Tutorial Awal memakai Zend Framework

Oleh Rob Allen, www.akrabat.com
Diterjemahkan oleh Riki Risnandar, www.risnandar.com
Revisi Dokumen 1.2.3

Tutorial ini bertujuan untuk memberikan penjelasan awal untuk menggunakan Zend Framework dimana contoh disini akan membuat suatu aplikasi sederhana yang memakai database.

CATATAN: Tutorial ini telah ditest dengan menggunakan Zend Framework versi 0.6 dan 0.7. Kemungkinan besar dapat dijalankan juga pada versi yang terakhir, dimana API untuk MVC terus diperbaharui. Tutorial ini tidak akan berjalan pada versi dibawah 0.6.

Arsitektur Model-View-Controller

Pada pemrograman PHP tradisional, pembuatan suatu aplikasi adalah seperti berikut ini :

```
<?php
include "common-libs.php";
include "config.php";
mysql_connect($hostname, $username, $password);
mysql_select_db($database);
?>

<?php include "header.php"; ?>
<h1>Home Page</h1>

<?php
$sql = "SELECT * FROM news";
$result = mysql_query($sql);
?>
<table>
<?php
while ($row = mysql_fetch_assoc($result)) {
?>
<tr>
    <td><?php echo $row['date_created']; ?></td>
    <td><?php echo $row['title']; ?></td>
</tr>
<?php
}
?>
</table>
<?php include "footer.php"; ?>
```

Sejalan dengan perkembangan waktu, aplikasi seperti diatas menjadi susah untuk dipelihara dan klien terus meminta tambahan – tambahan lain yang mengharuskan kita merubah banyak kode dilokasi file yang berbeda - beda.

Salah satu metode untuk mempermudah pemeliharaan suatu aplikasi adalah dengan cara memisahkan kode-kode menjadi tiga bagian yang terpisah (and biasanya dipecah menjadi beberapa file) :

Model	Bagian dari aplikasi yang berhubungan dengan data spesifik yang akan ditampilkan. Pada kode diatas, konsepnya adalah modul “news”. Pada model ini, semua transaksi masuk kedalam modul “business” sebagai bagian dari aplikasi yang fungsinya untuk mengambil atau menyimpan ke database.
View	Bagian ini merupakan modul yang berhubungan dengan tampilan untuk

	pengguna. Biasanya berbentuk HTML.
Controller	Modul Controller mengatur dan menggabungkan modul Model dan modul View untuk memastikan bahwa data yang benar tampil pada halaman web.

Zend Framework menggunakan arsitektur Model-View-Controller (MVC). Cara tersebut digunakan untuk memisahkan suatu aplikasi menjadi beberapa bagian sehingga aplikasi tersebut menjadi mudah ketika dibuat dan mudah dalam pemeliharaan.

Kebutuhan

Zend Framework membutuhkan syarat – syarat seperti dibawah ini:

- PHP 5.1.4 (atau versi diatasnya)
- Sebuah web server yang mendukung fungsi mod_rewrite. Tutorial disini menggunakan Apache.

Memperoleh Framework

Zend Framework dapat diperoleh di http://framework.zend.com/download dalam format .zip atau format .tar.gz. Sewaktu tulisan ini dibuat, versi 0.7 adalah versi yang terakhir. Anda harus menggunakan versi 0.7 untuk tutorial ini agar bisa berjalan.

Struktur Direktori

Dalam pembuatan aplikasi, Zend Framework tidak mengharuskan struktur direktori yang baku, tetapi dalam buku manual merekomendasikan struktur direktori yang standar. Direktori tersebut berasumsi bahwa anda mempunyai kontrol penuh terhadap konfigurasi Apache, tetapi untuk mempermudah hidup, kita akan menggunakan beberapa modifkasi.

Mulailah dengan membuat direktori dibawah direktori utama web server yang bernama `zf-tutorial`. Ini berarti bahwa URL untuk mengakses direktori tersebut adalah `http://localhost/zf-tutorial`.

Buatlah beberapa direktori sesuai struktur dibawah ini untuk menyimpan file – file aplikasi :

```
zf-tutorial/
    /application
           /controllers
           /models
           /views
    /library
    /public
           /images
           /scripts
           /styles
```

Seperti yang anda lihat, kita mempunyai direktori yang terpisah untuk penyimpanan file seperti model, view dan controller. Gambar – gambar tambahan, script dan file CSS disimpan pada direktori yang terpisah dibawah direktori publik. File Zend Framework akan disimpan pada direktori library. Jika anda menggunakan library yang lain, anda juga dapat menempatkannya disini.

Ekstrak file arsip yang sudah didownload (dalam contoh disini bernama ZendFramework-0.6.zip) ke dalam direktori sementara. Semua file dalam arsip akan ditempatkan dalam direktori dibawahnya yang bernama `ZendFramework-0.6`. Lalu Copy semua file dalam direktori `ZendFramework-0.6/library` kedalam direktori `zf-tutorial/library`. Direktori `zf-tutorial/library` sekarang akan berisi sub-directory yang bernama `Zend` dan sebuah file yang bernama `Zend.php`.

Bootstrapping

Bagian ini membahas Zend Framework's controller, Zend_Controller didesain untuk mendukung alamat pengaksesan website dengan alamat yang bersih. Untuk mengaktifkannya, semua permintaan halaman diharuskan untuk melalui file yang bernama index.php, yang dikenal sebagai bootstrapper. Fasilitas ini memberikan akses kepada kita untuk menangani semua alamat website dan untuk memastikan bahwa semua variable atau kondisi telah disetting secara benar. Fasilitas ini diaktifkan dengan menggunakan file .htaccess pada direktori root `zf-tutorial` :

zf-tutorial/.htaccess

```
RewriteEngine on
RewriteRule .* index.php

php_flag magic_quotes_gpc off
php_flag register_globals off
```

RewriteRule sangat mudah dipahami dan dapat diartikan "untuk setiap alamat, alihkan ke file index.php".

