

Memahami Workflow Zend_Controller dalam Zend Framework (Bag 2)

12 07 2008

Setelah mengetahui tentang komponen – komponen yang banyak digunakan dalam Controller Zend Framework seperti yang telah dibahas dalam artikel “Memahami Workflow Zend_Controller dalam Zend Framework (Bag 1)”, perlu diketahui juga aturan penyusunan folder yang baik dalam membangun aplikasi menggunakan Zend Controller sebagai berikut:

```
- etc
  - data
  - libs
    - services
    - share
  - modules
    - default
    - module_one
    - module_two
    - dst...
- www
  - js
  - css
  - image
  - dll...
```

Kenapa strukturnya seperti itu, berikut pembahasannya; dari dua folder root tersebut, etc berfungsi sebagai tempat penyimpanan file – file system yang berhubungan dengan aplikasi yang keberadaannya harus disembunyikan dari user baik itu browse internet atau intranet, sedangkan file www dipakai hanya untuk menyimpan file index.php dan .htaccess sebagai pintu gerbang utama untuk mengakses aplikasi.

Selain fungsi yang telah disebutkan diatas, folder “www” dan “etc” juga mempunyai fungsi lain, jika aplikasi yang kita bangun banyak menggunakan javascript maka file yang berextensi js lebih baik disimpan “www/js” dan diberikan penamaan yang spesifik untuk setiap fungsi yang ditangani oleh javascript tersebut, kemudian untuk mempercantik tampilan page seperti font, margin, indent, table, row dan lain – lain, akan lebih baik jika fungsi tampilan tersebut diserahkan pada file css (cascade style sheet), file css tersebut disimpan difolder “www/css”, kemudian jika aplikasi yang kita bangun tersebut menggunakan banyak gambar banner atau icon maka file yang berextensi jpg, gif, bmp dan lain – lain yang bersifat image disimpan di folder “www/image”, selain dari ketiga folder tersebut, root folder “www” masih dapat juga digunakan untuk folder lainnya jika diperlukan, seperti folder “www/ajax” jika ada fungsi ajax yang dipakai dalam aplikasi tersebut.

Folder etc berfungsi menyimpan file core dari aplikasi yang berextensi php dan phtml, juga file “ini” yang berfungsi sebagai konfigurasi (jika dipakai), jika ada file berextensi selain dari tiga diatas, maka file tersebut adalah file data dapat berupa pdf, doc, xls dan lain – lain, file tersebut harus

disimpan dalam folder "etc/data", di dalam folder "data" masih dapat dibuat sub folder lagi tergantung dari jumlah data dan pengelompokan data aplikasi tersebut. Folder "etc/libs" digunakan untuk menyimpan file dengan fungsi yang khusus menangani transaksi INSERT, SELECT, UPDATE, DELETE terhadap database yang biasa disebut file services, dengan aturan penamaan sebagai berikut "SuatuFungsiServices.php" (kata dengan cetak miring dan tebal akan disesuaikan dengan nama fungsi yang ditangani oleh file php service tersebut, file services tersebut disimpan dalam folder "etc/libs/services", setelah folder services ada juga folder share yang berfungsi untuk menyimpan file php yang mempunyai fungsi reusable atau fungsi yang dapat dipakai berkali – kali dan digunakan disetiap modul dalam aplikasi tersebut, dengan susunan folder "etc/libs/share".

Selain dari folder tersebut diatas masih ada beberapa folder lagi yang akan dibahas yaitu "data/modules", folder ini berfungsi menyimpan file php dan phtml yang digunakan sebagai source – code utama yang harus dimiliki oleh aplikasi web yang menggunakan Zend Framework, dan didalamnya masih terdapat beberapa subfolder lagi tergantung dari berapa jumlah modul yang akan dibangun dari aplikasi tersebut, tetapi minimum harus ada satu folder didalam "modules" yang wajib ada dengan nama "default", dengan susunan folder sebagai berikut "etc/modules/default".

