Portál pre odborné publikovanie ISSN 1338-0087

CakePHP framework

Morvay Tomáš · Informačné technológie, Študentské práce

07.01.2011

CakePHP je framework na rýchly vývoj webových stránok a aplikácií, ktorý využíva PHP. Poskytuje rozšíriteľnú architektúru na vývoj, údržbu a nasedenie pričom sa snaží vývojarom šetriť čas a umožňuje písať menej kódu. Využíva dobre známe postupy ako je MVC a ORM. Medzi stránky a aplikácie využivajúte tento framework patria napríklad stránka Miss Universe alebo doplnky pre Mozillu.

1. História

Začiatky CakePHP siahajú do roku 2005, kedy Michal Tatarynowicz vytvoril framework nazvaný Cake. V tom čase rástla obľúbenosť frameworku Ruby on Rail, ktorý inšpiroval svojou koncepciou Michala Tatarynowicza pri tvorbe konceptu CakePHP. CakePHP nie je framework, ktorý je odvodený od Ruby on Rails, ale iba prevzal od neho najužitočnejšie koncepty. CakePHP predstavuje jeden z najkomplexnejších open source frameworkov pre vývoj webových aplikácií. Samotný framework je vytvorený v jazyku PHP, pričom konceptuálne nadväzuje na Ruby on Rails. CakePHP je distibuovaný pod licenciou MIT.

Jedným z najhlavnejších pilierov fungovania CakePHP je používanie konvencií, ktoré predstavujú pravidlá pri vytváraní modelov,views a controllerov. Konvencie predstavujú formálne pravidlá pri pomenovaní súborov, tried a ich umiestnení v adresárovej štruktúre. Konvencie sa používajú pri vytváraní súborov, tried, tabuliek v databáze atď. Ako príklad je možné uviesť konvencie pre model a pre tabuľku v databáze, v ktorej sú uložené informácie o používateľoch. Model , uložený v súbore user.php, obsahuje triedu User, pričom v databáze sa nachádza tabuľka users. Vďaka konvenciam aplikácia automaticky použije tabuľku users pre model User. Samozrejme, je možné aj manuálne priradiť určitú tabuľku k danému modelu.

Pri dodržaní konvencí je zabezpečené určitým spôsobom automatická funkčnosť aplikácie, pričom je samozrejme možné vykonať zmeny použitím určitých metód. Pri použití konvencí prichádza k podstatnému šetreniu času, pretože nie je nutné manuálne písať SQL dotazy. Webové aplikácie majú určité spoločné činnosti, ktoré sa opakujú pri prakticky všetkých aplikáciách, preto sa CakePHP snaží zautomatizovať tieto operacie, aby sa zrýchlil čas vývoja aplikácie. Ako príklad je možné uviesť prácu s databázou a formulármi.

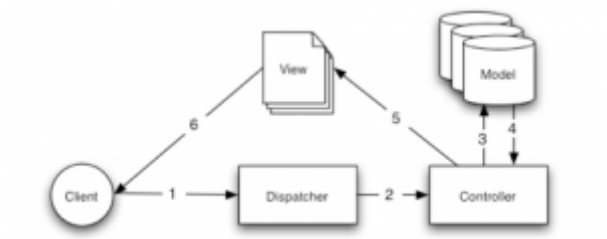
2. Inštalácia

Základnou požiadavkou pre inštaláciu CakePHP je http server, ktorým môže byť napríklad aj Apache server. Optimálne je pre server povoliť atribút mod_rewrite. Ďalšou základnou požiadavkou je PHP vo verzií 4.3.2 alebo vo vyššej. Najnovšiu verziu CakePHP je možné získať z oficiálnej stránky projektu (http://www.cakephp.org) alebo z repozitáru Git. Inštaláciu je potrebné uskutočniť do dokumentového root priečinka na serveri. Inštaláciu je možné vykonať jednoduchú alebo komplexnú. Pri jednoduchej inštalácií obsahuje aplikácia knižnicu CakePHP súborov, pri pokročilej je možná konfigurácia, aby viacero aplikácií využívalo spoločnú knižnicu CakePHP, ktorá je umiestnená mimo daných aplikácií.

