “aplikasi1”;

Sedang pada file index2.php rubah nilai variabel $application_folder menjadi aplikasi2.

$applicaition_folder = “aplikasi2”;

Kemudian lakukan konfigurasi seperlunya. Sesuai konfigurasi yang akan digunakan pada masing-masing website. Misalnya saja basis data, routing dan lain-lain. Pada contoh ini semua konfigurasi disamakan dengan contoh-contoh awal.

Kemudian buat sebuah controller untuk masing-masing website. Untub website pertama kita beri nama web1 dan untuk website kedua kita beri nama web2.

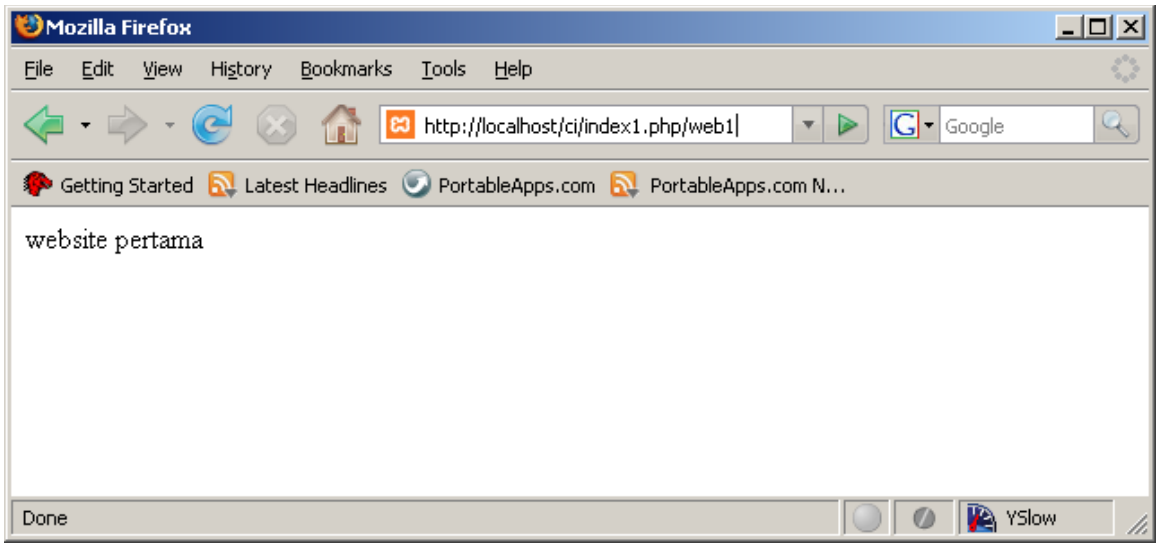
Untuk controller web1 buat file dengan nama web1.php dan simpan didalam direktori system/aplikasi1/controllers.

```
<?php
 class Web1 extends Controller {
  function index() {
   echo "website pertama";
  }
}
?>
```

Untuk controller web2 buat file dengan nama web2.php dan simpan didalam direktori system/aplikasi2/controllers.

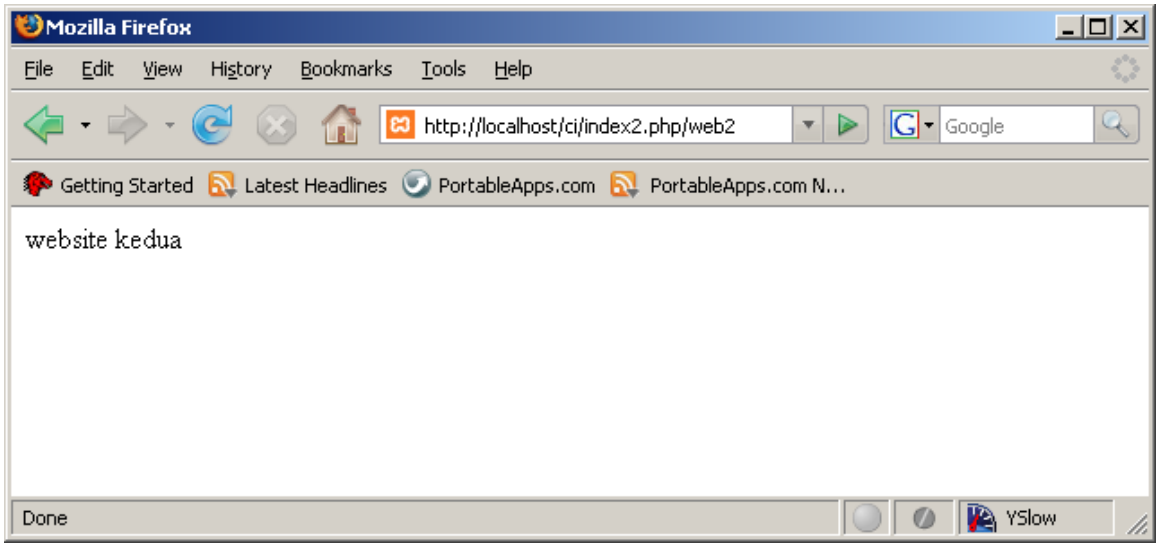
```
<?php
 class Web2 extends Controller {
  function index() {
   echo "website kedua ";
  }
}
?>
```

Kemudian untuk website pertama bisa dibuka dengan membuka alamat http://localhost/ci/index1.php/web1



Gambar : Website pertama

Sedangkan untuk website pertama bisa dibuka dengan membuka alamat http://localhost/ci/index2.php/web2



Gambar : Website kedua

Seakarang kita memiliki 2 buah website tetapi hanya menggunakan satu sistem inti CodeIgniter.

Bab 4 BASIS DATA

Basis data merupakan salah satu hal yang penting dalam website dinamis. Didalam basis data inilah semua data website disimpan. CodeIgniter mendukung beberapa aplikasi basis data, yaitu MySQL (4.1+), MySQLi, MS SQL, Postgre, Oracle, SQLite, dan ODBC. Dukungan aplikasi basis data tersebut sangat mungkin akan bertambah pada versi berikutnya.

Dengan dukungan ke berbagai aplikasi basis data tersebut membuat CodeIgniter semakin banyak digunakan oleh para pembuat website dari berbagai platform sistem operasi. Baik Linux, UNIX maupun Windows atau yang lainnya.