Semua subfolder yang berada dibawah modules akan banyak atau sedikit tergantung dari jumlah modul yang dibangun dalam aplikasi web tersebut, kita ambil contoh aplikasi ERP didalamnya terdiri dari modul; procurement, inventory, production, sales, accounting dan lain – lain, dari nama – nama modul tersebut maka kita dapat petakan folder yang berada dalam subfolder modules sebagai berikut:

```
- etc
  - modules
    - default
    - procurement
    - inventory
    - production
    - sales
    - accounting
    - dst...
```

Kemudian dari subfolder yang berada dibawah folder modules tersebut masih ada aturan Standard Zend Framework yang baik untuk diikuti mengenai penyusunan Zend Controller, yang jika kita petakan maka akan seperti berikut:

```
- etc
  - modules
    - default
      - controllers
        - file ControlSatuController.php
        - file ControlDuaController.php
        - dst
    - views
```

```

- filters
- helpers
- scripts
  - error
    - file error.phtml (untuk error handling)
  - index
    - file index.phtml
    - file viewsatu.phtml
    - file viewdua.phtml
  - submodul1
    - file index.phtml
    - file viewsatu.phtml
    - file viewdua.phtml
  - dst
- procurement
  - controllers
    - file ControlSatuController.php
    - file ControlDuaController.php
    - dst
  - views
    - filters
    - helpers
    - scripts
      - error
        - file error.phtml (untuk error handling)
      - index
        - file index.phtml
        - file viewsatu.phtml
        - file viewdua.phtml
      - submodul1
        - file index.phtml
        - file viewsatu.phtml
        - file viewdua.phtml
      - dst
- dst...

```

Setelah kita mengetahui tentang aturan folder, maka kita sekarang akan membahas tentang penamaan file php untuk controller dan file phtml untuk view yang sebaiknya diterapkan dalam aplikasi yang akan kita bangun, jika kita mempunyai sebuah file "IndexController.php" dalam folder "etc/modules/default/controllers" maka di dalam file controller tersebut baiknya ditulis seperti berikut:

```

<? require_once 'Zend/Controller/Action.php';

class IndexController extends Zend_Controller_Action {
    public function indexAction() {
        // indexAction default kepunyaan IndexController dalam modul default
    }

    public function __call($method, $args) {
        if ('Action' == substr($method, -6)) {
            // Jika method action tidak ditemukan, maka function akan render template
            error
            return $this->render('error');
        }
        // method action lain yang tidak dikenal akan throw sebuah exception
        throw new Exception('Invalid method "' . $method . '" called');
    }
} ?>

```

Dari kode contoh diatas dapat kita ketahui bahwa sebuah file "IndexController.php" baiknya menggunakan nama class "IndexController" dan setiap file Controller berarti sama dengan satu subfolder dalam folder scripts, dan juga setiap satu method Action dalam sebuah controller sama dengan satu file phtml dalam subfolder script tersebut, jadi jika kita petakan maka akan seperti berikut:

```
- default
- controllers
  - IndexController.php (satu method indexAction = satu file index.phtml di folder index)
  - ErrorController.php (satu method errorAction = satu file error.phtml di folder error)
- views
  - filters
  - helpers
  - scripts
    - error (folder ini harus ada jika ErrorController.php ada)
      - error.phtml (untuk error handling)
    - index (folder ini harus ada jika IndexController.php ada)
      - index.phtml
```

Berbeda halnya jika kita akan membuat suatu controller dalam subfolder modules selain dari subfolder "default", misal jika kita buat folder "procurement" maka kita akan mendapati suatu file "IndexController.php" dengan kode sebagai berikut:

```
<? require_once 'Zend/Controller/Action.php';

class Procurement_IndexController extends Zend_Controller_Action {
    public function init() {
        // Inisialisasi ini hanya berlaku untuk Controller ini saja;
        // Hanya berpengaruh terhadap seluruh actions, yang dipanggil oleh
        // Controller ini:

        // line ini dipakai jika seluruh function method tidak menggunakan file
        // *.phtml
        $this->_helper->viewRenderer->setNoRender(true);
    }

    public function indexAction() {
        // indexAction default kepunyaan IndexController
        // dalam modul procurement
    }
} ?>
```