3. MVC architektúra

Architektúra CakePHP je postavená na návrhovom vzore MVC. Skratka MVC predstavuje tri hlavné časti aplikácie: model, view a controller. Aplikácia vytvorená vo frameworku CakePHP musí striktne dodržiavať návrhový vzor MVC, pretože samotné jadro frameworku je založené taktiež na návrhovom vzore MVC. Preto nie je možné používať iný návrhový vzor ako je MVC.

- Model predstavuje aplikačné dáta
- View predstavuje samotné zobrazenie dát
- Controller získava požiadavky od klienta a riadni beh aplikácie



Obr. 1 princíp činnosti architektúry

Obrázok č.1 predstavuje request od klienta a jeho spracovanie. Obrázok predstavuje modelovú situáciu, keď si používateľ klikol na „Buy A Custom Cake Now!“ na web stránke.

1. Používateľ klikol na odkaz http://www.example.com/cakes/buy a následne prehliadač vykonal požiadavku (request) na server.
2. Dispatcher skontroloval požadovanú URL a následne pridelil požiadavku konkrétnemu controlleru
3. Controller vykoná špecifické operácie na úrovni aplikačnej logiky. Ako príklad môžeme ilustrovať kontrolu, či je používateľ prihlásený.
4. Controller sa používa aj na prístup aplikačných dát. Modely sú najčastejšie reprezentované tabuľkami v databáze, súbormi, LDAP entitami, RSS kanálmi a mnohými ďalšími. V ilustrovanom príklade controller používa model na získanie dát z databázy posledného nákupu používateľa.
5. Controller posiela dáta do ďalšej časti aplikácie- view. View slúži na samotné zobrazenie dát, najčastejším formátom na zobrazenie dát je HTML a medzi často používané formáty môžeme uviesť tiež PDF, XML alebo mnohé iné.
6. Dáta vytvorené view sú poslané ako odpoveď (response) pre používateľov prehliadač.

Návrhový vzor MVC poskytuje vývojárom obrovské výhody. MVC vytvára modulárnu štruktúru aplikácie, ktorá je jednoducho spravovateľná. Modulárna štruktúra poskytuje neoceniteľné výhody pri vývoji aplikácie, pretože umožňuje nezávislé vytváranie modulov, ich správu a zmenu. Návrhový vzor MVC súčasne umožňuje pracovať programátorom a grafikom a zároveň umožňuje rýchle prototypovanie.

4. Praktická časť aplikácie

Prvým krokom pri použití CakePHP je samotná inštalácia. Pri jednoduchej inštalácií je CakePHP potrebné nainštalovať do dokumentového adresáru (document root) serveru a podľa potreby nakonfigurovať súbor php.ini. Nasledujúcim krokom je vytvorenie databázy, v ktorej budú uložené informácie o článkoch (post). Názov databázy nastavíme ako cake_blog a vytvoríme tabuľku posts. Tabuľku posts vytvoríme nasledujúcim SQL dotazom:

```
CREATE TABLE posts (
id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
title VARCHAR(50),
body TEXT,
created DATETIME DEFAULT NULL,
modified DATETIME DEFAULT NULL
);
```

Tabuľku je vhodné naplniť aspoň tromi testovacími položkami, aby sme boli schopný otestovať funkčnosť aplikácie počas vývoja. Výber názvu tabuľky a jej stĺpcov nie je náhodný, ale riadi sa podla konvencií CakePHP. Konvencie v CakePHP predstavujú pomerne širokú tému, preto v práci nie sú presne definované, ale len okrajovo spomenuté.

Po úspešnom vytvorení databázy našej aplikácie, tabuľky posts a jej naplnením je nutné prepojiť databázu s aplikáciou. Vytvorenie spojenia medzi aplikáciou a databázou je uskutočnené pomocou konfiguračného súbora. Kópia konfiguračného súbora sa nachádza v /app/config/database.php.default. Je potrebné vytvoriť súbor database.php, ktorý umiestnime do rovnakého adresára v akom sa nachádza database.php.default. Obsah súboru database.php.default prekopírujeme do database.php a zmeníme hodnoty premennej $default podľa aktuálneho nastavenia databázy a súbor uložíme.

```
var $default = array(
'driver' => 'mysql',
'persistent' => false,
'host' => 'localhost',
'login' => 'admin',
'password' => 'test85',
'database' => 'cake_blog',
'prefix' => '',
);
```

Pri správnom postupe a pri korektných nastaveniach premennej $default by mala byť

databáza prepojená s aplikáciou.

