CakePHP

The Rapid Development PHP Framework

By Khalid Alharbi

Contents

Contents Cont.

What is CakePHP ?

- According to the official CakePHP website, cakephp.org,

  “CakePHP is a rapid development framework for PHP that provides an extensible architecture for developing, maintaining, and deploying applications. Using commonly known design patterns like MVC and ORM within the convention over configuration paradigm, CakePHP reduces development costs and helps developers write less code”

History

- CakePHP was created by Michal Tatarynowicz On April 15, 2005.
- Inspired by Roby on Rails.
- Michal published it under MIT licence and opened it to the community developers.
- In July 2005, Larry E Masters (aka PhpNut) took over as the lead developer.
- In December 2005, Larry and Garrett J. Woodworth (aka gwoo) founded the Cake Software Foundation to promote development related to CakePHP.

Features

- It’s PHP! Compatibles with versions 4 and 5.
- Open Source, MIT license.
- Object Oriented.
- Design Patterns: MVC and ORM.
- Convention over configuration.
- Framework not set of libraries.
- Bake Script: Automates CRUD scripting.

Features Cont.

- Scaffolding: single line view rendering.
- Helpers: standard HTML, Ajax, and JavaScript helpers to create views.
- Customizable Elements: add elements as plugins into the application.
- Input validation and data sanitization tools to help create secure application.
- Search engine friendly URLs.
- Large growing active community.
- Extremely simple – It’s a piece of Cake!

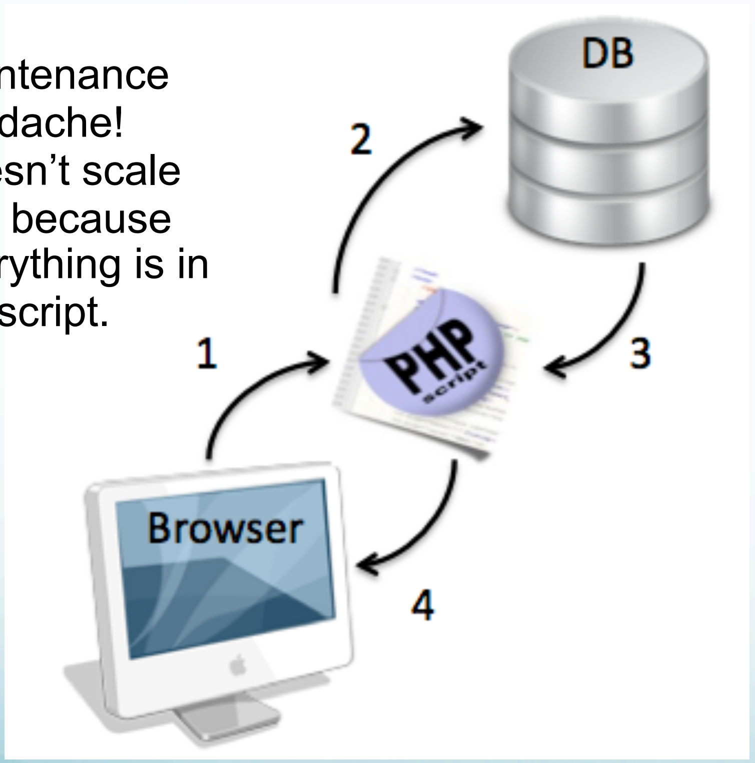
MVC Pattern

- Cake enforces Model View Controller (MVC) Pattern.
- Cake splits operations into three parts:
  - Models: used for all database interactions.
  - Views: used for all output and displays.
  - Controllers: used to control the application flow.