Disini kita juga akan mengeset beberapa setting di PHP untuk keperluan keamanan dan agar berjalan dengan semestinya. Biasanya pada beberapa konfigurasi yang sudah ada sudah diset secara benar tetapi kita harus memastikannya terlebih dahulu! Perhatikan baik – baik bahwa setting php_flag pada file .htaccess hanya berfungsi jika anda menggunakan modul mod_php pada Apache anda. Jika anda menggunakan CGI/FastCGI, maka anda harus memastikan bahwa settingan pada file php.ini anda sudah benar.

Bagaimanapun juga, permintaan untuk gambar, file JavaScript dan file CSS tidak boleh di alihkan kepada bootstrapper kita (index.php). Dengan menyimpan file – file tersebut didalam direktori `public`, kita dapat merubah sifat redirect tersebut dengan cara membuat versi lain dari file .htaccess yang disimpan pada direktori `zf-tutorial/public`:

zf-tutorial/public/.htaccess

```
RewriteEngine off
```

Sekarang kita akan menambahkan beberapa setting pada file .htaccess untuk memastikan bahwa direktori application dan direktori library tidak dapat diakses dari luar :

zf-tutorial/application/.htaccess

```
deny from all
```

zf-tutorial/library/.htaccess

```
deny from all
```

Ingat bahwa file .htaccess digunakan oleh Apache, konfigurasi directive pada file httpd.conf harus di set ke setting **All**. Ide tersebut yang menggunakan beberapa file .htaccess diambil dari artikel Jayson Minard's "Blueprint for PHP Applications: Bootstrapping (Part 2) ". Saya merekomendasikan anda untuk membaca semua seri dari artikel tersebut.

File Bootstrap : index.php

`zf-tutorial/index.php` adalah file bootstrap kita dan kita akan memulai dengan mengisi file tersebut dengan kode dibawah ini :

zf-tutorial/index.php

```
<?php
error_reporting(E_ALL|E_STRICT);
date_default_timezone_set('Europe/London');

set_include_path('.' . PATH_SEPARATOR . './library'
     . PATH_SEPARATOR . './application/models/'
```

```
      . PATH_SEPARATOR . get_include_path());
include "Zend.php";

Zend::loadClass('Zend_Controller_Front');

// setup controller
$baseUrl = substr($_SERVER['PHP_SELF'], 0,
        strpos($_SERVER['PHP_SELF'], '/index.php'));
$frontController = Zend_Controller_Front::getInstance();
$frontController->setBaseUrl($baseUrl);
$frontController->setControllerDirectory('./application/controllers');
$frontController->throwExceptions(true);

// run!
$frontController->dispatch();
```

Perlu dicatat bahwa kita tidak menulis tanda `?>` pada akhir file karena hal tersebut tidak diperlukan dan dapat menghindari error yang susah dilacak seperti adanya kode spasi yang berada setelah tanda `?>`.

Mari kita bahas tentang isi dari file ini.

```
error_reporting(E_ALL|E_STRICT);
date_default_timezone_set('Europe/London');
```

Baris diatas memastikan bahwa kita akan melihat semua jenis error yang kita buat (dengan catatan bahwa anda mengeset setting `display_errors` ke mode on). Kita juga mengeset setting time zone yang dibutuhkan oleh PHP 5.1+. Anda juga dapat mengisi konfigurasi tersebut dengan time zone anda.

```
set_include_path('.' . PATH_SEPARATOR . './library'
      . PATH_SEPARATOR . './application/models/'
      . PATH_SEPARATOR . get_include_path());
include "Zend.php";
```

Zend Framework didesain bahwa semua file – file yang diperlukan berada dalam lokasi include. Kita juga mengeset file penempatan model pada direktori include sehingga kita dapat memanggil class model secara mudah. Untuk memulai kita harus menyertakan file `Zend.php` yang memberikan akses ke class `Zend` yang mempunyai fungsi static agar kita dapat memanggil class lainnya dari Zend Framework

```
Zend::loadClass('Zend_Controller_Front');
```

Zend::loadClass berfungsi untuk memanggil class. Proses tersebut dilakukan dengan merubah tanda ‘garis bawah’ pada nama class menjadi pemisah direktori lalu menambahkan akhiran .php di akhir nama file. Sebagai contoh, dari nama class `Zend_Controller_Front` akan dipanggil dari file Zend/Controller/Front.php. Jika anda menggunakan metode yang sama untuk memanggil pustaka library anda, maka anda juga dapat menggunakan perintah `Zend::loadClass()`. Disini kita diharuskan untuk memanggil class front controller dan class router.

Front controller menggunakan class “router” untuk menterjemahkan alamat yang diminta / URL ke fungsi PHP yang akan digunakan untuk menampilkan halaman. Untuk menjalankan router agar bisa berjalan, router tersebut harus beroperasi dimana file index.php berada sehingga router dapat melihat elemen dari url tersebut. Secara singkat, request object (turunan class dari `Zend_Controller_Request_Http`) harus dapat mengenali secara otomatis URL utama. Dalam prakteknya, saya menemukan bahwa hal tersebut tidak berjalan dengan semestinya. Untungnya, kita dapat mengeset base URL kita dengan fungsi `setBaseUrl()`. Fungsi ini terdapat pada http request class dan juga dalam front controller sehingga kita tidak perlu menghawatirkan tentang menurunkan class dari request object.

Sekarang kita akan mengeset URL utama untuk router ke URI yang benar yang terdapat file index.php kita. Saya menggunakan `$_SERVER['PHP_SELF']` untuk menjalankannya dan

seharusnya juga berjalan dengan normal pada sebagian besar konfigurasi server. Jika hal tersebut tidak dapat dilakukan oleh anda, anda harus mengeset variable `$baseUrl` pada system anda ke URL yang benar yang terdapat file `index.php`.

```
// setup controller
$baseUrl = substr($_SERVER['PHP_SELF'], 0,
        strpos($_SERVER['PHP_SELF'], '/index.php'));
$frontController = Zend_Controller_Front::getInstance();
$frontController->setBaseUrl($baseUrl);
```

Kita juga harus mengeset front controller agar aplikasi kita mengenali dimana letak dari direktori controller kita berada.

```
$frontController->setControllerDirectory('./application/controllers');
```

Karena ini merupakan petunjuk awal dan kita menjalankannya dalam system untuk test, saya memutuskan agar front controller menampilkan semua error yang terjadi. Secara standar, front controller akan menangkap error tersebut dan menyimpannya dalam property `_exceptions` pada object “Response” yang dibuatnya. Object response menyimpan semua informasi tentang respon dari URL. Termasuk header, isi halaman dan error. Front controller secara otomatis akan mengirimkan headers dan menampilkan halaman sebelum semua halaman diproses.