CodeIgniter datang dengan library basis data yang sangat lengkap. Mulai dari operasi pada data seperti select, update, delete dan lain-lain dengan berbagai fungsi-fungsi built in serta operasi pada basis data seperti pembuatan basis data, menghapus basis data dan lain-lain dengan dukungan fungsi-fungsi pada library database forge.

Untuk dapat menggunakan library database bisa dilakukan load secara manual pada controller yang ingin menggunakan basis data atau agar library database di-load secara otomatis dan bisa digunakan secata global bsia dengan melakukan perubahan pada file system/application/config/autoload.php.

Untuk me-load library database secara manual gunakan :

$this->load->database();

Jika ingin agar library database di-load secata otomatis buka file system/application/config/autoload.php kemudian tambahkan database pada $config[‘libraries’].

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

Cara untuk melakukan konfigurasi agar CodeIgniter bisa mengakses ke basis data sudah dijelaskan pada bab 2 bagian konfigurasi. Setelah basis data dapat diakses dan sudah diload kita bisa menggunakan *$this->db->[nama_fungsi]* untuk menggunakan fungsi-fungsi untuk melakukan operasi ke basis data.

4.1 Menjalankan Query

Untuk dapat menjalankan query SQL ke basis data kita bisa menggunakan fungsi :

$this->db->query(‘[query_sql]’);

Misalnya saja kita ingin melakukan query select untuk menampilkan semua data pada tabel siswa, maka seperti ini yang kita gunakan :

*$this->db->query(‘SELECT * FROM siswa’);*

Fungsi *$this->db->query()* akan mengembalikan nilai object yang bisa dilakukan operasi lebih lanjut untuk menampilkan data basilnya saat menjalankan query yang sifatnya membaca isi tabel, dan akan mengembalikan nilai boolean TRUE atau FALSE saat menjalankan query yang sifatnya menulis ke tabel seperti perintah insert misalnya.

Jika hanya ingin melakukan cek apakah suatu query berhasil atau tidak bisa menggunakan fungsi :

$this->db->simple_query(‘[query_sql]’);

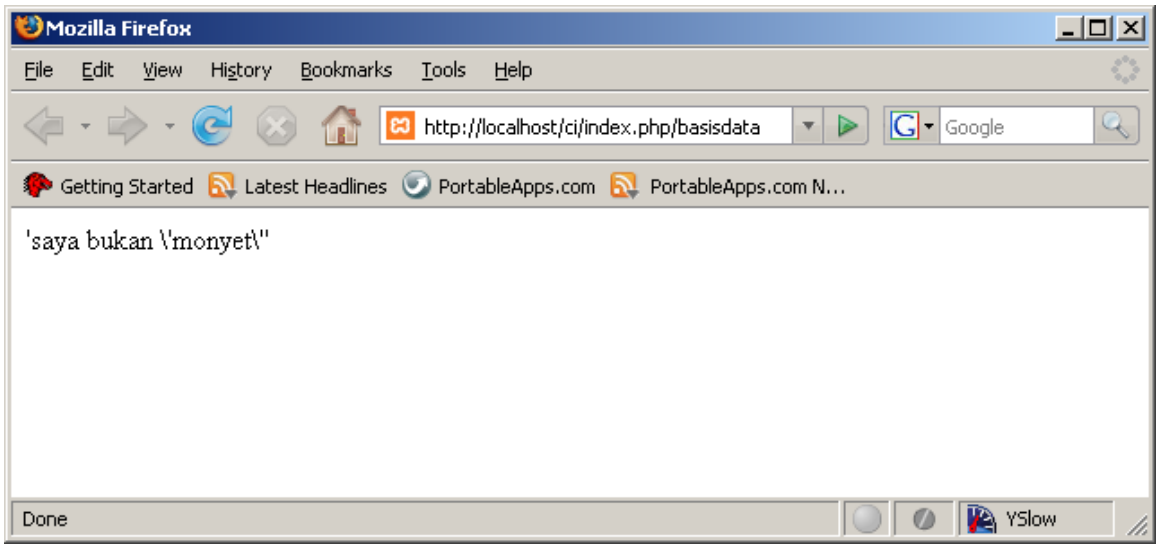
Fungsi ini hanya akan mengembalikan nilai boolean TRUE jika query berhasil atau FALSE jika query gagal dan tidak mengembalikan nilai object sehingga query ini tidak bisa digunakan untuk menampilkan data.

4.2 Melakukan Escaping

Escaping berguna untuk mengamankan basis data dari data-data berbahaya sebelum data dimasukan kedalam basis data, misalnya saja dari karakter-karakter 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->escape([string]);***
   Fungsi ini hanya bisa melakukan escaping pada data yang bertipe string. Fungsi ini akan secata otomatis memberikan single quote (‘) diantara string yang diberikan pada fungsi tersebut.
   Contoh buat sebuah controller dengan nama basisdata, kemudian pada fungsi index() masukan baris seperti berikut :

   echo $this->db->escape(“saya bukan monyet”);



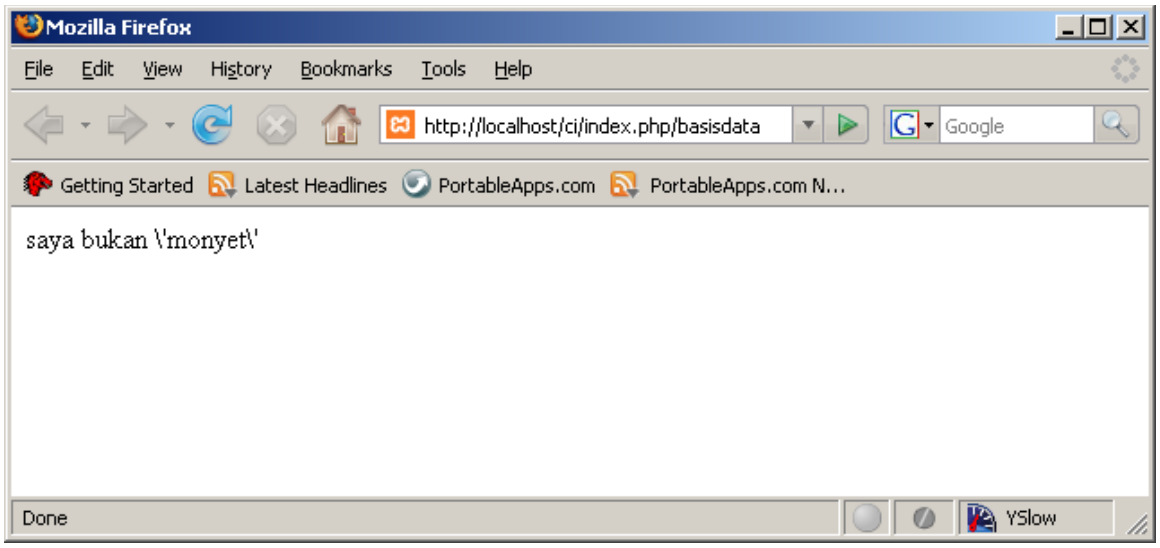
Gambar : Hasil escaping fungsi *$this->db->escape()*

2. ***$this->db->escape_str([string]);***

   Fungsi ini akan meng-escape data apapun yang diberikan tanpa memperdulikan tipe datanya. Fungsi ini tidak secara otomatis memberikan single quote (‘) pada data yang diberikan.