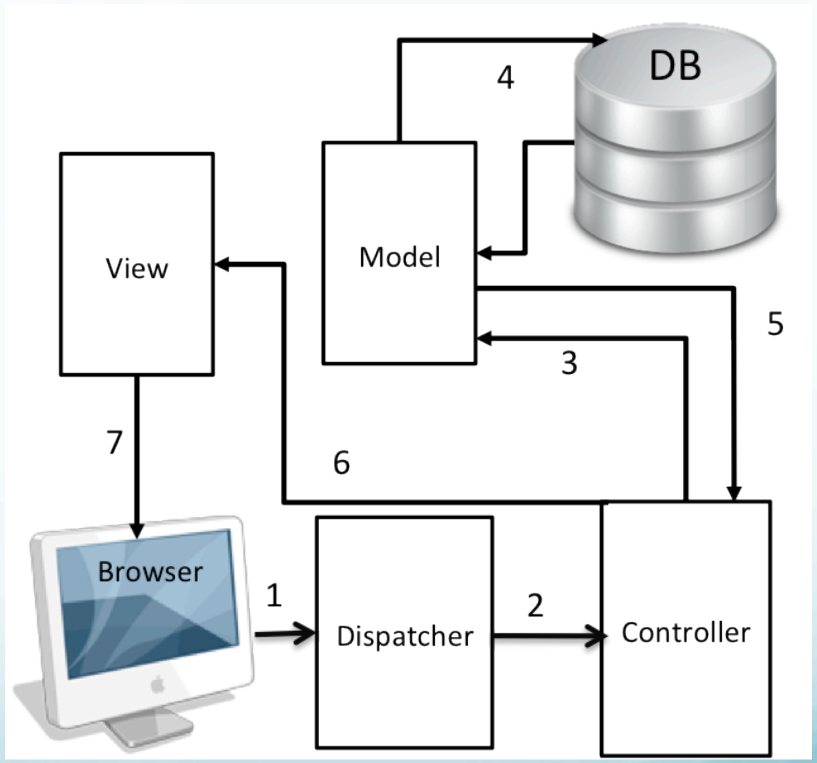
How Does MVC Work? (I)

No MVC PHP App

Maintenance headache! Doesn't scale well because everything is in the script.



CakePHP MVC App



How Does MVC Work? (II)

① The client sends request.

According to the CakePHP convention, the URL looks like:

http://{Domain} /{Application}/{Controller}/{Action}/{Parameter 1, etc.}

② The dispatcher script parses the URL, determines which controller to execute, and forwards the request to the Controller.

③ The action (method) in the controller needs to access data, so it sends a database request to the Model.

④ The model executes the database request, pulls the output,

⑤ The Model sends the output to the Controller.

⑥ The Controller sends the output to the corresponding view.

⑦ The View adds any design element to the output and sends it to the client's browser.

Why CakePHP MVC ?

- ❑ Reduce redundancy.
- ❑ Organize the different tasks of the web app.
- ❑ No need for include statements in PHP scripts.
- ❑ Follow Agile techniques.
- ❑ Easy to debug and maintain.

Cake Naming Convention (I)

- Cake adheres the idea of "convention over configuration."
- This organizes the operations of the web application.
- It's strongly recommended to follow Cake's naming convention.
- More than one word in the name, must be separated by _ when naming the file and camel cases when naming the class.

Cake Naming Convention (II)

- DB Table : Plural form in lowercase letters.
  ex: orders.

- The MVC parts must be named as the following:

	File Name	Class Name	Base Class Name	Location
Model	Singular form of the table name with .php extension ex: order.php	The file name in a Camel case. ex: Order	AppModel	/app/models
Controller	tablename_controller with .php extension. ex: orders_controller.php	The table name appended Controller. ex: OrdersController	AppController	/app/controllers
View	The action name in the controller with .ctp extension. ex: add.ctp	No classes; a view contains only html tags and php scripts.		/app/views/controllerName

Core Concepts

- Models and Associations.
- Controllers.
- Views.

Models and Association (I)

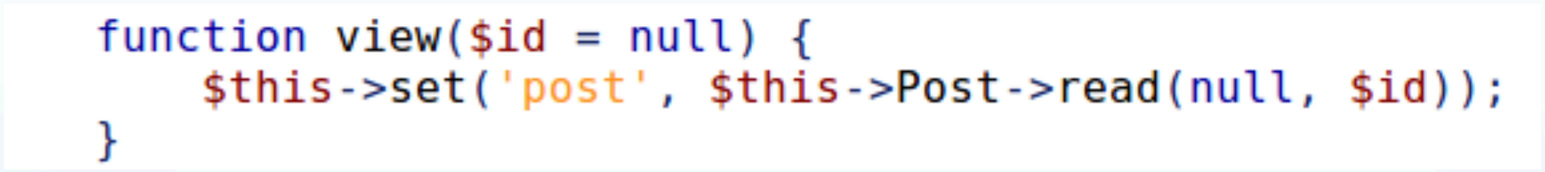
- The model is the access point to a table in the DB.
- The model can contain data validation rules and association information.
- Cake offers Object Relational Mapping (ORM) to handle database relationships.
- Using Cake's ORM, CRUD operations are done without writing complex SQL queries.
- Relationships between tables are defined through association.

