Pengantar

CodeIgniter adalah sebuah framework pengembangan aplikasi, toolkit untuk membangun situs web menggunakan PHP. Tujuannya adalah untuk memungkinkan pengembangan proyek web lebih cepat daripada menulis kode dari awal. Framework ini menyediakan satu set pustaka (*library*) yang kaya untuk tugas-tugas yang biasanya dibutuhkan dalam pekerjaan mengembangkan web. Antarmuka yang dimiliki sederhana dan terstruktur secara logis untuk mengakses pustaka. CodeIgniter memungkinkan pengembang web fokus secara kreatif pada proyek dengan meminimalkan jumlah kode yang diperlukan untuk suatu tugas.

Kelebihan yang dimiliki framework CodeIgniter:

- Memiliki kinerja yang luar biasa.
- Kompatibel dengan akun standar hosting yang menjalankan berbagai versi PHP dan konfigurasi.
- Hampir tidak memerlukan konfigurasi.
- Tidak memerlukan penggunaan baris-baris perintah.
- Sebuah kerangka kerja yang tidak mengharuskan patuh pada aturan pengkodean yang ketat.
- Tidak perlu belajar bahasa template untuk menggunakannya (meskipun secara opsional tersedia parser template jika diinginkan).
- Tidak kompleks dan mendukung solusi sederhana.
- Dokumentasi lengkap dan jelas.

Persyaratan Sistem

Penggunaan CodeIgniter memerlukan PHP versi 5.1.6 atau yang lebih baru. Database yang didukung adalah MySQL (4.1 keatas), MySQLi, MS SQL, Postgres, Oracle, SQLite, dan ODBC

Cara Menginstall

- Unduh paket CodeIgniter (versi terbaru 2.1.4)
- Unzip paket
- Upload (pindahkan) folder CodeIgniter dan file-filenya ke server. Secara normal file index.php ada di *root*
- Buka file *application/config/config.php* dengan sebuah teks editor dan tetapkan basis URL Anda. Jika bermaksud menggunakan *encrypton* atau *session*, tetapkan kunci enkripsi.
- Jika bermaksud menggunakan database, buka file application/config/database.php dengan teks editor dan atur setingan database.

Jika ingin meningkatkan keamanan dengan menyembunyikan lokasi file-file CodeIgniter, Anda dapat mengganti nama folder **system** dan **application** lalu buka file utama index.php dan tentukan variabel *$system_folder* dan *$application_folder* di baris atas file dengan nama yang sudah dipilih.

Agar lebih aman lagi, kedua folder diatas sebaiknya diletakkan dibawah folder root web sehingga tidak dapat diakses langsung melalui browser. Secara default, file .htaccess disertakan kedalam setiap folder untuk membantu mencegah akses langsung, namun akan lebih baik jika file-file ini dihapus dari akses publik jika konfigurasi server web mengalami perubahan.

Setelah menghapus file-file .htaccess, buka file utama index.php dan atur variabel *$system_folder* dan *$application_folder,* sebaiknya yang *full path*, misalnya 'www/myuser/sistem'.

Fitur

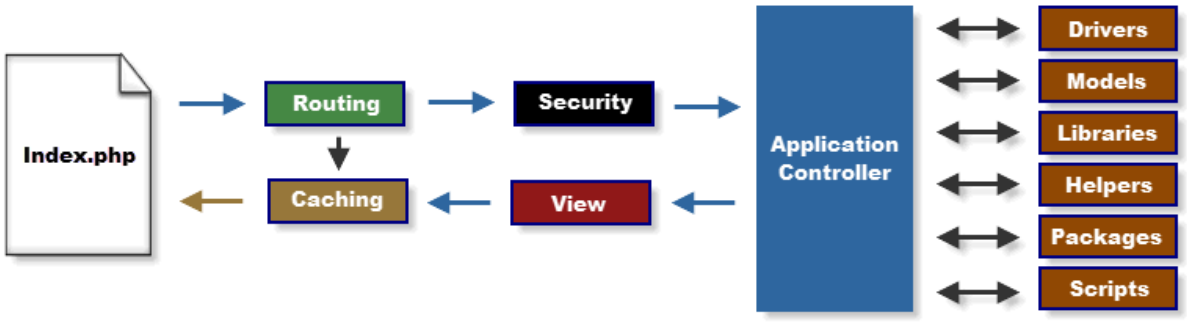
Fitur-fitur dalam CodeIgniter adalah:

- Sistem berbasis Model-View-Controller (MVC)
- Sangat ringan
- Fitur class database lengkap yang mendukung beberapa platform
- Mendukung Active Record Database
- Validasi Data dan Form
- Security and XSS Filtering
- Manajemen session
- Class Email Sending. Mendukung Attachments, HTML/Text email, multiple protocols (sendmail, SMTP, and Mail)
- Image Manipulation Library (cropping, resizing, rotating, dan lain-lain). Mendukung GD, ImageMagick, and NetPBM
- File Uploading Class
- FTP Class
- Localization
- Pagination
- Data Encryption
- Benchmarking
- Full Page Caching
- Error Logging
- Application Profiling
- Calendaring Class
- User Agent Class

- Zip Encoding Class
- Template Engine Class
- Trackback Class
- XML-RPC Library
- Unit Testing Class
- URL yang ramah terhadap Search-engine
- URI Routing yang fleksibel
- Dukungan terhadap Hooks and Class Extensions
- Large library of "helper" functions

Cara Kerja Framework

Gambar dibawah ini mengilustrasikan cara kerja CodeIgniter, bagaimana data mengalir melalui sistem:



1. File index.php bertindak sebagai *front controller*, menginisialisasi basis sumber daya yang dibutuhkan untuk menjalankan CodeIgniter
2. Router memeriksa request HTTP untuk menentukan apa yang harus dilakukan terhadap request tersebut
3. Jika file cache ada, file akan dikirim langsung ke browser, memotong alur normal sistem
4. Sebelum *application controller* dimuat, request HTTP dan setiap data yang dikirim user akan disaring untuk keamanan
5. *Controller* memuat model, pustaka inti, *helper*, dan sumber daya lain yang diperlukan untuk memproses request
6. View akhir dirender lalu dikirim ke browser untuk dilihat. Jika *chacing* aktif, view akan dicache dulu sehingga untuk request yang sama berikutnya dapat terlayani dengan *cache*.

Model-View-Controller

CodeIgniter berdasarkan pola pengembangan Model-View-Controller (MVC), suatu pendekatan perangkat lunak yang memisahkan logika aplikasi dengan presentasinya. Dalam penerapannya, pola ini akan memungkinkan halaman web berisi script minimal karena kode-kode presentasi dipisahkan dari script PHP.

- **Model** mewakili struktur data aplikasi. Biasanya class-class model akan berisi fungsi-fungsi yang membantu pemanggilan, penyisipan, dan perubahan informasi di database.
- **View** mewakili informasi yang disajikan ke user. Sebuah view secara normal berupa halaman web, tetapi dalam CodeIgniter, sebuah view dapat pula berupa bagian dari halaman seperti header atau footer, dapat berupa halaman RSS, atau segala jenis halaman.
- **Controller** berperan sebagai mediator antara model, view, dan sumber daya lainnya yang diperlukan untuk memproses request HTTP dan menghasilkan sebuah halaman web.

CodeIgniter memiliki pendekatan yang cukup longgar untuk MVC sejak Model tidak diperlukan. Jika Anda tidak perlu tambahan pemisahan, atau berpikir mempertahankan model memerlukan kompleksitas lebih dari yang Anda inginkan, Anda dapat mengabaikan mereka dan membangun aplikasi Anda secara minimal menggunakan Controller dan Views. CodeIgniter juga memungkinkan Anda untuk memasukkan script Anda sendiri, atau bahkan mengembangkan pustaka inti untuk sistem, memungkinkan Anda untuk bekerja dengan cara yang paling masuk akal bagi Anda.

Tujuan Desain dan Arsitektur

Tujuan pengembangan CodeIgniter adalah kinerja maksimum, kemampuan, dan fleksibilitas dalam paket yang sekecil dan seringan mungkin. Untuk memenuhi tujuan ini kami berkomitmen untuk benchmarking, dan menyederhanakan setiap langkah dari proses pembangunan.

Dari sudut pandang teknis dan arsitektur, CodeIgniter diciptakan dengan tujuan sebagai berikut:

- **Dynamic instantiation**. Pada CodeIgniter, komponen dimuat dan rutin dieksekusi hanya jika diminta, bukan global. Tidak ada asumsi yang dibuat oleh sistem tentang apa yang mungkin diperlukan di luar sumber daya inti minimal, sehingga secara default sistem ini sangat ringan.
- **Loose Coupling**. Coupling adalah sejauh mana komponen sistem saling mengandalkan. Semakin sedikit komponen saling bergantung satu sama lain, sistem semakin *reusable* dan fleksibel. CodeIgniter adalah sistem yang sangat longgar komponen-komponennya.
- **Component Singularity**. Singularity adalah sejauh mana komponen memiliki tujuan yang difokuskan secara sempit. Pada CodeIgniter, setiap kelas dan fungsinya sangat otonom untuk memungkinkan manfaat maksimal.