   Sebagai contoh, rubah isi fungsi index() pada controller basis data menjadi menggunakan fungsi $this->db->escape_str().

   echo $this->db->escape_str(“saya bukan monyet”);



Gambar : Hasil fungsi *$this->db->escape_str()*

4.3 Menampilkan Hasil Query

Setelah query dijalankan tentu kemudian yang perlu dilakukan adalah menampilkan hasil dari query jika query merupakan query untuk membaca isi dari suatu tabel. CodeIgniter menyediakan beberapa fungsi di library dan helpers untuk keperluan menampilkan data hasil query tersebut.

Untuk contoh kita akan mengunakan basis data dengan nama sekolah dan tabel siswa yang memiliki 4 buah field, yaitu id, nama, alamat dan kelas. Buat terlebih dahulu basis data dan tabelnya dan lakukan kongfigurasi seperlunya pada file system/application/config/database.php sesuai kondisi yang ada. Setelah itu jangan lupa isi tabel komentar dengan beberapa data. Kita bisa mempergunakan fitur scaffolding yang telah dijelaskan sebelumnya sekaligus untuk lebih mempelajari fitur-fitur CodeIgniter.

result()

Fungsi result() akan mengembalikan nilai dalam bentuk object array. Fungsi ini biasa digabungkan dengan looping foreach untuk menampilkan hasil query yang terdiri dari banyak baris.

Contoh untuk menampilkan semua data dalam tabel siswa :
Buat sebuah model dengan nama db_model dan simpan dengan nama file db_model.php didalam direktori models. Kemudian buat fungsi dengan nama ambil_data().

```
<?php
Class Db_model extends Model {
 function __construct() {
  parent::Model();
 }
 function ambil_data() {
  $data_siswa = $this->db->query("SELECT * FROM siswa");

  return $data_siswa;
 }
}
?>
```

Kemudian buat sebuah controller dengan nama Siswa dan simpan dengan nama siswa.php didalam direktori controllers.

```
<?php
 Class Siswa extends Controller {
  function __construct() {
   parent::Controller();
```

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

 function index() {
  $this->load->model('Db_model','',TRUE);
  $data["data_siswa"] = $this->Db_model->ambil_data();

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

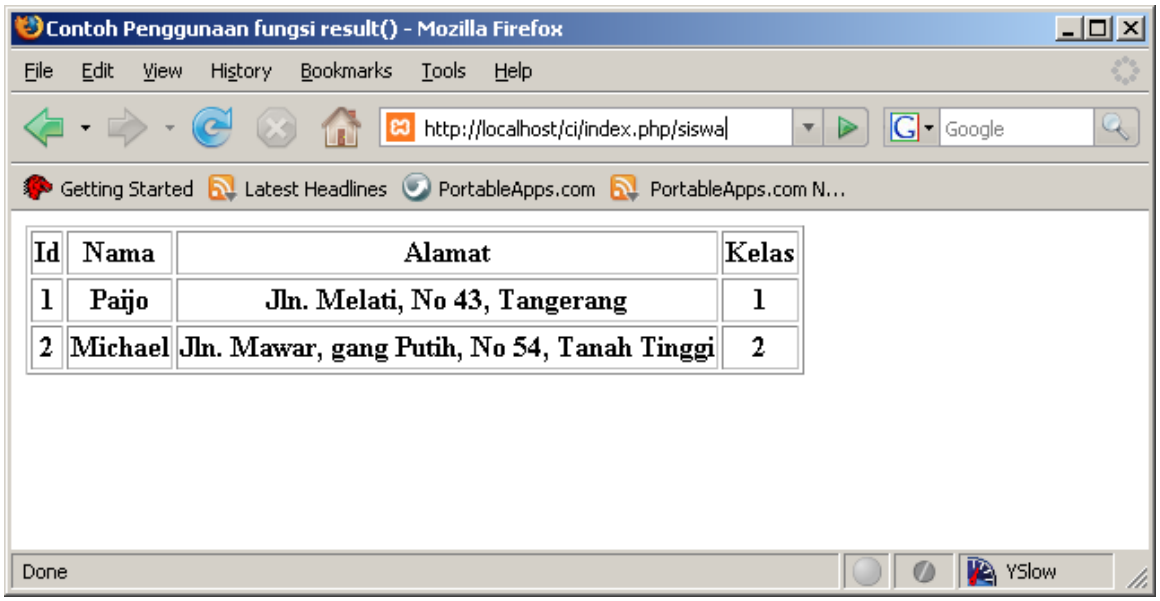
Setelah itu buat file view untuk menampilkan data dengan nama siswa.php dan simpan di direktori views.

```
<html>
<head>
<title>Contoh Penggunaan fungsi result()</title>
<head>
<body>
 <table border="1">
  <tr><th>Id</th><th>Nama</th><th>Alamat</th><th>Kelas</th></tr>
<?php
 foreach($data_siswa->result() as $row) {
  ?>
  <tr><th><?=$row->id?></th><th><?=$row->nama?></th><th><?=$row-
>alamat?></th><th><?=$row->kelas?></th></tr>
  <?php
 }
?>
 </table>
```

</body>

</html>

Buka alamat http://localhost/ci/index.php/siswa untuk melihat hasilnya.