Models and Association (II)

- Association helps cake link between models through relational mapping.
- Association types:
  - hasOne: one-to-one relationship. ex: user “has one” account.
  - hasMany: a user “has many” posts.
  - belongsTo: a post “belongs to” one user.
  - hasAndBelongsToMany: a post “has and belongs to many” tags.

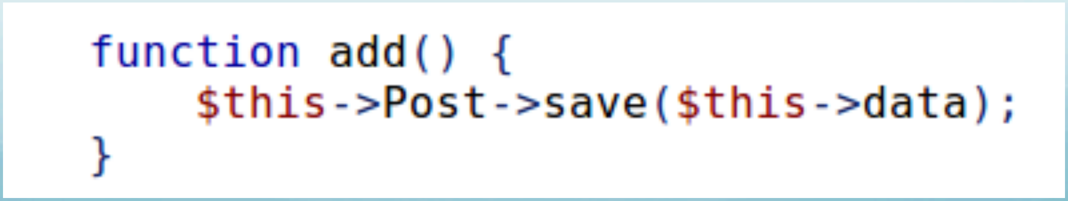
Controllers

- Controllers handle all logic and requests.
- Use functions like set(), and render() to call out the views and provide it with variables.
- Use read(), and find() model's methods to get a list of fields from the database.



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

- Use the standardized array `$this->data` to handle user form data in the view and runs it through the Model.



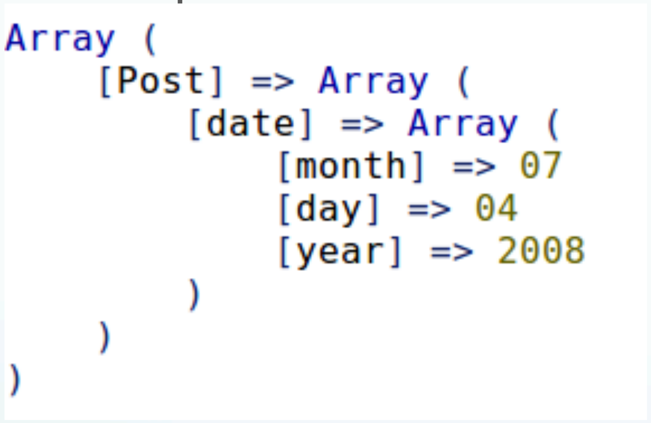
```
function add() {
    $this->Post->save($this->data);
}
```

Views (I)

- Views handle the presentation layer and displays.
- Views contain HTML and PHP scripts.
- Views are associated with actions in the controller.
- When Cake launches an action in the controller, it will automatically render the corresponding view for it or display error.
- If the action in the controller is only called by another action, then there is no need to create a view for it.

Views (II)

- When a user fills out a form, Cake places the form data into the `$this->data` standard array.
- Cake parses the `$this->data` like the following:



```
Array (
    [Post] => Array (
        [date] => Array (
            [month] => 07
            [day] => 04
            [year] => 2008
        )
    )
)
```

- User-submitted data can be pulled from `$this->data` array like any PHP array.
  Ex: `$this->data['Post']['date']['year’]`

Installing CakePHP (I)

- Before beginning, install these on your localhost:
  - Apache server with mod_rewrite.
  - PHP 4.3.2 or greater.
  - MySQL (the default database engine in Cake) or you can use any supported DBMSs: PostgreSQL, Microsoft SQL Server 2000, Firebird, IBM DB2,Oracle, SQLite, ODBC, or ADOdb.
- You can easily install all of these with XAMPP!
- Download the latest stable release version 1.3.8 of Cake from http://cakephp.org.
- Extract the compressed package to your local host root.