CodeIgniter berkarakter instansiasi dinamis, sistem yang longgar ditambah dengan singularitas komponen yang tinggi. Framework ini diusahakan sederhana, fleksibel, dan berkinerja tinggi dalam paket yang kecil.

Tutorial

Pendahuluan

Tutorial ini ditujukan untuk memperkenalkan pada Anda framework CodeIgniter dan prinsip-prinsip dasar arsitektur MVC. Hal Ini akan menunjukkan kepada Anda bagaimana sebuah aplikasi CodeIgniter dasar dibangun dalam langkah-demi-langkah.

Dalam tutorial ini, Anda akan membuat sebuah aplikasi berita dasar. Anda akan mulai dengan menulis kode yang dapat memuat halaman statis. Selanjutnya, Anda akan membuat bagian aplikasi yang membaca item berita dari database. Akhirnya, Anda akan menambahkan form untuk membuat item berita dalam database.

Tutorial ini terutama akan fokus pada:

- Dasar-dasar Model-View-Controller
- Routing dasar
- validasi form
- Melakukan query database dasar menggunakan "Active Record"

Seluruh tutorial dibagi atas beberapa halaman, masing-masing menjelaskan bagian kecil fungsi dari framework CodeIgniter. Tutorial terbagi beberapa bagian:

1. **Pendahuluan**, memberikan gambaran tentang apa yang diharapkan.
2. **Halaman statis**, mengajarkan dasar-dasar controller, view dan routing.
3. **Bagian berita**, yang akan mulai menggunakan model dan akan melakukan beberapa dasar operasi database.
4. **Buat item berita**, yang akan memperkenalkan operasi database lebih lanjut dan validasi form.

Halaman Statis

Hal pertama yang perlu dilakukan adalah membentuk *controller* untuk menangani halaman statis. *Controller*, secara sederhana, sebuah kelas yang membantu delegasi pekerjaan. *Controller* ini bertindak sebagai perekat aplikasi web yang dibuat dengan CodeIgniter.

Misalnya, ketika suatu panggilan dibuat seperti:

http://example.com/news/latest/10

Kita dapat bayangkan bahwa ada sebuah *controller* bernama *news*. Metode yang dipanggil pada *controller news* adalah *latest*. Tugas metode ini mungkin berupa mengambil 10 item berita, dan merendernya (menterjemahkan) ke halaman. Pola URL yang sering ada dalam MVC adalah:

http://example.com/[controller-class]/[controller-method]/[arguments]

Buat sebuah file di *application/controllers/pages.php* dengan isi kode sebagai berikut:

```
<?php
class pages extends CI_Controller {
        public function view($page='home')
        {
        }
}
?>
```

Kode diatas membuat sebuah kelas bernama *pages* dan sebuah metode *view* yang menerima sebuah argumen, yaitu *$page*. Kelas *pages* adalah perluasan dari kelas *CI_Controller*. Hal ini berarti kelas *pages* dapat mengakses metode dan variabel yang didefinisikan dalam kelas *CI_Controller* (*system/core/Controller.php*).

Controller adalah bagian yang akan menjadi pusat setiap request ke aplikasi web. Isitilah teknis dalam CodeIgniter adalah *super object*. Seperti kelas dalam PHP, untuk merujuk kepada metode dari dalam *controller* menggunakan *$this*.

Setelah metode dibuat, langkah selanjutnya adalah membuat beberapa *template* halaman dasar. Disini akan dibuat dua buah *view* (template halaman) yang bertindak sebagai **header** dan **footer**.