Gambar : Hasil penggunaan fungsi result()

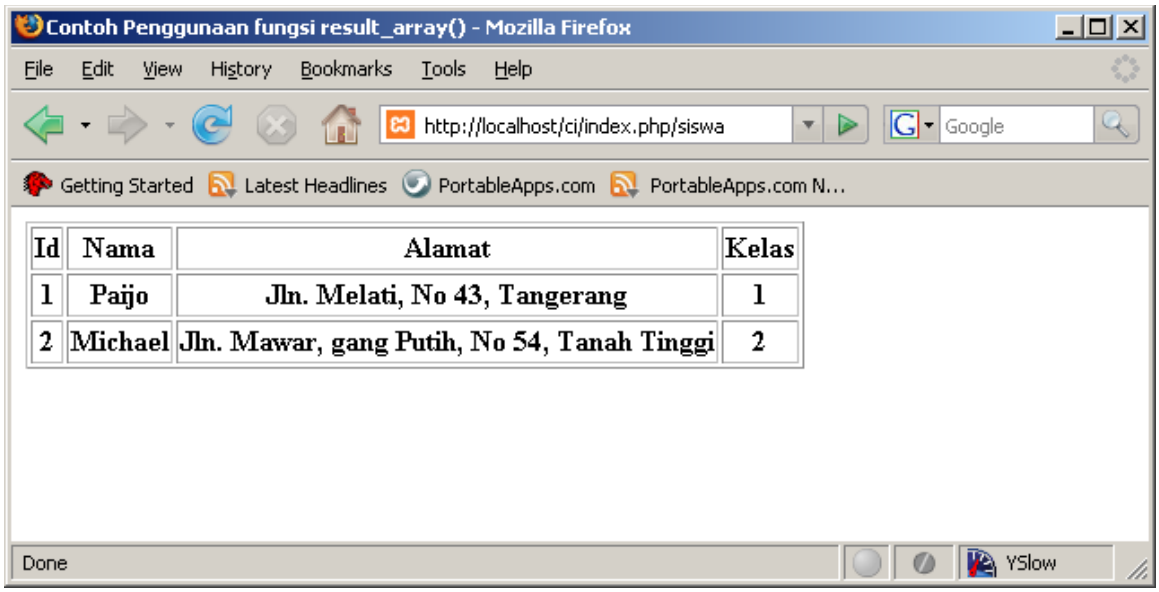
result_array()

Fungsi result_array() akan mengembalikan nilai dalam bentuk array murni dan akan mengasilkan array kosong jika query tidak menghasilkan apa-apa.

Sebagai contoh kita masih menggunakan model dan controller yang sama dengan contoh pada fungsi result, kita hanya melakukan sedikit perubahan pada file views-nya menjadi sebagai berikut :

```
<html>
<head>
<title>Contoh Penggunaan fungsi result_array()</title>
<head>
<body>
 <table border="1">
  <tr><th>Id</th><th>Nama</th><th>Alamat</th><th>Kelas</th></tr>
<?php
 foreach($data_siswa->result_array() as $row) {
  ?>
  <tr><th><?=$row["id"]?></th><th><?=$row["nama"]?></th><th><?=$row["alam
at"]?></th><th><?=$row["kelas"]?></th></tr>
  <?php
 }
?>
 </table>
</body>
</html>
```

Yang membedakan fungsi result() dan fungsi result_array() adalah pada fungsi result() nilai yang dikembalikan adalah berupa array object sehingga untuk menampilkannya menggunakan *$row->[nama_field]* sedangkan pada fungsi result_array() nilai yang dikembalikan dalam array biasa sehingga bisa ditampilkan dengan *$row[[nama_field]].*



Gambar : Hasil penggunaan fungsi result_array()

row()

Fungsi row() digunakan untuk menampilkan hasil query yang hanya terdiri dari satu baris saja. Jika hasil query terdiri dari banyak baris maka fungsi ini hanya akan menampilkan baris pertama saja. Fungsi row() akan mengembalikan nilai dalam bentuk array object.

Sebagai contoh, pada Db_model buat fungsi baru dengan nama ambil_satu().

```
function ambil_satu() {
  $data_siswa = $this->db->query("SELECT * FROM siswa WHERE kelas='1'");

  return $data_siswa;
 }
```

Kemudian buat controller baru dengan nama Satu_siswa dan simpan dengan nama file satu_siswa.php, simpan di direktori controllers.

```
<?php
 Class Satu_siswa extends Controller {
  function __construct() {
   parent::Controller();
   $this->load->database();
  }

  function index() {
   $this->load->model('Db_model','',TRUE);
   $data["data_siswa"] = $this->Db_model->ambil_satu();

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

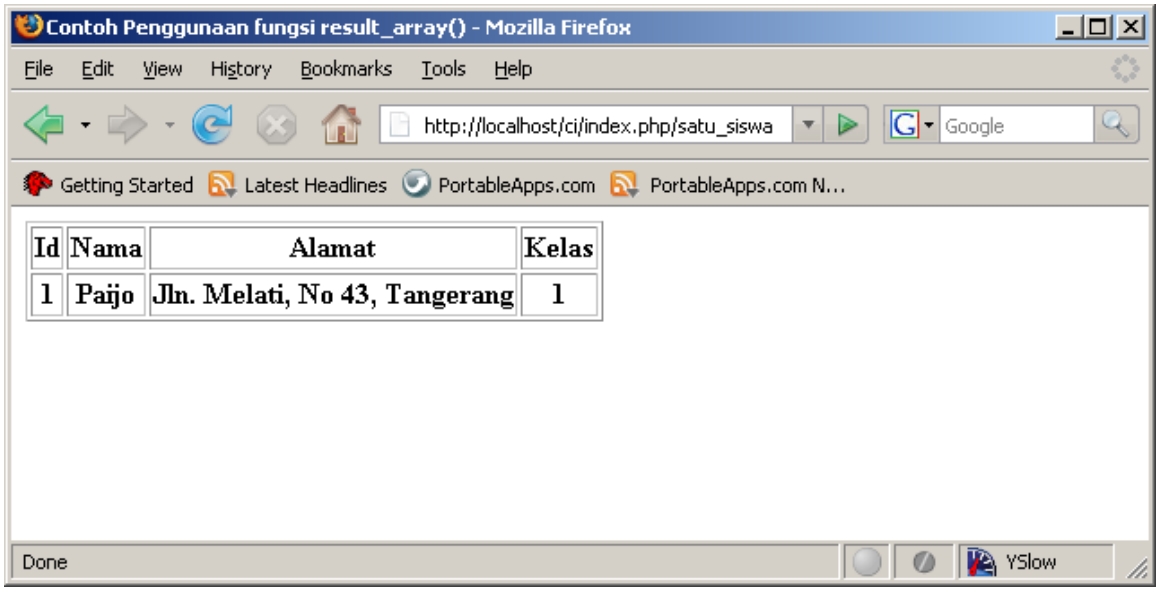
Untuk menampilkan datanya buat view dengan nama file satu_siswa.php dan simpan didalam direktori views.