Hal ini akan sedikit membingungkan bagi orang yang baru berkenalan dengan Zend Framework, fungsinya sendiri adalah untuk menampilkan error/exception sehingga mudah untuk dilacak. Tentu saja pada server untuk versi release anda tidak boleh menampilkannya kepada user / pengunjung website!

```
$frontController->throwExceptions(true);
```

Akhirnya kita berada dibaris terakhir pada file dimana aplikasi kita akan dijalankan :

```
// run!
$frontController->dispatch();
```

Jika anda menuju alamat http://localhost/zf_tutorial/ , anda seharusnya melihat fatal error seperti dibawah ini :

> **Fatal error**: Uncaught exception 'Zend_Exception' with message 'File "./application/controllers\IndexController.php" was not found.
> (etc.)

Baris diatas memberitahu kita bahwa kita belum mengeset aplikasi yang akan dijalankan. Sebelum kita beranjak lebih jauh, sebaiknya kita membahas aplikasi apa yang akan dibuat di bahasan selanjutnya.

Konsep Website

Kita akan membuat system inventory yang sederhana untuk menampilkan koleksi CD. Halaman utama kita akan memperlihatkan daftar dari koleksi CD dan memperbolehkan kita untuk menambah, merubah dan menghapus CD. Kita akan menyimpan daftar CD tersebut dalam database dengan skema seperti dibawah ini :

Nama Kolom	*Tipe*	*Apakah Null?*	*Catatan*
id	Integer	No	Primary key, Autoincrement
artist	Varchar(100)	No	
title	Varchar(100)	No	

Halaman yang dibutuhkan

Halaman – halaman dibawah ini dibutuhkan untuk menyelesaikan panduan ini.

Halaman Utama	Menampilkan semua daftar album dan menyediakan alamat untuk merubah dan menghapus album. Dan ada juga link untuk menambahkan album baru.
Tambah Album Baru	Halaman ini adalah form untuk menambahkan album baru
Ubah Album	Halaman ini digunakan untuk merubah album yang sudah ada
Hapus Album	Halaman ini akan mengkonfirmasi ketika kita akan menghapus album dan selanjutnya akan menghapus album tersebut.

Mengorganisir Halaman

Sebelum kita mempersiapkan semua file diatas, sangat penting untuk dipahami tentang cara kerja framework dan bagaimana cara kerja setiap file. Setiap halaman dari aplikasi dikenal sebagai "action" dan actions di groupkan menjadi "controllers". Contoh : untuk url dengan format `http://localhost/zf-tutorial/news/view`, controllernya adalah `news` dan actionnya adalah `view`. Cara ini memperbolehkan untuk mengroupkan beberapa action dari controller tersebut. Sebagai contoh, controller `news` dapat mempunyai beberapa action seperti `current`, `archived` dan `view`.

Zend Framework controller memakai special action yang disebut `index` sebagai action yang standar. Misalnya, jika ada url seperti `http://localhost/zf-tutorial/news/` , action `index` dalam controller news akan dijalankan. Untuk url `http://localhost/zf-tutorial/` , akan menyebabkan action `index` dalam controller `index` controller akan dijalankan.

Karena ini adalah tutorial sederhana, kita tidak akan membahas aplikasi yang kompleks seperti login kedalam aplikasi! Modul tersebut dapat menunggu pada tutorial yang terpisah …

Disini kita akan mempunyai 4 halaman yang akan mengorganisir album dan kita akan mengroupkannya kedalam satu controller yang mempunyai 4 action. Kita akan menggunakan controller standar dan ke empat action tersebut adalah :

Halaman	*Controller*	*Action*
Halaman Utama	Index	Index
Tambah Album	Index	Add
Ubah Album	Index	Edit
Hapus Album	Index	Delete

Bersih dan mudah!

Mempersiapkan Controller

Sekarang kita akan mempersiapkan komponen controller. Dalam Zend Framework, controller adalah sebuah class yang dipanggil dengan metode `{Controller name}Controller`. Harap diperhatikan bahwa penulisan `{Controller name}` harus diawali dengan huruf besar. Class ini harus dinamai dengan bentuk `{Controller name}Controller.php` dalam direktori controller. Lagi, penamaan file `{Controller name}` harus dimulai dengan huruf besar dan selanjutnya ditulis dengan huruf kecil. Setiap action adalah fungsi yang dapat diakses dari luar dalam komponen controller dan harus dinamai dengan format `{action name}Action`. Dalam kasus ini, nama `{action name}` harus dimulai dengan huruf kecil.

Pada contoh ini, komponen controller kita bernama `IndexController` yang disimpan pada di `zf-tutorial/application/controllers/IndexController.php`:

zf-tutorial/application/controllers/IndexController.php

```
<?php

class IndexController extends Zend_Controller_Action
{
    function indexAction()
    {
        echo "<p>in IndexController::indexAction()</p>";
    }

    function addAction()
    {
        echo "<p>in IndexController::addAction()</p>";
    }

    function editAction()
    {
        echo "<p>in IndexController::editAction()</p>";
    }

    function deleteAction()
    {
        echo "<p>in IndexController::deleteAction()</p>";
    }
}
```

Pada contoh diatas, kita memberikan perintah agar setiap action pada controller tersebut menampilkan nama actionnya. Test dengan cara mengakses halaman berikut ini :

Alamat	*Teks yang muncul*
http://localhost/zf_tutorial/	in IndexController::indexAction()
http://localhost/zf_tutorial/index/add	in IndexController::addAction()
http://localhost/zf_tutorial/index/edit	in IndexController::editAction()
http://localhost/zf_tutorial/index/delete	in IndexController::deleteAction()

Sekarang kita telah mempunyai pengatur halaman yang sudah berfungsi dan setiap action pada setiap halaman dijalankan dengan menampilkan tulisan. Jika pada tahap ini aplikasi anda tidak berjalan dengan semestinya, silahkan memeriksa bagian *Penanganan Masalah* pada bagian akhir tutorial dan bacalah jika anda menemukan kesulitan.