Sebagaimana telah kita lihat dalam kode diatas, dapat kita ketahui jika controller tersebut digunakan selain dari modul default maka baiknya penamaan class menggunakan formula "NamaModul_NamaControllerController", kemudian penamaan method function pun baiknya menggunakan formula "NamaFilePhtmlAction()" kecuali jika suatu function tidak akan diasosiasikan terhadap suatu file phtml maka diawal function harus mendeklarasikan perintah sebagai berikut "\$this->_helper->viewRenderer->setNoRender(true);" dengan kegunaan supaya disaat function tersebut dijalankan, maka proses render terhadap file phtml dengan asosiasi nama

function yang sama tidak dijalankan. Jika kita asosiasikan struktur controller diatas maka akan menjadi seperti berikut:

```
- procurement
  - controllers
    - IndexController.php (dengan isi satu method indexAction = satu file
      index.phtml di folder index)
    - ErrorController.php (dengan isi satu method errorAction = satu file
      error.phtml di folder error)
  - views
    - filters
    - helpers
    - scripts
      - error (folder ini harus ada jika ErrorController.php ada)
        - error.phtml (untuk error handling)
      - index (folder ini harus ada jika IndexController.php ada)
        - index.phtml
```

Sekarang kita akan coba membahas bagaimana sebuah aplikasi Zend dibangun dari sebuah controller, sebagaimana telah dijelaskan diatas, maka kita akan mulai membahas dari pertama kali sebuah aplikasi Zend dibangun, sebelum kita melangkah ke bagian controller ada baiknya kita mengenal dulu file "index.php" dan ".htaccess" sebagai pintu masuk utama (bootstrap) kedalam aplikasi.

File htaccess adalah sebuah file yang aturan penamaannya unik ".htaccess" (perhatikan tanda titik di depan htaccess) yang berisikan aturan yang akan diterapkan pada direktori dimana file ini disimpan. Yang berisi juga aturan user siapa, atau group mana saja yang boleh mengakses direktori tersebut. File .htaccess merupakan sebuah file teks yang bisa dibuat dengan text editor biasa. Sering kali juga disebutkan sebagai htaccess saja, atau seringkali disebutkan juga sebagai distributed configuration file. Disebut distributed configuration file karena ".htaccess" bisa di-copy-kan ke direktori mana saja dan akan langsung berdampak pada direktori tersebut ketika diakses melalui web browser.

Jika kita sudah mempunyai file .htaccess di folder www maka isilah file tersebut dengan script sebagai berikut:

```
RewriteEngine on
#RewriteBase /nama_sub_domain
RewriteRule !\.(js|ico|gif|jpg|png|css|html)$ index.php
```

Penulisan kode diatas mengatur beberapa file selain source-code seperti file image, css (stylesheets) dan file html agar terdaftar pada front-controller, jika masih ada file lain yang ingin didaftarkan lagi terhadap front-controller maka extension file yang dimaksud dapat dituliskan dalam "RewriteRule". Selain itu kita juga dapat menggunakan "RewriteBase" jika kita ingin membuat

aplikasi yang kita bangun merupakan subdomain dari aplikasi lain. Untuk kondisi diatas ditambahkan tanda “#” berarti fungsi tersebut saat ini sedang tidak digunakan.

Setelah penulisan .htaccess selesai maka kita akan menuju “bootstrap-file”, dan tugas ini dilakukan oleh file “index.php” yang berada di folder “www”, jika susunan file terhadap folder “www” kita petakan maka akan seperti berikut:

```
- etc (pembahasan tentang folder etc diatas sudah)
- www
  - .htaccess
  - index.php
```

Untuk pertama kali penulisan kode didalam “index.php” tidak perlu terlalu rumit, cukup hanya beberapa baris saja, tetapi hal ini akan terus berkembang sesuai dengan kebutuhan aplikasi yang kita bangun, berikut contoh sederhananya:

```
<? set_include_path('.' . PATH_SEPARATOR . '../etc/libs'
    . PATH_SEPARATOR . '../etc/data'
    . PATH_SEPARATOR . get_include_path());

require_once 'Zend/Loader.php';

Zend_Loader::loadClass('Zend_Registry');
Zend_Loader::loadClass('Zend_Controller_Front');

$registry = Zend_Registry::getInstance();

$registry->set('basepath', '');

$ctl = Zend_Controller_Front::getInstance()
    ->setBaseUrl($registry->get('basepath'))
    ->addModuleDirectory('../etc/modules');
$ctl->dispatch(); ?>
```

Sekarang kita akan bahas baris kode diatas secara bertahap. Pada baris ke nol sampai dua kita lihat “set_include_path” yang berfungsi untuk override path yang sudah dituliskan dalam “DocumentRoot” pada file “httpd.conf” dengan terlebih dahulu meng-include DocumentRoot sebelumnya dengan perintah “get_include_path()” kedalam path yang baru.