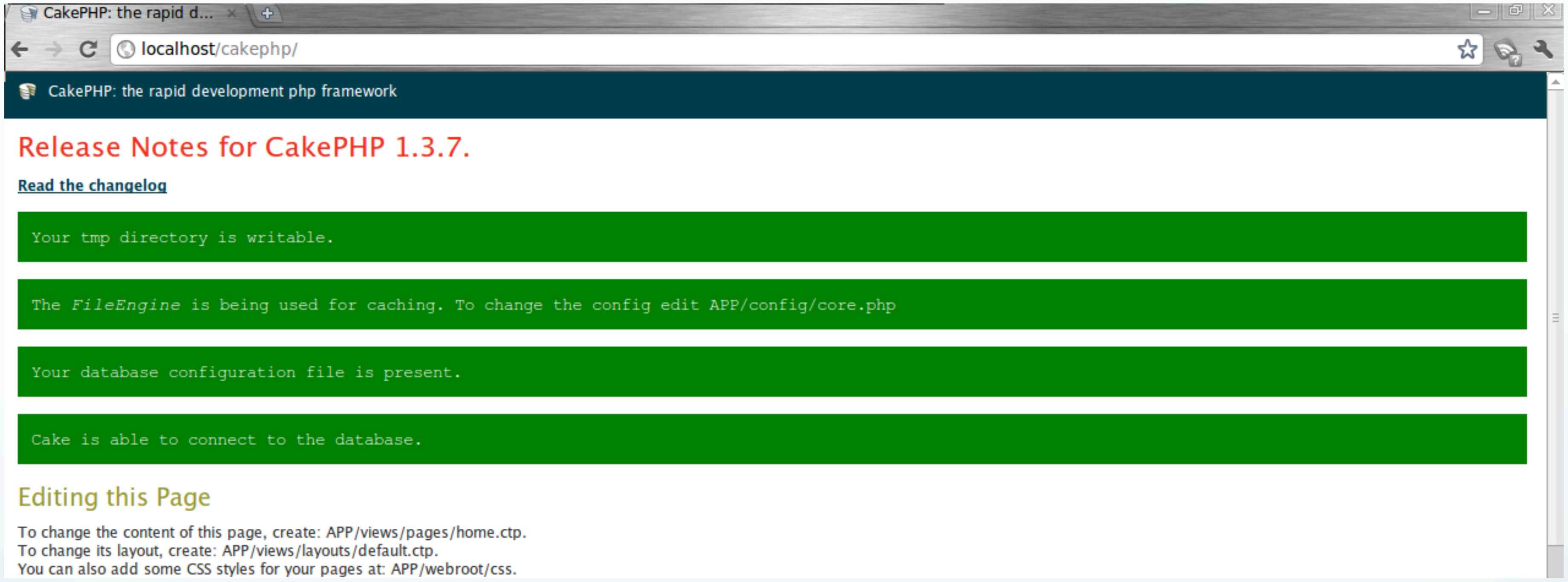
Installing CakePHP (II)

- You will end up with the following directories:
  - */app* contains all controllers, models, views, and anything related to your application.
  - */cake* contains the cake's libraries and scripts.
  - */plugins* contains plugins packages.
  - */vendors* contains other independent Cake scripts.

Running CakePHP

- Before running:
  - Change the /*tmp* folder permission to read and write.
  - Change the security.salt value in *app/config/core.php*
  - Create the database.
  - Rename *app/config/database.php.default* to *database.php* and edit the connection settings.
- Go to http://localhost/cakephp

The Resulting Application



CakePHP: the rapid development php framework

Release Notes for CakePHP 1.3.7.

Read the changelog

```
Your tmp directory is writable.

The FileEngine is being used for caching. To change the config edit APP/config/core.php

Your database configuration file is present.

Cake is able to connect to the database.
```

Editing this Page

To change the content of this page, create: APP/views/pages/home.ctp.
To change its layout, create: APP/views/layouts/default.ctp.
You can also add some CSS styles for your pages at: APP/webroot/css.

The oven is ready!

Let’s put on the cooking hat and bake our first cake ☺

Hello World! (I)

- The required steps:
  - Create the Model: */app/models/example.php*

```
<?php class Example extends AppModel{
    var $useTable = false; // This model does not use a database table
    var $name='Example'; // for backward compatibility with PHP 4
    var $helloWorldvar='Hello World!';
}
?>
```