Sekarang adalah waktunya untuk membuat komponen yang akan ditampilkan yaitu komponen view.

Setting Awal komponen View

Komponen view pada Zend Framework dinamai `Zend_View`. Komponen tersebut memperbolehkan kita untuk memisahkan bagian untuk tampilan dan bagian untuk pemrosesan data.

Penggunaan sederhana dari `Zend_View` adalah :

```
$view = new Zend_View();
$view->setScriptPath('/path/to/view_files');
echo $view->render('view.php');
```

Disini dapat dilihat bahwa jika kita menuliskan struktur ini kedalam setiap action kita akan mengulang ulang kode yang tidak perlu. Kita akan menempatkan setup untuk view ini cukup sekali saja disuatu tempat dan menggunakannya dalam setiap action jika diperlukan.

Desainer dari Zend Framework melihat masalah tersebut dan menyediakan registry untuk menyimpan dan mengambil data dari object. Untuk menyimpan object adalah dengan cara :

```
Zend::register('obj', $object);
```

Dan cara untuk mengambil object :

```
$object = Zend::registry('obj')
```

Untuk mengintegrasikan komponen view kedalam aplikasi kita, caranya adalah dengan membuat komponen view dan menyimpannya kedalam registry dalam file bootstrap kita (zf-tutorial/index.php):

Isi dari file zf-tutorial/index.php

```
...
Zend::loadClass('Zend_Controller_Front');
Zend::loadClass('Zend_Controller_RewriteRouter');
Zend::loadClass('Zend_View');

// register the view we are going to use
$view = new Zend_View();
$view->setScriptPath('./application/views');
Zend::register('view', $view);

// setup controller
$route = new Zend_Controller_RewriteRouter();
$controller = Zend_Controller_Front::getInstance();
...
```

Perubahan pada file tersebut ditandai dengan huruf tebal agar gampang untuk dibaca dan dijelaskan. Kita harus mengambil class `Zend_View` terlebih dahulu dengan menggunakan fungsi Zend::LoadClass() sebelum kita membuat object nya dan mengeset alamat script nya ke dalam direktori khusus untuk komponen views.

Setelah meregister komponen view, sekarang kita membutuhkan file untuk view dengan menampilkan data percobaan dan memanggilnya dengan controller.

Rubah file `IndexController` seperti dibawah ini, sekali lagi perubahan filenya ditulis dengan huruf tebal :

zf-tutorial/application/controllers/IndexController.php

```
<?php

class IndexController extends Zend_Controller_Action
{
    function indexAction()
    {
        $view = Zend::registry('view');
        $view->title = "My Albums";
        $this->_response->setBody($view->render('indexIndex.tpl.php'));
    }

    function addAction()
    {
        $view = Zend::registry('view');
        $view->title = "Add New Album";
        $this->_response->setBody($view->render('indexAdd.tpl.php'));
    }

    function editAction()
    {
        $view = Zend::registry('view');
        $view->title = "Edit Album";
        $this->_response->setBody($view->render('indexEdit.tpl.php'));
    }

    function deleteAction()
    {
        $view = Zend::registry('view');
        $view->title = "Delete Album";
        $this->_response->setBody($view->render('indexDelete.tpl.php'));
    }
}
```

Dalam setiap fungsi, kita mengambil object view dari registry dan merubah isi dari variable Judul/Title lalu menampilkan halaman tersebut dengan template halaman tersebut. Daripada menampilkan (echo) data dari fungsi `render()`, kita menggunakan property body yang merupakan bagian dari system Zend Framework MVC. Response tersebut digunakan untuk menggabungkan mulai dari bagian header, isi body dan error yang mungkin muncul dengan menggunakan system MVC. Komponen front controller kemudian secara otomatis menampilkan header dan diikuti dengan isi dari body pada akhir script.

Sekarang kita akan menambahkan empat file view kedalam aplikasi kita. File – file tersebut disebut sebagai template dan agar mudah, saya menamai setiap template tersebut sama seperti nama file action tetapi dengan menggunakan akhiran file .tpl.php untuk menunjukan bahwa file tersebut adalah file template.

zf-tutorial/application/views/indexIndex.tpl.php

```
<html>
<head>
    <title><?php echo $this->escape($this->title); ?></title>
</head>
<body>
    <h1><?php echo $this->escape($this->title); ?></h1>
</body>
</html>
```

zf-tutorial/application/views/indexAdd.tpl.php

```
<html>
<head>
    <title><?php echo $this->escape($this->title); ?></title>
</head>
<body>
    <h1><?php echo $this->escape($this->title); ?></h1>
</body>
</html>
```

zf-tutorial/application/views/indexEdit.tpl.php

```
<html>
<head>
    <title><?php echo $this->escape($this->title); ?></title>
</head>
<body>
    <h1><?php echo $this->escape($this->title); ?></h1>
</body>
</html>
```

zf-tutorial/application/views/indexDelete.tpl.php

```
<html>
<head>
    <title><?php echo $this->escape($this->title); ?></title>
</head>
<body>
    <h1><?php echo $this->escape($this->title); ?></h1>
</body>
</html>
```

Testing each controller/action should display the four titles in bold.

Kode HTML

Sangat jelas sekali bahwa banyak kode HTML yang terdapat dalam file view kita. Kita akan memisahkan kode html standar yang akan ditampilkan pada setiap action, file tersebut bernama `site.tpl.php`. Dengan cara ini, kita akan memisahkan bagian “luar” dari halaman lalu menampilkan isi dari file yang berisi kode spesifik dari setiap action didalam `site.tpl.php`.