4.1 Vytvorenie modelu Post

Model predstavuje v architektúre návrhového vzoru MVC najdôležitejšiu časť, pretože zabezpečujú interakciu aplikácie s dátami. Model pre tabuľku posts je uložený v súbore post.php, ktorý uložíme do adresára /app/models. Samozrejmosťou je použite konvencií, ktoré sa v tomto prípade týkaju názvu súboru, ktorý má rovnaký názov ako model, ale s malým začiatočným písmenom. Názov modelu tvorí singulár názvu tabuľky v databáze, ktorá je v plurále. V našom prípade je názov modelu Post a tabuľky posts.

```
<?php
class Post extends AppModel
{
var $name = 'Post';
}
?>
```

4.2 Vytvorenie controlleru pre Post

Controller predstavuje aplikačnú logiku aplikácie. Vykonáva operácie s dátami poskytnutými modelom a spracované dáta odosiela na zobrazenie. Najčastejšie sa používa odoslanie dát do grafickej šablóny, reprezentovanej pomocou poslednej časti MVC- view. Controller najčastejšie obsahuje aj metódy, ktoré predstavujú akcie pre jednotlivý controller. V našom prípade controller obsahuje metódu index, ktorá zobrazí všetky riadky tabuľky posts, to znamená všetky články.

Ako dokáže aplikácia rozoznať akú akciu má vykonať? Jednoducho, pomocou kontroly URL adresy. Štandartná URL adresa aplikácie vytvorenej v CakePHP má štruktúru www.example.com/controller/akcia/parameter/. V praxi to znamená, že URL adresa www.example.com/posts/edit/3 vykoná metódu edit s parametrom 3 z controlleru posts

```
<?php
class PostsController extends AppController
{
var $name = 'Posts';
function index() {
$this->set('posts', $this->Post->find('all'));
}
}
?>
```

Podstatou metódy index je príkaz $this->set(‘posts’, $this->Post->find(‘all’)) . Príkaz načíta z tabuľky všetky záznamy a uloží ich do premennej posts, vďaka ktorej bude možné pristupovať k dátam v zobrazení (view) pre danú metódu. Controller je uložený v súbore posts_controller.php, a v adresári /app/controllers.

4.3 Zobrazenie dát- view

View je časť návrhového vzoru MVC, ktorá slúži na grafickú reprezentáciu dát, získaných z controlleru. Potrebné dáta sú v našom prípade uložené v premennej $posts, ktorá predstavuje asociatívne pole, vďaka čomu je možné jednoducho pristupovať k jednotlivým dátam. Na obrázku je zobrazená štruktúra premennej $posts.

```
// print_r($posts) output:

Array
(
    [0] => Array
        (
            [Post] => Array
                (
                    [id] => 1
                    [title] => The title
                    [body] => This is the post body.
                    [created] => 2008-02-13 18:34:55
                    [modified] =>
                )
        )
    [1] => Array
        (
            [Post] => Array
                (
                    [id] => 2
                    [title] => A title once again
                    [body] => And the post body follows.
                    [created] => 2008-02-13 18:34:56
                    [modified] =>
                )
        )
    [2] => Array
        (
            [Post] => Array
                (
                    [id] => 3
                    [title] => Title strikes back
                    [body] => This is really exciting! Not.
                    [created] => 2008-02-13 18:34:57
                    [modified] =>
                )
        )
)
```

