15/01/10 - Petita guia de Zend Framework

Zend Framework és un framework per a PHP, és a dir un sistema de treball i conjunt de llibreries per a desenvolupar aplicacions en PHP.

Zend es basa en el MCV ? Model Control Viewer, el que significa:

- Control ? El controlador (controller) és qui rep la petició. Fa les tasques que hagi de fet, com càlculs, i demana les dades al Model.
- Model ? El Model és qui recupera o introdueix les dades de la base de dades.
- Viewer ? Visualitzador. És la part que conté l'HTML i que pot disposar de les dades que genera el Model si el Controller l'autoritza.

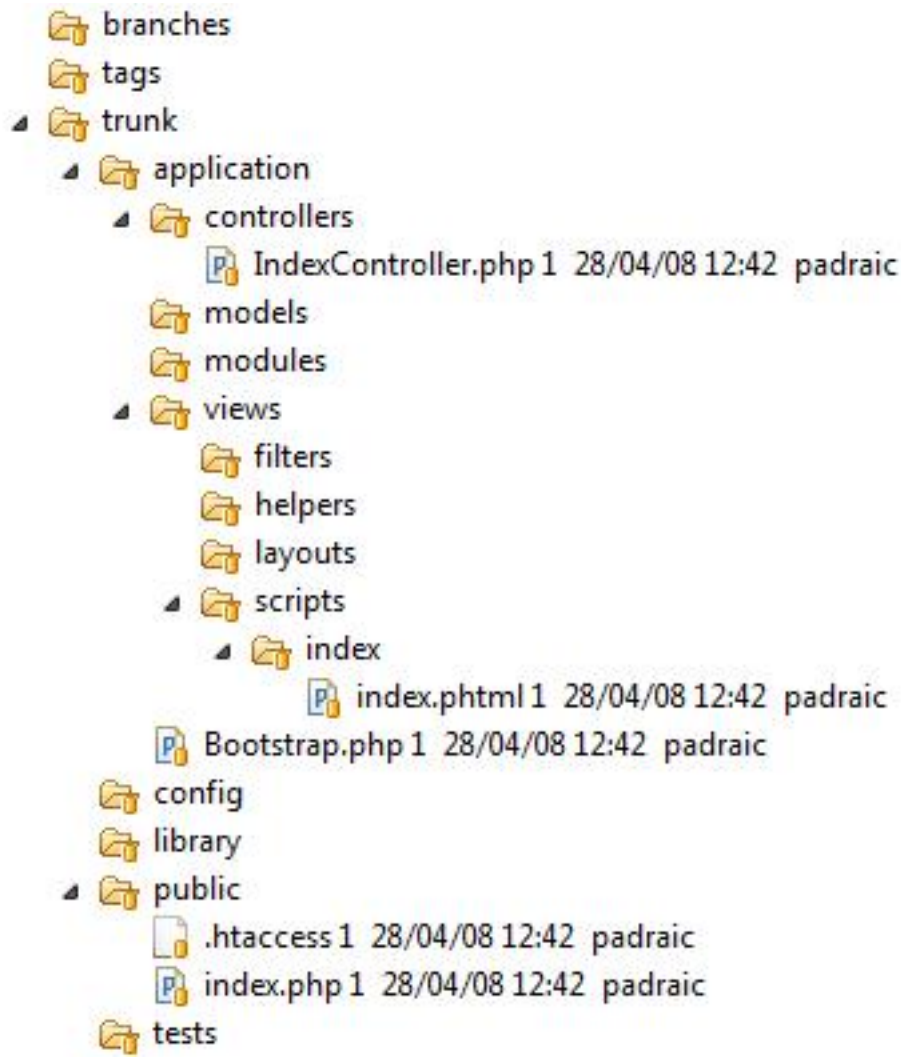
Zend ofereix abstraccions. Per exemple, Zend_DB per a base de dades, ofereix unes funcions per les quals tant és si el servidor de base de dades és un MySql, un oracle, un sql server de microsoft o un altre, les crides sempre són les mateixes i a Zend_DB.

El framework s'encarrega de fer que les dades s'entreguin a l'aplicació de forma transparent.

Algunes de les facilitats que ofereix el framework són:

- Zend_Feed ? per a consumir i publicar feeds RSS i Atom.
- Zend_Pdf ? per a crear, editar i carregar documents Pdf.
- Zend_Search ? Per a fer cerques sofisticades sobre els nostres texts.
- Zend_Service_Amazon, Zend_Service_Flickr, and Zend_Service_Yahoo ? per a emprar les API (funcions) d'aquestes empreses

Zend és tot programació orientada a objectes.