Sekali lagi, file controller kita membutuhkan beberapa revisi :

zf-tutorial/application/controllers/IndexController.php

```
<?php

class IndexController extends Zend_Controller_Action
{
    function indexAction()
    {
        $view = Zend::registry('view');
        $view->title = "My Albums";
        $view->actionTemplate = 'indexIndex.tpl.php';
        $this->_response->setBody($view->render('site.tpl.php'));
    }

    function addAction()
    {
        $view = Zend::registry('view');
        $view->title = "Add New Album";
        $view->actionTemplate = 'indexAdd.tpl.php';
        $this->_response->setBody($view->render('site.tpl.php'));
    }

    function editAction()
    {
        $view = Zend::registry('view');
        $view->title = "Edit Album";
        $view->actionTemplate = 'indexEdit.tpl.php';
        $this->_response->setBody($view->render('site.tpl.php'));
    }

    function deleteAction()
    {
        $view = Zend::registry('view');
```

```
            $view->title = "Delete Album";
            $view->actionTemplate = 'indexDelete.tpl.php';
            $this->_response->setBody($view->render('site.tpl.php'));
        }
    }
```

Kita menggunakan variable baru yang bernama actionTemplate dan menampilkan isi dari file template `site.tpl.php` disemua action.

File view isinya adalah sebagai berikut :

zf-tutorial/application/views/site.tpl.php

```
<html>
<head>
    <title><?php echo $this->escape($this->title); ?></title>
</head>
<body>
       <div id="content">
    <?php echo $this->render($this->actionTemplate); ?>
    </div>
</body>
</html>
```

zf-tutorial/application/views/indexIndex.tpl.php

```
<h1><?php echo $this->escape($this->title); ?></h1>
```

zf-tutorial/application/views/indexAdd.tpl.php

```
<h1><?php echo $this->escape($this->title); ?></h1>
```

zf-tutorial/application/views/indexEdit.tpl.php

```
<h1><?php echo $this->escape($this->title); ?></h1>
```

zf-tutorial/application/views/indexDelete.tpl.php

```
<h1><?php echo $this->escape($this->title); ?></h1>
```

Model Tulisan

Walaupun ini tutorial sederhana, kita membutuhkan file CSS untuk aplikasi ini agar lebih menarik untuk dilihat !

zf-tutorial/application/views/site.tpl.php

```
...
<head>
    <title><?php echo $this->escape($this->title); ?></title>
    <link rel="stylesheet" type="text/css" media="screen"
             href="/zf-tutorial/public/styles/site.css" />
</head>
...
```

zf-tutorial/public/styles/site.css

```
body,html {
    font-size:100%;
    margin: 0;
    font-family: Verdana,Arial,Helvetica,sans-serif;
    color: #000;
    background-color: #fff;
}

h1 {
    font-size:1.4em;
    color: #000080;
    background-color: transparent;
}

#content {
```

```
    width: 770px;
    margin: 0 auto;
}

label {
    width: 100px;
    display: block;
    float: left;
}

#formbutton {
    margin-left: 100px;
}

a {
    color: #000080;
}
```

Koneksi Database

Sekarang kita telah memisahkan antara file pengontrol dan file untuk menampilkan data., sekarang waktunya untuk melihat model dari aplikasi kita dengan lebih dekat. Harap diingat bahwa bagian dari inti aplikasi kita (yang biasa disebut dengan “peraturan bisnis”) terletak pada koneksi dengan database. Kita akan menggunakan class `Zend_Db_Table` di dalam Zend Framework yang digunakan untuk mencari, menyisipkan, merubah, dan menghapus data dari table database.

Konfigurasi

Untuk menggunakan class `Zend_Db_Table`, kita harus menyertakan data seperti nama database yang akan dipakai beserta nama user dan passwordnya. Disini kita tidak akan menggunakan hard-code lagi untuk menyimpan informasi tetapi kita kan menggunakan file konfigurasi untuk menyimpan informasi tersebut.

Zend Framework mempunyai `Zend_Config` yang merupakan object oriented fleksibel untuk mengakses file konfigurasi. File konfigurasi tersebut bisa berbentuk PHP array, file INI atau file XML. Disini kita menggunakan file INI :

zf-tutorial/application/config.ini

```
[general]
db.adapter = PDO_MYSQL
db.config.host = localhost
db.config.username = rob
db.config.password = 123456
db.config.dbname = zftest
```

Konfigurasi diatas silahkan diisi dengan konfigurasi anda, bukan sama seperti milik saya.!

Meggunakan `Zend_Config` sangat mudah sekali, lihat kode dibawah ini:

```
$config = new Zend_Config_Ini('config.ini', 'section');
```

Lihat pada kasus ini, `Zend_Config_Ini` memanggil data dari setiap bagian tertentu dari file INI, bukan semua konfigurasi yang dipanggil. `Zend_Config` memperlakukan tanda “titik” pada parameter sebagai pemisah yang berguna untuk mengelompokkan konfigurasi yang saling terkait. Pada file config.ini kita, nama host, nama user, password, dan nama database dikelompokkan dalam perintah `$config->db->config`.

Kita akan memanggil file konfigurasi dalam file bootstrapper kita (index.php):

File yang berubah di zf-tutorial/index.php

```
...
Zend::loadClass('Zend_Controller_Front');
```

```
Zend::loadClass('Zend_View');
Zend::loadClass('Zend_Config_Ini');

// load configuration
$config = new Zend_Config_Ini('./application/config.ini', 'general');
Zend::register('config', $config);

// register the view we are going to use
$view = new Zend_View();
$view->setScriptPath('./application/views');
...
```

Perubahan ditulis dengan tanda cetak tebal. Kita memanggil class config yang diperlukan (`Zend_Config_Ini`) lalu memanggil bagian yang bernama '`general`' dari file `application/config.ini` kedalam object kita yang bernama `$config`.

Catatan: Dalam tutorial ini, kita tidak benar – benar menyimpan variabel `$config` ke registry, tetapi ini adalah contoh yang bagus dimana pada prakteknya anda tidak harus menyimpan data – data konfigurasi database saja pada file INI tersebut.