Berikutnya pada baris perintah “require_once ‘Zend/Loader.php’;” perintah ini mendefinisikan file “Loader.php” agar terbaca oleh bootstrap dan merupakan file library untuk baris perintah berikutnya, kemudian dua baris berikutnya merupakan perintah turunan dari baris sebelumnya, yang merupakan perintah untuk inisialisasi class “Zend_Registry” dan “Zend_Controller_Front” dalam class Loader, sehingga jika kedua class tersebut dibutuhkan di file controller maka tidak perlu di include ulang disetiap class php, cukup di bootstrap saja. Jumlah file library Zend yang di daftarkan dalam class Loader dapat bertambah banyak disesuaikan dengan kebutuhan aplikasi.

Sekarang kita lihat baris kode berikut “\$registry = Zend_Registry::getInstance();” perintah ini digunakan untuk inialisasi object registry yang nilainya tidak dapat dirubah, sedangkan Zend_Registry sendiri berfungsi untuk menyimpan object disertai nilainya dalam memory buffer server. Dengan menyimpan suatu nilai dalam object registry maka kita dapat mengakses nilai itu dibagian manapun dalam aplikasi tersebut. Mekanisme ini merupakan alternatif penggunaan global storage.

Dalam perintah “\$registry->set('basepath', ”);” kita melihat penggunaan object static \$registry melakukan proses save object bernama basepath dengan nilai blank, dimana object ini dapat diakses dibagian manapun dalam aplikasi tersebut, sedangkan object basepath sendiri harus disetting nilai selain blank jika aplikasi yang sedang kita bangun merupakan bagian subdomain dari sebuah application-server atau domain yang lebih besar.

Kemudian pada 4 baris terakhir yang merupakan perintah utama dalam bootstrap file, baris ini “\$ctl = Zend_Controller_Front::getInstance()” berfungsi membuat static object class \$ctl yang akan menunjukan posisi file controller yang dapat dieksekusi pada baris terakhir terdapat perintah “\$ctl->dispatch();” yang berguna untuk mengakhiri perintah “Zend_Controller_Front::getInstance()” tetapi sebelum baris tersebut dieksekusi ada variabel yang diinputkan ke method “setBaseUrl” dan “addModuleDirectory” yang diperlukan untuk menentukan aktifitas aplikasi dan folder modules yang akan di eksekusi.

Seperti yang telah dijelaskan diatas, controller mempunyai ketentuan tertentu dalam pengaturan folder dan penamaan class php. Dan dengan aturan ini didapatkan view dalam browser yang terproteksi source-code-nya dari user. Misal jika kita akan mengakses file main.phtml yang berada di modul procurement maka address bar browser hanya akan memperlihatkan address-path sebagai berikut “http://localhost/procurement/index/main” padahal posisi sebenarnya dari file tersebut jika kita petakan dalam folder sebagai berikut:

```
- etc
  - modules
    - procurement
      - controllers
        - IndexController.php
      - views
        - filters
        - helpers
        - scripts
          - index (folder ini harus ada jika IndexController.php ada)
            - main.phtml (file ini yang hendak diambil pada address-path diatas)
```

dan dengan kode pada controller sebagai berikut:

```
<? require_once 'Zend/Controller/Action.php';
```

```
class Home_IndexController extends Zend_Controller_Action {
    public function init() {
        //di isi sesuai keperluan
    }

    public function indexAction() {
        //di isi sesuai keperluan
    }

    public function mainAction() {
        //di isi sesuai keperluan
    }
} ?>
```

Jika ada kekurangan atau masukan yang dapat meningkatkan kemampuan jangan sungkan untuk memberikan komentar.