# Projet Anca
# Un site web avec Cake PHP

Serkan Eryilmaz

14 juin 2011

## Table des matières

# 1 Introduction

CakePHP est un framework de développement rapide de sites web en PHP. Il propose des outils pour aider à la création de sites web rapidement, en utilisant le design pattern MVC. Il permet de se concentrer sur le coeur du développement, avec une approche modulaire et respectant les principes de programmation SOLID, sans trop se soucier de ce qui se passe "derrière".

**M** Il intègre un puissant ORM, gère la validation de formulaires, les relations de type OneToOne, OneToMany, ManyToMany et ManyToOne, etc... Les classes de modèles sont les classes qui contiennent principalement les données de notre site web.

**V** Les vues sont en fait des templates PHP (fichiers .ctp) qui ont accès à la session, aux données passées par le controlleur et aux 'helpers'. Ces 'helpers' sont en fait des API qui permettent de faire des taches répétitives rapidement (exemple : générer un formulaire, générer un tableau, etc.). Les helpers permettent aussi d'injecter du code javascript (AJAX) directement dans le template sans se soucier d'écrire du javascript, par exemple.

**C** Les controlleurs sont les classes qui contiennent la logique applicative. C'est eux qui vont être appellés à chaque requête, qui vont aller chercher les modèles correspondant de la base de donnée et choisir la vue à afficher et leur passer les données nécessaires. Chaque méthode d'un controlleur correspond à une "action" à implémenter, et la méthode correspondante est executée automatiquement par le système de routage de CakePHP en fonction de l'url de la requête.

# 2 Routage

CakePHP a un système de routage complet qui permet d'associer une URL à un controlleur et une action. Par défaut, le routage est composé comme celui-ci : *http ://www.site.com/{controlleur}/{action}{[/paramètre]*}*. Si aucune action n'est définie dans l'URL, alors l'action par défaut est "index". Le controlleur et l'action par défaut (pour l'url "*http ://www.site.com/*") sont définis dans la configuration du routeur, c'est à dire dans le fichier */app/config/routes.php*.

### Exemple

Router : :connect('/', array('controller' => 'catalog', 'action' => 'index'));

## 3 Modèle

Les modèles sont définis dans le dossier /app/models et sont découvert automatiquement par CakePHP, il suffit donc simplement de créer les fichiers modèles dans ce dossier. Le nom d'un modèle doit correspondre à la table relationnelle sous-jacente, sans le s. Par exemple, si nous avons une table relationnelle d'utilisateurs nommé "users", notre modèle s'appellera "User", et représentera donc un tuple de notre table.

Bien que nous ne devons pas définir les attributs de notre table dans le modèle, nous pouvons définir les relations que l'entité a vis-à-vis des autres entitées. On définit ceux-ci dans une variable et CakePHP les découvrira automatiquement. Le modèle est aussi l'endroit où nous décrivons la validité d'une instance, dans la variable $validate.

Il y a aussi une série de méthodes que l'on peut redéfinir afin d'avoir une maitrise supplémentaire sur la validation. Dès que la validation n'est plus superficielle, nous pouvons implémenter ces méthodes afin d'avoir un contrôle supplémentaire sur la manière dont on valide un tuple.

Listing 1 – Exemple modèle

```
1  class User extends AppModel {
2    var $name = 'User';
3    var $hasAndBelongsToMany = array(
4      'Product' => array(
5        'className' => 'Product',
6        'joinTable' => 'users_products',
7        'foreignKey' => 'user_id',
8        'associationForeignKey' => 'product_id',
9        'unique' => true
10     )
11   );
12   var $validate = array(
13     'username' => array(
14       'rule' => '/^[a-z0-9]{4,40}$/i',
```

```
15        'message' => 'This field must have between\
16                     4 and 40 alphanumeric characters.'
17      ),
18      'password' => array(
19        'rule' => '/^[a-z0-9]{4,40}$/i',
20        'message' => 'This field must have between\
21                     4 and 40 alphanumeric characters.'
22      ),
23      'email' => array(
24        'rule' => 'email',
25        'message' => 'Please supply a valid\
26                     email address.'
27      )
28    );
29
30    function beforeValidate() {
31      if (!$this->id) {
32        if ($this->findByUsername(
33          $this->data['User']['username'])
34        ) {
35          $this->invalidate('username_unique');
36          return false;
37        }
38        return true;
39      }
40    }
41  }
```

# 4 Controlleur

Les classes controlleur, qui sont définis dans le dossier */app/controllers/*, sont les classes où nous auront donc toute notre logique applicative. C'est là que nous allons récupérer les paramètres de la requête, chercher l'information (si nécessaire) dans la base de donnée, et finalement passer ces données à la vue afin qu'elle l'affiche comme elle le souhaite.

CakePHP permet de définir une classe "AppController", qui est en fait la classe "mère" de tous les autres controlleurs. Nous pouvons y insérer la

logique globale de notre application. Par exemple s'assurer que l'utilisateur est bien enregistré et a les privilèges pour accéder à certaines actions. Ca permet de factoriser le code de manière à ne pas se soucier de ces problèmes dans plusieurs endroits du code en même temps.

Comme pour les modèles, les controlleurs peuvent définir certaines méthodes qui seront appellées automatiquement par CakePHP au moment opportun. La méthode "*beforeFilter*", par exemple, est appellée avant que l'action associée à la requête ne soit executée. Il peut donc être très intéressant d'y définir les actions autorisées selon le niveau de privilèges de l'utilisateur, selon qu'il soit authentifié ou non, etc...