```
<html>
<head>
<title>Contoh Penggunaan fungsi row()</title>
<head>
<body>
 <table border="1">
  <tr><th>Id</th><th>Nama</th><th>Alamat</th><th>Kelas</th></tr>
<?php
  $row = $data_siswa->row();
  ?>
```

```
  <tr><th><?=$row->id?></th><th><?=$row->nama?></th><th><?=$row->alamat?></th><th><?=$row->kelas?></th></tr>
  <?php

?>
 </table>
</body>
</html>
```

Untuk melihat hasilnya buka alamat http://localhost/ci/index.php/satu_siswa



Gambar : Hasil penggunaan fungsi row()

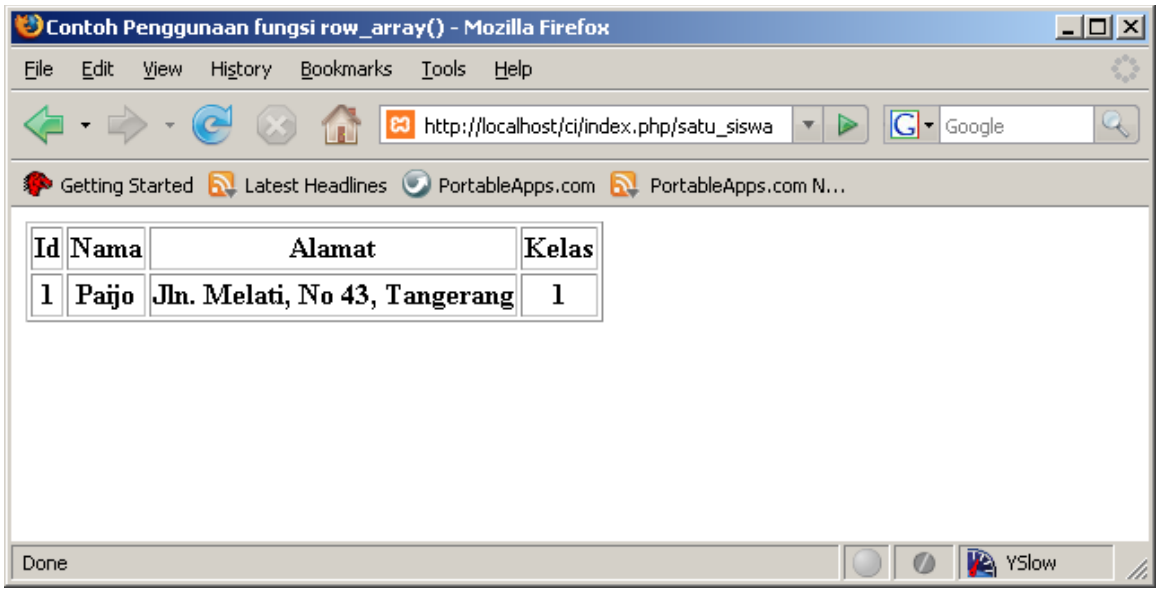
row_array()

Fungsi row_array() sama seperi fungsi row(), bedanya jika fungsi row() menghasilkan array object maka fungsi row_array() mengembalikan nilai array biasa.

Sebagai contoh kita menggunakan model dan controller yang digunakan pada fungsi row(), kita hanya sedikit melakukan modifikasi pada file views-nya.

```
<html>
<head>
<title>Contoh Penggunaan fungsi row_array()</title>
<head>
<body>
 <table border="1">
  <tr><th>Id</th><th>Nama</th><th>Alamat</th><th>Kelas</th></tr>
<?php
  $row = $data_siswa->row_array();
  ?>
  <tr><th><?=$row["id"]?></th><th><?=$row["nama"]?></th><th><?=$row["alam
at"]?></th><th><?=$row["kelas"]?></th></tr>
  <?php

?>
 </table>
</body>
</html>
```



Gambar : Hasil penggunaan fungsi row_array()

Jika misalnya ingin menampilkan beberapa baris dari hasil query yang terdiri dari banyak baris maka bisa dispesifikasikan dengan menyebutkan jumlah baris yang ingin ditampilkan sebagai argumen pada fungsi row_array();

$row = $data_siswa->row_array(2);

4.4 Fungsi-fungsi Lain

Fungsi-fugsi diatas termasuk kedalam library database, selain fungsi-fungsi dalam library database masih ada lagi fungsi-fungsi yang termasuk kedalam helpers database.

num_rows()

Fungsi ini digunakan untuk menampilkan jumlah baris hasil dari query.

*$query = $this->db->query("select * from siswa");*
echo $query->num_rows();

num_fields()

Fungsi ini digunakan untuk menampilkan jumlah field yang ada didalam tabel.

```
$query = $this->db->query("select * from siswa");
echo $query->num_fields();
```

free_result()

Fungsi ini akan membersihkan atau menghapus semua data yang dihasilkan dari query. Fungsi ini berguna saat kita melakukan beberapa query didalam satu file. Letakan fungsi ini untuk menghapus query sebelumnya saat ingin melakukan query baru.

```
$query = $this->db->query("select * from siswa");
foreach($query->result() as $row) {
  // tampilkan hasil query
}

$query->free_result(); // hasil $query sudah tidak bisa dipakai

$query2 = $this->db->query("select * from siswa where id='1'");
$row = $query->row();
// tampilkan hasil query
```

$this->db->insert_id()

Fungsi ini digunakan untuk memasukan nomor ID saat melakukan insert ke tabel.

$this->db->insert_id();

$this->db->affected_rows()

Fungsi ini digunakan untuk menampilkan jumlah tabel yang terpengaruh oleh sebuah query, dapat digunakan untuk query yang melakukan penulisan ke basis data, misalnya insert, update dan lain-lain.

$this->db->affected_rows();

Sebagai contoh buat sebuah fungsi baru didalam model, misalnya saja dengan nama fungsi update(). Kemudian buat sebuah query untuk melakukan update kedalam data di tabel siswa.