Obr. 2 Štruktúra premennej $posts

Samotnú šablónu je treba uložiť do adresáru /app/views, kde je potrebné vytvoriť adresár s názvom plurálu názvu modelu v angličtine. To znamená že musíme vytvoriť adresár posts. Do adresára uložíme súbor index.ctp, pričom je nutné dodržať konvenciu, aby bol názov súboru totožný s názvom akcie controllera, pre ktorú je šablóna vytvorená. V našom prípade vytvárame šablónu pre akciu index.

```
<!-- File: /app/views/posts/index.ctp -->
<h1>Blog posts</h1>
<table>
<tr>
<th>Id</th>
<th>Title</th>
<th>Created</th>
</tr>
<!—cyklus ktorý prečíta všetky prvky pola uloženého v premennej $posts a zobrazí ich -->
<?php foreach ($posts as $post): ?>
<tr>
<td><?php echo $post['Post']['id']; ?></td>
<td><?php echo $this->Html->link($post['Post']['title'], array('controller' => 'posts', 'action' => 'view', $post['Post']['id'])); ?>
</td>
<td><?php echo $post['Post']['created']; ?></td>
</tr>
<?php endforeach; ?>
```

```
</table>
```

Vytvorená šablóna zobrazí zoznam článkov z tabuľky posts, pričom pri kliknutí na názov je možné si zvolený článok zobraziť. Zobrazenie článkov bude zabezpečené pomocou akcie view, ktorú vložíme do posts controlleru.

```
function view($id = null)
{
$this->Post->id = $id;
$this->set('post', $this->Post->read());
}
```

Použitie funkcie read() nám vráti len jeden výsledok na rozdiel od find(). Parameter predstavuje id článku, ktorý sa načíta do premennej post. Dáta uložené v premennej $post sa zobrazia podľa šablóny definovanej pre akciu view. Samotnú šablónu je treba uložiť do adresáru /app/views/posts v súbore view.ctp.

4.4 Pridávanie článkov

Pridávanie článkov predstavuje vkladanie dát do databázy. Pridávanie je realizované pomocou akcie add controlleru posts. URL adresa akcie na priávanie má tvar www.example.com/posts/add. Metóda add() má tvar:

```
function add()
{ if (!empty($this->data)) {
if ($this->Post->save($this->data))
{
$this->Session->setFlash('Your post has been saved.');
$this->redirect(array('action' => 'index'));
}
}
}
```

Po vykonaní metódy add sú dáta z formuláru uložené v premennej $this->data, ktorá predstavuje asociatívne pole. Ak sú nie sú vložené žiadne dáta, nevykoná sa žiadna operácia, znova sa načíta prázdny formulár. Ak premenná $this->data obsahuje nejaké dáta, tak sa dáta uložia pomocou Post modelu a zobrazí sa hlásenie o úspešnej operácií. Hlásenie je realizované pomocou komponentu Session, ktorého použitie je potrebné definovať v controllere pomocou zadefinovania premennej $components: var $components = array(‘Session’);.

Pomocou metódy $this->Session->setFlash(‘Your post has been saved.’) odošleme do session premennej text ‘Your post has been saved.’ V šablóne je potom zobrazený obsah premennej. Metódu add(), takisto ako aj definíciu premennej $components vložíme do súboru posts_controller.

Pridávanie dát je realizované pomocou formuláru, ktorý po odoslaní vykoná metódu add. Formulár predstavuje súbor šablóny pre metódu add, preto súbor uložíme ako add.ctp do priečinku posts

```
<!--Súbor: /app/views/posts/add.ctp -->
<h1>Add Post</h1>
<?php
echo $this->Form->create('Post');
echo $this->Form->input('title');
echo $this->Form->input('body', array('rows' => '3'));
echo $this->Form->end('Save Post');
?>
```

Formulár aj jednotlivé elementy formuláru su vytvorené pomocou FormHelperu, vďaka ktorému je tvorba html kódu podstatne zjednodušená a prehľadnejšia. Pomocou metódy $this->Form->create('Post') je vytvorený formulár pre model Post. Jednotlivé elementy sú vytvárané pomocou metódy $this->Form->input(), pričom prvý parameter metódy určuje, ktorému stĺpcu v modeli (tabuľke v databáze) patrí /prestylizovat/. Metóda $this->Form->end('Save Post') vytvorí vo formulári button sumbmit s textom Save Post. Ako už bolo spomenuté, k metóde add() je možné pristupovať pomocou URL adresy www.example.com/posts/add alebo vytvorením odkazu pomocou HTML helperu v šablóne index.ctp. Odkaz vytvoríme vložením nasledujúceho kódu:

```
<?php echo $this->Html->link('Add Post', array('controller'
=> 'posts', 'action' => 'add')); ?>
```