Mempersiapkan Zend_Db_Table

Ketika menggunakan `Zend_Db_Table`, kita harus memberitahu konfigurasi database yang telah dipanggil sebelumnya. Untuk menggunakannya kita membutuhkan class turunan dari `Zend_Db` lalu memanggilnya dengan fungsi static `Zend_Db_Table::setDefaultAdapter()`. Sekali lagi, kita akan melakukannya dalam file bootstrapper kita (penambahan kode ditulis dengan cetak tebal):

Isi dari file zf-tutorial/index.php

```
...
Zend::loadClass('Zend_Config_Ini');
Zend::loadClass('Zend_Db');
Zend::loadClass('Zend_Db_Table');


// load configuration
$config = new Zend_Config_Ini('./application/config.ini', 'general'));
Zend::register('config', $config);

// setup database
$db = Zend_Db::factory($config->db->adapter,
                  $config->db->config->asArray());
Zend_Db_Table::setDefaultAdapter($db);

// register the view we are going to use
$view = new Zend_View();
$view->setScriptPath('./application/views');
Zend::register('view', $view);
...
```

Membuat Table

Saya akan menggunakan database MySQL lalu perintah untuk membuat tabelnya dapat dilihat dibawah ini :

```
CREATE TABLE album (
  id int(11) NOT NULL auto_increment,
  artist varchar(100) NOT NULL,
  title varchar(100) NOT NULL,
  PRIMARY KEY (id)
)
```

Jalankan perintah ini dalam aplikasi MySQL client seperti phpMyAdmin atau dengan standar MySQL command-line client.

Menyisipkan Album Test

Kita akan menyisipkan beberapa data kedalam table sebelumnya sehingga kita mempunyai beberapa contoh data untuk ditampilkan dalam aplikasi ini. Saya akan menyisipkan 2 album terlaris dari web Amazon.co.uk:

```
INSERT INTO album (artist, title)
VALUES
       ('James Morrison', 'Undiscovered'),
       ('Snow Patrol', 'Eyes Open');
```

Model

`Zend_Db_Table` `Zend_Db_Table` adalah sebuah class yang abstrak, jadi kita harus menggunakan `Zend_Db_Table` yang secara khusus digunakan untuk menangani data – data album. Secara standar, `Zend_Db_Table` menyarankan agar nama class yang dipakai sama dengan nama table yang dipakai. Jadi, nama class kita adalah `Album` karena nama table `album`. Tambahan lainnya, `Zend_Db_Table` juga berasumsi bahwa tabel anda mempunyai primary key yang bernama `id` yang secara otomatis bertambah sewaktu penyisipan data baru. Kedua sifat tersebut bisa diganti jika memang diperlukan.

Sekarang kita akan menyimpan tabel Album kita dalam direktori 'models':

zf-tutorial/application/models/Album.php

```
<?php

class Album extends Zend_Db_Table
{

}
```

Tidak susah bukan ? Untungnya, untuk kebutuhan disini kita membutuhkan fungsi yang sangat sederhana dan class `Zend_Db_Table` menyediakan fungsi – fungsi tersebut. Jika anda membutuhkan fungsi yang lebih spesifik untuk menangani model anda, class tersebut dapat digunakan.

Menampilkan Daftar Album

Sekarang kita sudah mengeset konfigurasi dan informasi koneksi database, sekarang kita akan mencoba untuk menampilkan beberapa Album pada layar komputer anda. Hal ini bisa dilakukan dengan menggunakan class `IndexController`.

Sangat jelas bahwa setiap tindakan dalam `IndexController` akan memanipulasi database dari `album` dengan class `Album`, jadi sebaiknya kita memanggil class album secara otomatis ketika controller dibuat. Hal ini dilakukan pada fungsi `init()`:

zf-tutorial/application/controllers /IndexController.php

```
<?php

class IndexController extends Zend_Controller_Action
{
    function init()
    {
        Zend::loadClass('Album');
    }

    function IndexAction()
    {

...
```

Ini adalah contoh menggunakan fungsi `Zend::loadClass()` untuk memanggil class kita dan berjalan dengan lancar karena kita telah menyertakan direktori models dengan perintah include pada file `index.php`.

Sekarang kita akan menampilkan daftar album melalui fungsi `indexAction()`:

zf-tutorial/application/controllers/IndexController.php

```
...
function indexAction()
{
    $view = Zend::registry('view');
    $view->title = "My Albums";

    $album = new Album();
    $view->albums = $album->fetchAll();

    $view->actionTemplate = 'indexIndex.tpl.php';
    $this->_response->setBody($view->render('site.tpl.php'));
}
...
```

Fungsi dari `Zend_Db_Table::fetchAll()` mengembalikan `Zend_Db_Table_Rowset` yang memperbolehkan kita untuk mengambil data satu persatu dari setiap baris dan menampilkannya dalam file template :

zf-tutorial/application/views/indexIndex.tpl.php

```
<h1><?php echo $this->escape($this->title); ?></h1>
<p><a href="/zf-tutorial/index/add">Add new album</a></p>
<table>
<tr>
    <th>Title</th>
    <th>Artist</th>
    <th> </th>
</tr>

<?php foreach($this->albums as $album) : ?>
<tr>
    <td><?php echo $this->escape($album->title);?></td>
    <td><?php echo $this->escape($album->artist);?></td>
    <td>
      <a href="/zf-tutorial/index/edit/id/<?php echo $album->id;?>"
          >Edit</a>
      <a href="/zf-tutorial/index/delete/id/<?php echo $album->id;?>"
          >Delete</a>
    </td>
</tr>
<?php endforeach; ?>
</table>
```

Halaman http://localhost/zf-tutorial/ sekarang akan menampilkan daftar (dua) album. Jika aplikasi anda tidak disimpan pada direktori `/zf-tutorial`, rubahlah letak direktorinya sesuai dengan konfigurasi anda.

Menggunakan variable Post dan Get

Dalam aplikasi PHP yang tradisional, perintah global $_POST dan $_GET digunakan untuk mengambil variable dari user. Masalahnya adalah sangat mudah sekali kita menjadi lupa untuk memvalidasi setiap variable sesuai dengan yang seharusnya dimasukkan. Jika tidak dilakukan validasi maka celah keamanan bisa terbuka atau aplikasinya malah bisa menjadi tidak berfungsi. Zend Framework menyediakan class `Zend_Filter_Input` untuk memvalidasi data yang dimasukkan oleh user dengan lebih mudah.