```
function update() {
  $update_data = $this->db->query("UPDATE siswa SET nama='Pailul' WHERE id='1'");

  return $update_data;
 }
```

Buat controller dengan nama Affected, simpan dengan nama file affected.php didalam direktori controllers.

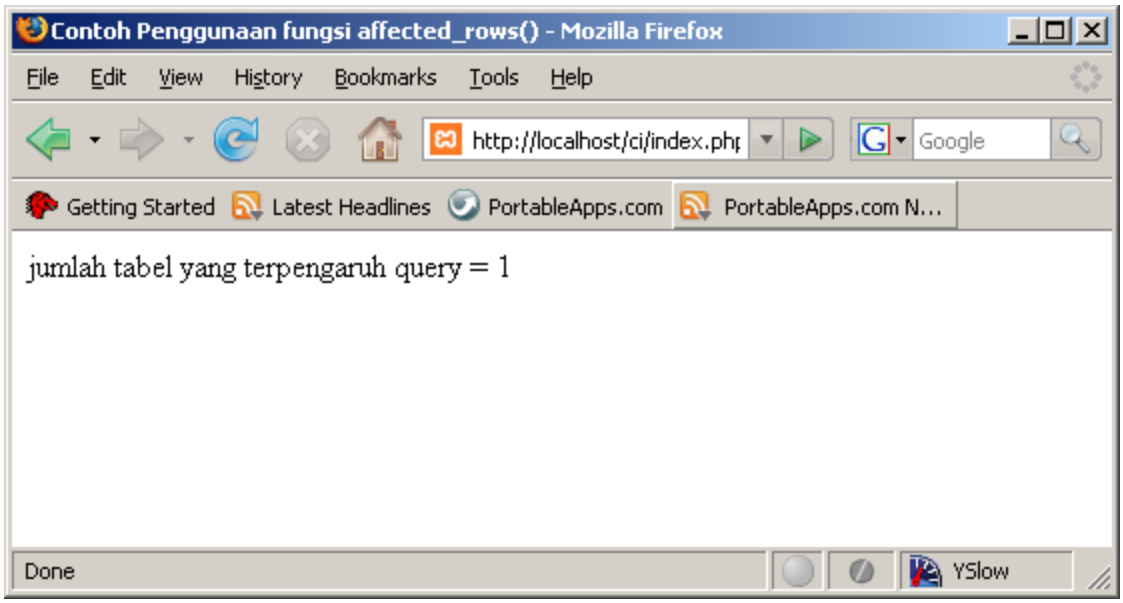
```
<?php
 Class Affected extends Controller {
  function __construct() {
   parent::Controller();
   $this->load->database();
  }
```

```
 function index() {
  $this->load->model('Db_model','',TRUE);
  $data["update_siswa"] = $this->Db_model->update();
  $this->load->view('affected',$data);
 }
}
?>
```

Kemudian buat file views dengan nama file affected.php dan simpan di direktori views.