En la imatge de l'esquerra podem veure un arbre de directoris d'una aplicació escrita en Zend.trunk és el nom del projecte.

Dins d'application hi tenim el directori controller, que és on hi haurà els controladors (el codi).

També hi tenim model, on aniran les classes que accedeixen a base de dades.

I views que és on hi haurà les vistes, o codi html amb crides per a fer servir les dades accedides pels Models.

A la carpeta controllers /application/controllers/ de la il·lustració, hi trobem un arxiu que es diu IndexController.php

Aquest arxiu: IndexController.php és qui rebrà les crides que fem a l'aplicació web.

Es el controlador Index.

En Zend el nom dels arxius és molt important, ja que el nom de l'arxius és la manera per la que es localitza l'encarregat de fer determinades tasques.

La plana més senzilla que faríem amb Zend és un "Hola món!".

```
<?phpArrayArrayclass IndexController extends
Zend_Controller_ActionArray{ArrayArray
public function indexAction()Array    {Array
$this->view->title = 'Hola, món!';Array    }ArrayArray}
```

Les majúscules també són molt importants com ara en el cas d' indexAction.

Zend_Db_Table sempre assumirà que la primary key (la clau primària d'una taula) és **id** si no s'especifica el contrari.

Els paràmetres de connexió a la base de dades s'especifiquen a **/config/config.ini**

```
db.adapter = PDO_MYSQLArraydb.host = localhostArraydb.username
= usuari_de_la_base_de_dadesArraydb.password =
mot_de_pasArraydb.dbname =
```

Els Helpers o View Helpers són unes funcions d'ajuda, per exemple per a construir formularis.

Per exemple el **formCheckbox**:

```
echo $this->formCheckBox('Independència', null, null, array
(Array
    'Checked' => 'Sí',Array
                          'unChecked' => 'No'));
```

El codi anterior generarà un control de tipus checkbox amb l'opció Sí marcada i la No desmarcada.

Els Helpers, quan se'ls crida, es carreguen en memòria i s'instancien automàticament si és necessari.

Un que és especialment útil és **BaseUrl**.

```
$this->baseUrl(); // Això ens retorna la Url on es troben els scripts.
```

Emprant baseUrl ens assegurem que cridem els scripts de l'indret pertinent, i podem construir aplicacions que no depenguin de rutes absolutes o estàtiques al servidor web.

En el Model podrem recuperar les dades i ho farem mitjançant **get**?s i **set**?s.

Per exemple: getNom o getEmail.

Emprem get's predefinits al Model enlloc del "magic method" __set o __get perquè es calcula que és un 300% més ràpid fer-ho així.

Pareu atenció que de nou les majúscules són molt importants.

getNom ha de coincidir amb el nom de la variable que ens passaran pel formulari **nom** , en minúscules, ja que a setOptions es convertiran els paràmetres rebuts del formulari en crides a *setNom* (es converteix la primera lletra de la variable a majúscula).

```
public function setOptions(array $opcions)Array{Array $metodes = get_class_methods($this);Array     for each ($opcions as $clau => $valor)Array     {Array         $metode = "set".ucfirst($clau); // això és el que posa en majúscula la primera lletra com setIdArray         if (in_array($metode, $metodes))Array         {Array $this->$metode($valor);Array         }Array     }Array return $this;Array}
```

Habitualment tindrem un únic mètode anomenat **save** que farà un insert o un update en funció de si ens passen un id o no, i comprovant si rebem un post de formulari (isPost).

Els arxius a **application/components/** com ara test.yml serveixen per a accedir a la base de dades si ho volem, encara que jo em decanto per utilitzar directament el

Model.

Les crides al projecte via web es fan seguint la següent estructura:

```
http://servidor/nom_projecte/Nom_Controller/Nom_Action/Nom_Paràmetre1/Valor_Paràmetre1/.../Nom_ParàmetreN/Valor_ParàmetreNPer exemple:http://codic.cat/ProjecteC/indexCom index és el Controller per defecte si no diem res s'assumirà index: http://codic.cat/ProjecteC o per exemple: http://codic.cat/ProjecteC/edit/id/23
```

La pàgina final es renderitzarà en base a les vistes (views).

```
views/scripts/index/index.phtml
```

Per al controller Index tindrem a **views/scripts/index/** un arxiu .phtml per a cadascuna de les accions.