Untuk menggunakan `Zend_Filter_Input`:

```
$postArray = new Zend_Filter_Input($_POST, false);
$username = $postArray->testName('username');
if ($username !== false) {
  // $username is a valid name
}
```

Satu hal yang perlu diingat tentang class `Zend_Filter_Input` adalah class tersebut menghapus array yang telah diproses. Misalnya setelah membuat class `Zend_Filter_Input` untuk $_POST, $_POST kemudian diset menjadi null. Hal ini mengganggu sistem dari MVC sehingga kita menghentikan proses penghapusan array yang dimasukkan dengan mengirimkan parameter false sebagai parameter kedua kepada constructor class `Zend_Filter_Input`.
Disini, kita akan memasang filter tersebut dalam file bootstrap index.php dan menyimpannya dalam `Zend::registry` sehingga kita dapat mengaksesnya sewaktu diperlukan.

Perubahan pada file zf-tutorial/index.php

```
...
Zend::loadClass('Zend_Db');
Zend::loadClass('Zend_Db_Table');
Zend::loadClass('Zend_Filter_Input');

// register the input filters
Zend::register('post', new Zend_Filter_Input($_POST, false));
Zend::register('get', new Zend_Filter_Input($_GET, false));

// memanggil konfigurasi
...
```

Sekarang kita dapat mengambil variabel post dengan cara berikut ini:

```
$post = Zend::registry('post');
$myVar = $post->testAlpha('myVar');
```

Menambahkan Album Baru

Kita telah memasang filter input untuk variabel Post sekarang kita akan menambah kode untuk menambahkan album baru. Bagian ini terdiri dari 2 bagian :

- Menampilkan form baru kepada user untuk mengisi data
- Memproses isi form yang telah dikirimkan dan menyimpannya kedalam database

Hal diatas dilakukan dalam fungsi `addAction()`:

zf-tutorial/application/controllers/IndexController.php

```
...
function addAction()
{
    $view = Zend::registry('view');
    $view->title = "Add New Album";

    if (strtolower($_SERVER['REQUEST_METHOD']) == 'post') {
        $post = Zend::registry('post');
        $artist = trim($post->noTags('artist'));
        $title = trim($post->noTags('title'));

        if ($artist != '' && $title != '') {
            $data = array(
                'artist' => $artist,
                'title'  => $title
            );
            $album = new Album();
            $album->insert($data);

            $this->_redirect('/');
            return;
        }
    }
```

```
        // set up an "empty" album
        $view->album = new stdClass();
        $view->album->artist = '';
        $view->album->title = '';

        // additional view fields required by form
        $view->action = 'add';
        $view->buttonText = 'Add';

        $view->actionTemplate = 'indexAdd.tpl.php';
        $this->_response->setBody($view->render('site.tpl.php'));
    }
    ...
```

Perhatikan bagaimana kita memeriksa variabel `$_SERVER['REQUEST_METHOD']` untuk memeriksa apakah form akan diproses atau tidak. Ketika form diproses, kita akan mengambil nilai dari nama artis dan nama judul album dari array post dengan menggunakan fungsi `noTags()` untuk memastikan bahwa tidak boleh ada kode html dan mengasumsikan bahwa data tersebut telah diisi. Kita menggunakan class model `Album()` untuk menyisipkan informasi yang baru kedalam data baru di database.

Setelah kita menambahkan album baru, kita mengarahkan halaman kehalaman lain dengan menggunakan metode `_redirect()`. Metode `_request()` mengetahui tentang base URL dari aplikasi sehingga kita tidak menggunakan alamat `'/zf-tutorial/'` ketika mengarahkan halaman.

Akhirnya, sekarang kita siap untuk menggunakan form baru pada template. Kita dapat melihat bahwa form action untuk merubah data sama dengan form ini, jadi kita akan menggunakan template yang sama yang digunakan pada file indexAdd.tpl.php dan indexEdit.tpl.php:

Template untuk menambahkan album adalah:

zf-tutorial/application/views/indexAdd.tpl.php

```
<h1><?php echo $this->escape($this->title); ?></h1>
<?php echo $this->render('_indexForm.tpl.php'); ?>
```

zf-tutorial/application/views/_indexForm.tpl.php

```
<form action="/zf-tutorial/index/<?php echo $this->action; ?>"
method="post">
<div>
    <label for="artist">Artist</label>
    <input type="text" class="input-large" name="artist"
        value="<?php echo $this->escape(trim($this->album->artist));?>"/>
</div>
<div>
    <label for="title">Title</label>
    <input type="text" class="input-large" name="title"
        value="<?php echo $this->escape($this->album->title);?>"/>
</div>

<div id="formbutton">
    <input type="hidden" name="id"
        value="<?php echo $this->album->id; ?>" />
    <input type="submit" name="add"
        value="<?php echo $this->escape($this->buttonText); ?>" />
</div>
</form>
```

Ini adalah kode yang mudah. Karena kita akan menggunakan file template `_indexForm.tpl.php` di action edit, kita menggunakan variable `$this->action` daripada

membuat kode yang harus ditulis secara manual. Disini, kita menggunakan variabel untuk teks yang akan dimunculkan pada tombol submit.

Mengubah Album

Mengubah album hampir identik dengan menambahkan album baru, jadi kodenya juga tidak berbeda jauh:

zf-tutorial/application/controllers/IndexController.php

```
...
function editAction()
{
    $view = Zend::registry('view');
    $view->title = "Edit Album";
    $album = new Album();

    if (strtolower($_SERVER['REQUEST_METHOD']) == 'post') {
        $post = Zend::registry('post');
        $id = $post->testInt('id');
        $artist = trim($post->noTags('artist'));
        $title = trim($post->noTags('title'));

        if ($id !== false) {
            if ($artist != '' && $title != '') {
                $data = array(
                    'artist' => $artist,
                    'title'  => $title,
                );
                $where = 'id = ' . $id;
                $album->update($data, $where);

                $this->_redirect('/');
                return;
            } else {
                $view->album = $album->find($id);
            }
        }
    } else {
        // album id should be $params['id']
            $id = (int)$this->_request->getParam('id', 0);
               if ($id > 0) {
                   $view->album = $album->find($id);
               }
    }

    // additional view fields required by form
    $view->action = 'edit';
    $view->buttonText = 'Update';

    $view->actionTemplate = 'indexEdit.tpl.php';
    $this->_response->setBody($view->render('site.tpl.php'));
}
...
```

Perhatikan bahwa kita tidak dalam mode “post”, kita mengambil parameter id dengan menggunakan `getParam()`.