Les controlleurs peuvent inclure certaines dépendances, notament envers des classes du modèle, ou vers des helpers qui pourraient être utilisés par la vue. Le controlleur aura donc accès à la couche DAO des différents modèles qu'elle gère. Chacune de ses méthodes correspondront généralement à l'une des vues qu'elle peut offrir, ceci dit une action X peut tout à fait afficher une vue Y. Le controlleur a aussi accès à la variable Session (*$this->Session*) dans lequel il peut écrire, ou lire, mais aussi à la variable Auth (*$this->Auth*) afin de vérifier que l'utilisateur est bien authentifié.

Dans mon application, j'ai définit une dépendance vers Auth dans mon controlleur AppController, ce qui fait que par défaut, toutes les actions requièrent une authentification de la part de l'utilisateur. C'est seulement dans les controlleurs que j'ai levé cette obligation, via la méthode beforeFilter (*$this->Auth->allow("*")*). Le controlleur passe des données à la vue via les méthodes *$this->set(clé, valeur)*. La variable est ensuite directement accessible depuis la vue depuis la variable *$clé*.

Listing 2 – Exemple controlleur

```
1  class CatalogController extends AppController {
2    var $name = 'Catalog';
3    var $uses = array('Product', 'User');
4    var $helpers = array( 'Ajax', 'Html', 'Javascript' );
5
6    function index() {
7      $this->set('products',
8        $this->Product->find('all'));
9
10     if( $this->Auth->user() ) {
11       $userWithProducts = $this->User->findById(
```

```
12            $this->Auth->user('id')
13          );
14          $this->set('card',$userWithProducts['Product']);
15        }
16      }
17      function search() {
18        if( !empty($this->data['term']) ) {
19          $term = '%'.$this->data['term'].'%';
20
21          $conditions = array("OR" =>  array(
22            'Product.name LIKE' => $term,
23            'Product.description LIKE' => $term ));
24
25          $this->set('products',
26            $this->Product->find('all', array(
27              'conditions' => $conditions)));
28
29          if( $this->Auth->user() ) {
30            $userWithProducts = $this->User->findById(
31              $this->Auth->user('id'));
32            $this->set('card',
33              $userWithProducts['Product']);
34          }
35          $this->render('index');
36        } else {
37          $this->redirect(array( 'action' => 'index'));
38        }
39      }
40      function beforeFilter() {
41        $this->Auth->allow("*");
42        parent::beforeFilter();
43      }
44  }
```

# 5 Vue

La vue est l'endroit où nous allons donc définir la manière dont on va afficher les données du modèle à l'utilisateur. CakePHP permet aux vues d'accéder aux variables passées par le controlleur en créant un namespace virtuel dans la vue. La vue a donc un accès privilégié aux données qui lui sont passées, et y accède directement dans son scope, sans que les autres vues qui pourraient être affichées ne soient obstruées par ces données.

CakePHP catégorise en 3 grandes partie la hiérarchie visuelle :

**Layout** Un squelette appliqué à toutes les vues. Définis dans *app/views/layouts*

**Vue** Le template qui va récupérer les données du controlleur. Définies dans */app/views/{controlleur correspondant}*

**Élément** Utilisés par la vue pour afficher certaines parties qu'elle ne sait ou ne doit pas gérer. La vue peut ou pas passer des données à chaque élément qu'elle veut afficher. Définis dans */app/views/elements/* et sont accessibles d'utilisation à toutes les vues.

Listing 3 – Exemple vue

```
1 <?
2   echo $this->Html->script(array(
3     'jquery-1.4.3.min',
4     'jquery.fancybox-1.3.4',
5     'image-popup'));
6   echo $this->Html->css(array(
7     'catalog',
8     'jquery.fancybox-1.3.4'));
9 ?>
10 <div class="catalog">
11   <?
12       echo $form->create(false, array(
13         'action' => 'search'));
14       echo $form->input('term', array(
15         'label' => false,
16         'div' => false));
17       echo $form->end('Search');
18   ?>
```

```
19   <table>
20   <?php foreach( $products as $product ) {
21     $p = $product['Product'];
22   ?>
23
24     <tr>
25       <td width="150">
26         <a class="gallery-img" href="<?=$p['image']?>">
27           <img src="<?=$p['image']?>"
28                alt="<?=$p['name']?>"
29                height="100">
30         </a>
31       </td>
32       <td>
33         <h3 style="display: inline;">
34           <?=$p['name'];?>
35         </h3>
36         <h5><?=$p['price'];?></h5>
37         <br />
38         <p><?=$p['description'];?></p>
39         <?
40           echo $html->link('Add to card',
41             "/card/add/".$p['id']);
42         ?>
43       </td>
44     </tr>
45   <? } ?>
46   </table>
47 </div>
48 <? if( isset($card) && count($card) ) { ?>
49   <div class="inline-card">
50     <? echo $this->element('inlinecard', array(
51       'card' => $card)) ?>
52   </div>
53 <? } ?>
```

# Références

[1] *Le modèle MVC*, Wikipedia
http ://en.wikipedia.org/wiki/Model–view–controller

[2] *CakePHP*
http ://cakephp.org/

[3] *Manuel de CakePHP*
http ://book.cakephp.org/