En aquest cas: com el controller es diu Index i l'acció es diu index tindrem:

En qualsevol moment podem fer un debug (veure com estan les coses) amb:

```
Zend_Debug::dump(); exit;
```

Dins de index.phtml per a tenir accès a les dades definides al Controller i recuperades pel Model, farem el següent:

```
<?phpArray    $dades_index = $this->index_db;Array    echo $dades->id;Array?>
```

On previament hem definit dins el controller index_db.

```
$indexModelObj = new Model_Index();Array$this->view->index_db = $indexModelObj->getDades;
```

On getDades és un mètode al Model index que recupera uns valors.

D'aquesta manera amb **$this->view>index_db =** ... permetem que index_db estigui disponible a la vista index.phtml, és a dir, que tingui accès a aquelles dades.

També podem obtenir informació sobre les dades fent:

```
$o_info = $this->_table->info();
```

O també:

```
$o_promo = new Promocio();Array$st_informacioPromocio = $o_promo->info();
```

On **Promocio()** és el nom de la classe sobre la que volem extreure informació.

Per exemple:

Array ([schema] => [name] => promocions [cols] => Array ([0] => promo_PK [1] => promo_desc) [primary] => Array ([1] => promo_PK) [metadata] => Array ([promo_PK] => Array ([SCHEMA_NAME] => [TABLE_NAME] => promocions [COLUMN_NAME] => promo_PK [COLUMN_POSITION] => 1 [DATA_TYPE] => int [DEFAULT] => [NULLABLE] => [LENGTH] => [SCALE] => [PRECISION] => [UNSIGNED] => 1 [PRIMARY] => 1 [PRIMARY_POSITION] => 1 [IDENTITY] => 1) [promo_desc] => Array ([SCHEMA_NAME] => [TABLE_NAME] => promocions [COLUMN_NAME] => promo_desc [COLUMN_POSITION] => 2 [DATA_TYPE] => varchar [DEFAULT] => [NULLABLE] => [LENGTH] => 255 [SCALE] => [PRECISION] => [UNSIGNED] => [PRIMARY] => [PRIMARY_POSITION] => [IDENTITY] =>)) [rowClass] => Zend_Db_Table_Row [rowsetClass] => Zend_Db_Table_Rowset [tableClass] => [referenceMap] => Array ( ) [dependentTables] => Array ( ) [sequence] => 1)

Uns bons consells per als XHTML, per tal que es validin bé són:

- Posar els noms dels tags HTML en minúscules
- Tancar els tags com a:
.
- En els camps select quan és el valor seleccionat fet **<option selected="selected" value>** i no el clàssic <option selected value>
- Posar un **alt** sempre a les imatges.

Per exemple: <input type="text" name="camp_text" />

Pe:

Zend_DB_Table és el que s'acostuma a utilitzar per a treballar amb la base de dades.

És a més alt nivell, per a treballar amb conjunts de registres o recordsets.

De vegades necessitarem treballar amb Zend_DB per a treballar amb les dades a nivell de files i no a nivell de recordsets.

Zend_DB proporciona:

- select
- fetchAll
- fetchRow

- find
- delete
- quote
- quoteinto

Quote i quoteinto s'asseguren que no utilitzem noms de variables que són camps reservats de la base de dades.

Algunes bases de dades tenen paraules reservades que serveixen per a dur a terme determinades funcionalitats, i d'altres tenen altres. Per exemple, EMAIL pot ser una paraula reservada en un sqlserver mentre que en un MySql no.

Com Zend_DB_Table no té quote ni quoteinto necessitem l'Adapter per a obtenir un Zend_DB.

Nogensmenys Zend_DB fa un quoteinto per a les selects però no per als delete.

Podeu debuguejar els errors dins del catch de les Exception:

```
throw new Exception("Exception a l'arxiu ".__FILE__.' línia
'.__LINE__.', class '.__CLASS__.' funció '.__FUNCTION__.':
'.$e->getMessage());
```

Adjunto algunes guies:

Understanding the Zend Framework (IBM)

Part 3: A Simple Hello World Tutorial

Part 4: Setting the Design Stage with Blueprint CSS Framework and Zend_Layout

Part 5: Creating Models with Zend_Db and adding an Administration Module