Templatenya adalah sebagai berikut ini:

zf-tutorial/application/views/indexEdit.tpl.php

```
<h1><?php echo $this->escape($this->title); ?></h1>
<?php echo $this->render('_indexForm.tpl.php'); ?>
```

Menghapus Album

Sebagai pelengkap aplikasi ini, kita akan menambahkan fungsi penghapusan Album. Kita akan menyediakan link Delete untuk setiap Album yang muncul dan ketika di klik fungsi tersebut akan menghapus Album yang dipilih.

Kita akan menampilkan form konfirmasi ketika user akan menghapus Album dan ketika dipilih "yes" maka penghapusan Album akan dilakukan.

Kode dibawah ini hampir mirip dengan action tambah dan rubah:

zf-tutorial/application/controllers/IndexController.php

```
...
function deleteAction()
{
    $view = Zend::registry('view');
    $view->title = "Delete Album";

    $album = new Album();
    if (strtolower($_SERVER['REQUEST_METHOD']) == 'post') {
        $post = Zend::registry('post');
        $id = $post->getInt('id');
        if (strtolower($post->testAlpha('del')) == 'yes' && $id > 0) {
            $where = 'id = ' . $id;
            $album->delete($where);
        }
    } else {
        // album id should be $params['id']
        $id = (int)$this->_request->getParam('id', 0);
        if ($id > 0) {
           $view->album = $album->find($id);
           $view->actionTemplate = 'indexDelete.tpl.php';

           // only render if we have an id.
           $this->_response->setBody($view->render('site.tpl.php'));
           return;
        }
    }
    // redirect back to the album list in all cases unless we are
    // rendering the template
    $this->_redirect('/');
}
...
```

Sekali lagi, disini kita menggunakan trik yang sama ketika mengecek notifikasi dari user apakah halaman tersebut menampilkan form konfirmasi atau melakukan proses penghapusan melalui class Album(). Sama seperti insert dan update, penghapusan Album dilakukan dengan memanggil `Zend_Db_Table::delete()`.

Perhatikan bahwa setelah proses penghapusan selesai, aplikasi akan mengarahkan user ke daftar Album di akhir fungsi. Jika ada pengecekan yang gagal, kita akan kembali lagi ke daftar album tanpa memanggil `_redirect()` secara berulang ulang.

Templatenya sendiri berisi form yang sederhana:

zf-tutorial/application/views/indexDelete.tpl.php

```
<h1><?php echo $this->escape($this->title); ?></h1>
<?php if ($this->album) :?>
<form action="/zf-tutorial/index/delete" method="post">
    <p>Are you sure that you want to delete
           '<?php echo $this->escape($this->album->title); ?>' by
           '<?php echo $this->escape($this->album->artist); ?>'?</p>
    <div>
        <input type="hidden" name="id"
```

```
            value="<?php echo $this->album->id; ?>" />
        <input type="submit" name="del" value="Yes" />
        <input type="submit" name="del" value="No" />
    </div>
</form>
<?php else: ?>
<p>Cannot find album.</p>
<?php endif;?>
```

Penanganan Masalah

Jika anda mengalami masalah ketika mengakses halaman lain daripada index/index, sepertinya masalah tersebut terjadi ketika router tidak dapat mendeteksi subdirektori mana aplikasi tersebut berjalan. Dari investigasi saya, hal ini biasanya terjadi ketika alamat website anda berbeda dengan letak direktori utama.

Saat ini, dalam file `index.php`, kita mencoba untuk menyelesaikannya dengan mengambil variabel `$_SERVER['PHP_SELF']`. Jika hal tersebut tidak menyelesaikan masalah, anda harus mengeset variabel $baseURL dengan nilai yang sesuai dengan server anda:

zf-tutorial/index.php

```
...
// setup controller
$baseUrl = '/mysubdir/zf-tutorial';
$frontController = Zend_Controller_Front::getInstance();
$frontController->setBaseUrl($baseUrl);
$frontController->setControllerDirectory('./application/controllers');
...
```

Anda harus mengganti tulisan `'/mysubdir/zf-tutorial/'` dengan alamat yang tepat dimana file index.php berada. Sebagai contoh, jika URL untuk index.php berada di http://localhost/~ralle/zf_tutorial/index.php maka nilai yang cocok untuk variabel `$baseUrl` adalah `'/~ralle/zf_tutorial/'`.

Kesimpulan

Kesimpulannya, kita telah mencoba membuat aplikasi yang sederhana, yang berfungsi, sebuah aplikasi MVC dengan menggunakan Zend Framework. Saya harap anda tertarik dengan hal ini dan tutorial ini cukup informatif. Jika anda menemukan sesuatu yang tidak berjalan sebagaimana mestinya, beritahu saya melalui email ke rob@akrabat.com!

Tutorial ini hanya membahas konsep dasar dari framework; masih banyak class yang harus dipelajari! Anda harus mempelajari buku manual (http://framework.zend.com/manual) dan melihat pada wiki (http://framework.zend.com/wiki) untuk pandangan yang lebih luas! Jika anda tertarik dalam pembuatan framework, maka anda harus mengunjungi website development wiki (http://framework.zend.com/developer).

Akhir Kata

Ketika menulis tutorial ini, satu hal yang hilang adalah cara yang lebih baik untuk membuat model. Saya dapat melihat kenapa pattern ActiveRecord sekarang ini sangat populer!

Saya melihat bahwa `Zend_Db_Table` akan lebih lengkap dalam beberapa bulan mendatang sebelum keluar versi 1.0. Dalam Framework wiki ada juga proposal untuk `Zend_Db_Model`. Penjelasan singkat dari `Zend_Db_Model` adalah: “Sebuah object yang membungkus tabel database dan view, termasuk akses database dan memperbolehkan domain logic dalam data tersebut”. Beberapa fungsi dari class tersebut akan sangat berguna ketika membuat suatu aplikasi dengan Zend Framework.

Secara umum, Zend Framework mempunyai bentuk struktur yang bagus.