Pri vkladaní dát je dôležitá validácia vstupných dát, aby sme zabránili vkladaniu nekorektných dát. Validácia dát je v CakePHP realizovaná pomocou vloženia validačných pravidiel v modeli. Pravidlá sú v našom prípade definované pre jednotlivé elementy formuláru v premennej $validate. Model s nadefinovanými pravidlami má nasledovný tvar:

```
<?php class Post extends AppModel
{
var $name = 'Post';
var $validate = array(
'title' => array( 'rule' => 'notEmpty' ),
'body' => array( 'rule' => 'notEmpty' )
);
}
?>
```

Implementácia vložených pravidiel spôsobí skutočnosť, že budú akceptované iba dáta z formuláru, ktoré obsahujú neprázdne hodnoty elementov 'title' a 'body'. CakePHP poskytuje širokú paletu pravidiel pre validáciu rôznych údajov. V prípade zadania nekorektných dát sa automaticky zobrazí chybové hlásenie.

4.5 Odstránenie článkov

Odstránenie článkov je ralizované pomocou metódy delete(id). Metóda obsahuje jeden parameter, ktorý predstavuje id článku, ktorý sa má zmazať. Metóda je volaná pomocou URL adresy v tvare www.example.com/posts/delete/id. Pre metódu nie je potrebné vytvoriť view, pretože hneď po odstránení článku prichádza k presmerovaniu

na stránku index pomocou metódy $this->redirect(array(‘action’ => ‘index’)). Odstránenie je ralizované pomocou kliknutia na odkaz delete. Upravená verzia súboru, kde je pridaný link na odstránenenie bude popísaný ďalej v článku.

```
function delete($id)
{ if ($this->Post->delete($id)) {
$this->Session->setFlash('The post with id: ' . $id . ' has
been  deleted.');  $this->redirect(array('action'  =>
'index'));
}
}
```

4.6 Editácia článkov

Editácia článkov je ralizovaná pomocou metódy edit(id). Metóda obsahuje jeden parameter, ktorý určuje id článku, ktorý sa bude editovať. Metóda je volaná pomocou URL adresy www.example.com/posts/edit/id. Metódu edit je potrebné vložiť do controlleru pre Post, do súboru post_controller.php . Metóda na začiatku skotroluje, či premenná $this->data obsahuje nejaké dáta.

Ak nie, znamená to že nebol odoslaný formulár s editovanými údajmi a tým pádom je potrebné do formuláru načítať dáta článku, ktorý sa upravuje. Načítanie je automatická činnosť, ktorá ale závisí na korektnom použití konvencí. Ak premenná $this->data obsahuje dáta, tak potom nastala situácia keď bol odoslaný obsah formuláru a prichádza k update záznamu v databáze, pre konkrétny článok.

```
function edit($id = null)
{ $this->Post->id = $id;
if (empty($this->data)) { $this->data = $this->Post->read();
}
else { if ($this->Post->save($this->data)) {
$this->Session->setFlash('Your post has been updated.');
$this->redirect(array('action' =>'index'));
}
}
}
```

Pre metódu edit() je potrebné vytvoriť šablónu edit.ctp, ktorú je potrebné vložiť do adresára

```
<!-- Súbor: /app/views/posts/edit.ctp -->
<h1>Edit Post</h1>
<?php echo $this->Form->create('Post', array('action' =>
'edit'));
echo $this->Form->input('title');
echo $this->Form->input('body', array('rows' => '3'));
echo $this->Form->input('id', array('type' => 'hidden'));
echo $this->Form->end('Save Post');
?>
```

Použité zdroje literatúry

1. Beginning With CakePHP, http://book.cakephp.org/view/879/Beginning-With-CakePHP
2. Blog, http://book.cakephp.org/view/1528/Blog
3. Tvoríme CMS s CakePHP – úvod, http://ims.rockretail.com/2007/12/12/1-tvorime-cms-s-cakephp-uvod/