Buatlah **header** di *application/views/templates/header.php* dan tulislah kode berikut:

```
<html>
<head>
<title><?php echo $title ?></title>
</head>
<body>
<h1>Tutorial CodeIgniter</h1>
```

Header diatas berisi kode HTML dasar yang akan ditampilkan sebelum *loading* (memuat) *view* utama. Header juga akan mencetak variabel *$title*, yang akan didefinisikan nanti didalam *controller*.
Footer kemudian dibuat di *application/views/templates/footer.php* yang berisi kode:

```
<strong>&copy; 2011</strong>
</body>
</html>
```

Menambahkan Logika Ke Controller

Sebelumnya telah dibuat *controller* dengan metode *view().* Metode ini menerima satu parameter, yaitu nama halaman yang akan dimuat. *Template* halaman statis akan berlokasi di *aplikasi/views/pages/.*

Dalam direktori tersebut, dibuat dua file bernama *home.php* dan *about.php*. File-file tersebut dapat diisi dengan beberapa teks bebas, misalnya teks klasik belajar pemrograman, yaitu "Hello World!", lalu file tersebut disimpan.

Kembali pada metode *view()*. Untuk memuat halaman-halaman diatas, perlu diperiksa dulu apakah halaman yang diminta benar-benar ada. Pemeriksaan ini dilakukan dalam metode *view()*. Maka, kode dalam metode ini seperti berikut:

```
public function view($page="home")
{
        if (!file_exists('application/views/pages/'.$page.'.php'))
        {
                show_404();
        }
        $data['title']=ucfirst($page);      //Membuat huruf pertama kapital
        $this->load->view('templates/header', $data);
        $this->load->view('pages/'.$page, $data);
        $this->load->view('templates/footer', $data);
}
```

Maka, jika halaman ada, akan dimuat, termasuk **header** dan **footer**, dan ditampilkan ke user. Jika halaman tidak ada, maka pesan kesalahan "404 page not found" yang akan tampil.

Baris pertama pada metode ini akan memeriksa apakah halaman ada. Fungsi bawaan PHP *file_exists()* digunakan untuk memeriksa apakah file ada di lokasi yang disebutkan dalam parameter. Metode *show_404()* adalah fungsi terpasang dalam CodeIgniter yang menerjemahkan halaman error default.

Dalam template **header**, variabel *$title* digunakan untuk mengubahsesuaikan judul halaman. Nilai variabel didefinisikan dalam metode ini. Tidak seperti biasanya pemberian nilai yang ditempatkan ke sebuah variabel, nilai judul halaman (*$title*) ini ditempatkan ke elemen array "title" didalam variabel array *$data*.

Hal terakhir yang dilakukan adalah memuat semua *view* dengan urutan yang sesuai (header, page, footer). Parameter kedua dari metode *view()* digunakan untuk melewatkan nilai ke *view*. Masing-masing nilai didalam array *$data* ditempatkan ke sebuah variabel yang bernama sesuai dengan kunci indeksnya. Artinya, nilai dari *$data['title']* didalam *controller* setara dengan *$title* didalam *view.*

Routing

Sampai disini controller sudah berfungsi. Untuk melihat halaman-halaman home dan about menggunakan URL *[url_situs]/index.php/pages/view*. Jadi, untuk melihat halaman about alamatnya *index.php/pages/view/about*. Ketika halaman about ditampilkan pada browser file header.php dan footer.php juga ikut ditampilkan.

Dalam CodeIgniter ada aturan routing yang dapat diubahsesuaikan. Dengan ini, programmer dapat mengendalikan URI ke controller dan metode yang manapun, dan tidak terikat kepada aturan normal pengaksesan:

http://example.com/[controller-class]/[controller-method]/[arguments]

Untuk melakukan hal ini, buka file pengatur routing yang terletak di *application/config/routes.php* dan tambahkan dua baris dibawah ini. Hapus semua kode lainnya yang mengatur sembarang elemen dalam array $route.

```
$route['default_controller']='pages/view';
 $route['(:any)']='pages/view/$1';
```

Simpan perubahan ini. CodeIgniter akan membaca aturan routing pada file *routes.php* dari atas ke bawah dan mengarahkan request ke controller default yang baru ('pages/view'). Aturan disebelah kiri tanda sama dengan adalah Regular expression yang dipetakan ke sebuah controller dan nama metode yang dipisahkan tanda *slash* (aturan disebelah kanan tanda sama dengan). Jika ada *request*, CodeIgniter akan mencocokkannya dengan aturan routing dan memanggil controller dan metode yang tepat sesuai aturan. Kadang-kadang request juga disertai parameter.

Aturan kedua (baris kedua) akan digunakan untuk request yang disertai parameter dan akan melewatkan parameter ke metode view dari kelas *pages*. Untuk mencoba aturan routing yang baru tersebut, buka halaman about dengan index.php/about. Seharusnya halaman about tampil dengan benar.