```
<html>
<head>
<title>Contoh Penggunaan fungsi affected_rows()</title>
<head>
<?php
echo "jumlah tabel yang terpengaruh query = ".$this->db->affected_rows();
?>
 </table>
</body>
</html>
```

Kemudian buka alamat http://localhost/ci/index.php/affected

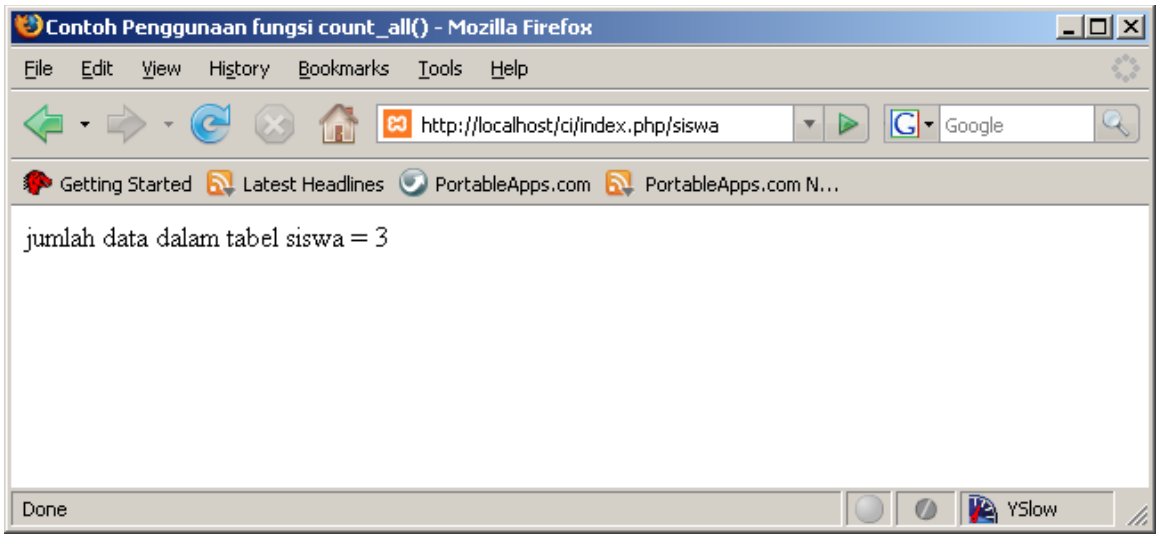


Gambar : Hasil penggunaan fungsi afected_rows()

$this->db->count_all()

Fungsi ini digunakan untuk mencetak jumlah data dalam suatu tabel. Letakan nama tabel sebagai parameter dalam fungsi ini. Misalnya kita ingin menghitung jumlah data dalam tabel siswa maka kodenya:

$this->db->count_all('siswa');



Gambar : Hasil penggunaan fungsi count_all()

$this->db->platform()

Fungsi ini digunakan untuk menampilkan platform server basis data yang kita gunakan. Misalnya mysql, postgresql, dan lain-lain.

$this->db->platform();

$this->db->version()

Fungsi ini akan menampilkan versi dari server basis data yang digunakan.

$this->db->version();

$this->db->last_query()

Fungsi ini akan mengembalikan nilai berupa query terakhir yang dijalankan.

$this->db->last_query()

Jika query terakhir yang dijalankan adalah “SELECT * FROM siswa” maka hasil dari fungsi tersebut adalah “SELECT * FROM siswa”.

$this->db->insert_string()

Fungsi ini digunakan untuk melakukan insert data ke tabel. Dengan menggunakan fungsi ini kita tidak perlu menulis query secara lengkap, cukup cukup definisikan nama field dan data yang akan dimasukan kedalam array.

Data yang ingin dimasukan :

Id	Nama	Alamat	Kelas
4	Ahmad	Jln. Mawar No.5	3

Maka kode untuk memasukan data diatas :

$data = array(‘id’ => ‘4’, ‘nama’ => ‘Ahmad’, ‘alamat’ => ‘Jln. Mawar No.5’, ‘kelas’ => ‘3’) ;
$query = $this->db->insert_string(‘siswa’,$data);

Penggunaan fungsi insert_string() diatas akan sama dengan perintah query SQL sebagai berikut : *INSERT INTO siswa (id,nama,alamat,kelas)*
VALUES(‘4’,’Ahmad’,’ Jln. Mawar No.5’,’3).

$this->db->update_string()

Fungsi ini digunakan untuk melakukan update pada data yang sudah ada.Misalnya saja nama Ahmad ingin diganti dengan Amir Karimudin maka kodenya :
$data_pengganti = array(‘nama’ => ‘Amir Karimudin’);

$where = “ud=’4’”;
$query = $this->db->update_string(‘siswa’,’$data_pengganti’,$where);

Penggunaan fungsi update_string() tersebut akan sama dengan perintah query SQL seperti berikut : *UPDATE siswa SET nama=’Amir Karimudin’ WHERE id=’4’*.

4.5 Active Record Class

Penggunaan active record memungkinkan penggunaan kode yang minimal untuk melakukan operasi-operasi ke basis data seperti insert, update, delete dan lain-lain. Dengan menggunakan active record, diharapkan kode yang kita buat adalah tidak tergantung pada aplikasi server basis data yang digunakan. Apapun aplikasi server basis data yang digunakan maka perpindahan antar server basis data tidak lagi memerlukan perubahan kode secara keseluruhan. Cukup lakukan perubahan nama aplikasi server basis data konfigurasi maka semua sudah selesai.

Active record banyak ditemukan pada berbagai aplikasi framework yang memerlukan akses ke basis data. CodeIgniter juga menyediakan fungsionalitas active record. Penggunaan active record akan menyederhanakan kode-kode untuk melakukan operasi ke basis data.

Fungsi-fungsi Untuk Mengambil Data

$this->db->get()

Fungsi untuk mengambil data dari sebuah tabel. Dengan memasukan nama tabel sebagai parameter pertama pada fungsi ini sama dengan perintah SQL select untuk mengambil data dari nama tabel tersebut.

$this->db->get('siswa') akan sama dengan perintah query SQL "*SELECT * FROM siswa*"

Selain parameter pertama untuk nama tabel, kita juga bisa memberikan parameter kedua sebagi klausa limit dan parameter ketiga sebagai klausa offset. Contoh jika kita ingin mengambil 10 data saja dimulai dari data pertama maka kodenya :

$query = $this->db->get('siswa',10,0);

Kode diatas akan sama dengan perintah query SQL "*SELECT * FROM siswa LIMIT 0,10*". Query tersebut adalah query SQL pada MySQL, aplikasi server basis data yang lain mungkin akan memiliki syntax klausa limit yang berbeda.

$this->db->getwhere()

Fungsi ini seperti fungsi $this->db->get() tetapi mengijinkan penggunaan klausa where. Dengan fungsi ini parameter pertama adalah nama tabel, parameter kedua adalah klausa where, parameter ketiga klausa limit dan parameter keempat adalah offset.

$query = $this->db->gewheret('siswa',array('id' => '3'),10,0);

Kode diatas sama seperti perintah query SQL "*SELECT * FROM siswa WHERE id='3' LIMIT 0,10*";

$this->db->select()

Fungsi ini digunakan untuk menentukan field apa saja yang akan diambil datanya.

$this->db->select('id','nama','kelas');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query SQL "SELECT *id,nama,kelas FROM siswa*". Fungsi *$this->db->select()* hanya untuk menentukan field apa saja yang ingin diambil datanya sedangkan untuk menjalankan querynya tetap harus menggunakan fungsi *$this->db->get()*.

$this->db->from()

Fungsi ini digunakan untuk menentukan nama tabel yang ingin ditampilkan. Jika fungsi ini digunakan maka fungsi *$this->db->get()* dijalankan tanpa memberikan parameter apa-apa.

$this->db->from('siswa');
$this->db->get();

Kode diatas akan sama dengan perintah query SQL "*SELECT * FROM siswa*";

$this->db->where()

Fungsi ini digunakan untuk menentukan nilai untuk klausa where.

$this->db>where('nama','Michael');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama='Michael'*". Kita juga bisa menggunakan fungsi ini beberapa kali untuk menggunakan menerapkan klausa where dengan and.

$this->db>where('nama','Michael');
$this->db->where('kelas','2');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama='Michael' AND kelas='2'*"

Jika memerlukan menggunakan operator seperti sama dengan, lebih kecil atau yang lain kita bisa memasukan operator yang diinginkan kedalam parameter pertama dari fungsi.

$this->db->where('nama !=','Michael');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama != 'Michael'*";

$this->db->or_where()

Jika beberapa fungsi $this->db->where() digunakan bersama maka akan menghasilkan klausa where dengan operator and mana penggunaan fungsi ini akan menghasilkan operator or.

$this->db->where('nama','Michael');
$this->db->or_where('kelas','2');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama='Michael' OR kelas='2'*".

$this->db->like()

Fungsi ini digunakan untuk menghasilkan klausa like yang berguna untuk pencarian data didalam basis data.

$this->db->like('nama','Michael');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama LIKE '%Michael%'*";

Kita juga bisa menggunakan beberapa fungsi $this->db->like() bersama-sama untuk menghasilkan gabungan klausa like yang dihubungkan dengan and.

$this->db->like('nama','Michael');
$this->db->like('alamat','mawar');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama LIKE '%Michael%' AND alamat LIKE '%mawar%'*".

Secara default fungsi ini akan menambahkan wildcard % pada awal dan akhir kata yang diberikan pada parameter kedua. Sebenarnya kita bisa menentukan apakah wildcard akan diletakan diawal dan diakhir, atau hanya diawal dan diakhir saja. Untuk itu kita bisa menambahkan parameter ketiga dengan nilai *before* maka wildcard hanya akan diletakan diawal, *after* maka wildcard hanya akan diletakan diakhir dan *both* maka wildcard akan diletakan diawal dan diakhir.

$this->db->like('nama','Michael','before');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama LIKE '%Michael'*";

```
$this->db->like('nama','Michael','after');
$this->db->get('siswa');
```

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama LIKE 'Michael%'*";

Selain menggunakan fungsi $this->db->like() beberapa kali, kita juga bisa menyederhanakannya menggunakan array untuk membuat gabungan beberapa klausa like dengan and.

```
$aray = array('nama' => 'Michael', 'alamat' => 'Mawar');
$this->db->like($array);
$this->db->get('siswa');
```

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama LIKE '%Michael%' AND alamat LIKE '%Mawar%'*".

$this->db->or_like()

Fungsi ini akan menggabungkan beberapa klausa like dengan or.

```
$this->db->like('nama','Michael');
$this->db->or_like('alamat','mawar');
$this->db->get('siswa');
```

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama LIKE '%Michael%' OR alamat LIKE '%Mawar%'*".

$this->db->not_like()

Fungsi ini menghasilkan klausa not like.

$this->db->not_like('nama','Michael');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa WHERE nama NOT LIKE '%Michael%'*".

$this->db->groupby()

Fungsi ini akan menghasilkan klausa GROUP BY.

$this->db->groupby('id');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa GROUP BY 'id'*".

$this->db->order_by()

Fungsi ini akan menghasilkan klausa ORDER BY.

$this->db->order_by('id');
$this->db->get('siswa');

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa ORDER BY 'id'*".

$this->db->limit()

Fungsi ini akan menghasilkan klausa LIMIT.

```
$this->db->limit('5');
$this->db->order_by('id');
$this->db->get('siswa');
```

Kode diatas akan sama dengan perintah query sql "*SELECT * FROM siswa OR DER BY 'id' LIMIT 5*".

Kita juga bisa memberikan parameter kedua sebagai offset.

```
$this->db->limit('5','0');
$this->db->order_by('id');
$this->db->get('siswa');
```

Maka kode diatas akan sama dengan perintah query sql sql "*SELECT * FROM siswa OR DER BY 'id' LIMIT 0,5*".

Sebenarnya masih banyak fungsi-fungsi lain untuk mengambil data. Tetapi fungsi-fungsi diatas sudah cukup untuk membuat query-query standar sql yang diperlukan saat ingin mengambil data dari basis data.

Fungsi Untuk Input Data

Fungsi-fungsi untuk pembuatan query sql untuk keperluan input data tidaklah sebanyak fungsi-fungsi untuk mengambil data. Karena dalam input data hanya memerlukan satu buah perintah sql yaitu INSERT.

$this->db->insert()

Fungsi ini akan menghasilkan peritah query sql untuk melakukan insert data berdasarkan data yang kita masukan sebagai parameter. Kita bisa memasukan data-data tersebut kedalam sebuah array, kemudian array tersebut dijadikan parameter untuk fungsi $this->db->insert(). Parameter pertama berisi nama tabel sedangkan parameter kedua adalah array data yang ingin diinput.

$input = array('nama' => 'Arya', 'alamat' => 'Jl. Melati No. 30', 'kelas' => '1');
$this->db->insert('siswa',$input);

Kode diatas akan sama dengan perintah query sql "*INSERT INTO siswa(nama, alamat, kelas) VALUES('Arya', Jl. Melati No. 30','1')".*

$this->db->set()

Fungsi ini bisa digunakan untuk men-set nilai yang akan dimasukan. Bisa digunakan untuk INSERT maupun UPDATE.

$this->db->set('nama','Arya');
$this->db->insert('siswa');

Kode diatas akan sama dengan perintah query sql "*INSERT INTO siswa(nama) VALUES('Arya')*".

Kita juga bisa menggunakan array jika ingin men-set beberapa nilai sekaligus.

$this->db->set('nama','Arya');
$this->db->set('kelas',2');
$this->db->insert('siswa');

Kode diatas akan sama dengan perintah query sql "*INSERT INTO siswa(nama,kelas) VALUES('Arya,'2')*".

Fungsi Untuk Update Data

$this->db->update()

Fungsi ini akan menghasilkan perintah query sql untuk melakukan update pada data di dalam basis data sesuai dengan data yang kita masukan.

Kita bisa menggunakan array untuk memberi nilai pada fungsi ini.

```
$this->db->where('id','3');
$array = array('nama' => 'Arya', 'kelas' => '2');
$this->db->update('siswa',$array);
```

Kode diatas akan sama dengan perintah query sql "*UPDATE siswa SET nama='Arya',kelas='2' WHERE id='3'*". Kita perlu menggunakan fungsi $this->db->where() untuk membuat klausa where untuk menentukan data mana yang akan di update.

Selain menggunakan array kita juga bisa menggunakan object class untuk memberi nilai pada fungsi.

Pertama buat sebuah class, misal dengan nama Data

```
Class Data {
  var $nama = 'Arya';
  var $kelas = '2';
}
```

$data = new Data();
$this->db->where('id','3');
$this->db->update('siswa','$data');

Kode diatas akan sama dengan perintah query sql "*UPDATE siswa SET nama='Arya',kelas='2' WHERE id='3'*".

Fungsi Untuk Menghapus Data

$this->db->delete()

Fungsi ini akan menghasilkan perintah query sql untuk menghapus data yang sudah tidak diinginkan. Parameter pertama adalah nama tabel dan parameter kedua adalah nilai untuk menentukan data mana yang akan dihapus.

$this->db->delete('siswa',array('id' => '2'));

Kode diatas akan sama dengan perintah query sql "*DELETE FROM siswa WHERE id='2'*". Kita juga bisa memanfaatkan fungsi $this->db->where() untuk membuat klausa where.

$this->db->where('id','2');
$this->db->delete('siswa');

Kode diatas akan sama dengan perintah query sql "*DELETE FROM siswa WHERE id='2'*".

$this->db->empty_table()

Fungsi ini bisa digunakan jika kita ingin menghapus seluruh data yang ada didalam sebuah tabel.

$this->db->empty_table('siswa');

Kode diatas akan sama dengan perintah query sql "*DELETE FROM siswa*" yang akan menghapus semua data yang ada didalam tabel siswa.

Method Chaining

Jika menggunakan PHP versi 5 maka kita bisa menggunakan apa yang disebut sebagai method chaining. Dengan menggunakan method chaining kita bisa menyederhanakan kode PHP dengan menggabungkan beberapa fungsi active record. Contoh :

$this->db->select('nama')->from('siswa')->where('id',3);

Kode diatas akan sama dengan perintah query SQL "*SELECT nama FROM siswa WHERE id='3'*".

Referensi

1. User Guide CodeIgniter, http://www.codeigniter.com/user_guide/
2. Wiki CodeIgniter, http://www.codeigniter.com/wiki/