  - Create the controller: */app/controllers/examples_controller.php*

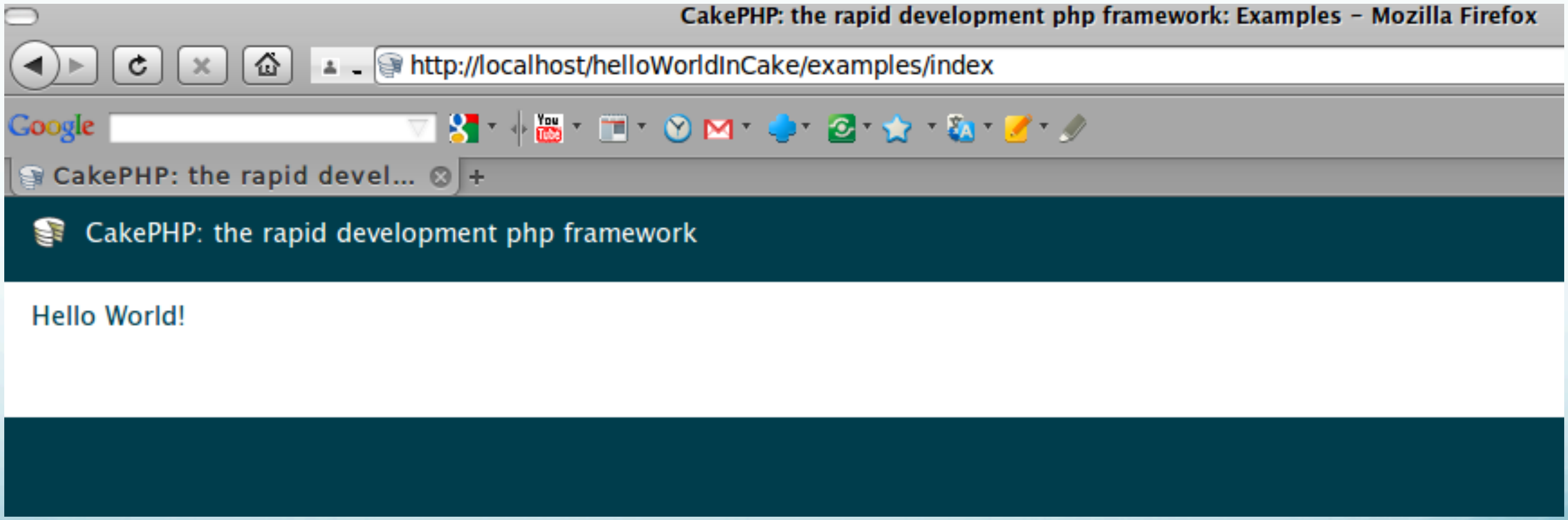
```
<?php
    class ExamplesController extends AppController{
        var $name='Examples';
        function index(){
            $this->set('myvar',$this->Example->helloWorldvar);
        }
    }
```

Hello World! (II)

- Create the view: */app/views/examples/index.ctp*

```
<h1><?php echo $myvar?></h1>
```

- Running the app: navigate to http://localhost/application_name/examples/index



CakePHP: the rapid development php framework: Examples – Mozilla Firefox

http://localhost/helloWorldInCake/examples/index

CakePHP: the rapid devel...

CakePHP: the rapid development php framework

Hello World!

Blog Application (I)

- Creating a blog using CakePHP.
- The required steps:
  1) Create a new cake app in localhost root named blog.
  2) Create the blog's database.
  3) Create five tables: users, posts, comments, tags, posts_tags.
  4) Create the models and specify the association relationship between them.
  5) Create the controllers and test the association using scaffolding.

Create the blog's database (I)

❑ users table:

```
CREATE TABLE `users` ( `id` int(11) unsigned NOT NULL auto_increment,
`name` varchar(100), `email` varchar(150),`firstname` varchar
(60),`lastname` varchar(60),PRIMARY KEY (`id`));
```

❑ posts table:

```
CREATE TABLE `posts` ( `id` int(11) unsigned NOT NULL auto_increment,
`name` varchar(255), `date` datetime, `content` text, `user_id` int(11),
PRIMARY KEY (`id`), FOREIGN KEY(user_id) REFERENCES users(id) ON DELETE
CASCADE ON UPDATE CASCADE );
```

❑ comments table:

```
CREATE TABLE `comments` ( `id` int(11) unsigned NOT NULL auto_increment,
`name` varchar(100), `content` text, `post_id` int(11), PRIMARY KEY
(`id`), FOREIGN KEY(post_id) REFERENCES posts(id) ON DELETE CASCADE ON
UPDATE CASCADE );
```

Create the blog's database (II)

- tags table:

```
CREATE TABLE `tags` ( `id` int(11) unsigned NOT NULL auto_increment,
`name` varchar(100), `longname` varchar(255), PRIMARY KEY (`id`) );
```

- posts_tags table:

```
CREATE TABLE `posts_tags` ( `id` int(11) unsigned NOT NULL
auto_increment, `post_id` int(11) unsigned, `tag_id` int(11) unsigned,
PRIMARY KEY (`id`), FOREIGN KEY(post_id) REFERENCES posts(id) ON DELETE
CASCADE ON UPDATE CASCADE, FOREIGN KEY(tag_id) REFERENCES tags(id) ON
DELETE CASCADE ON UPDATE CASCADE );
```

The naming convention for foreign keys is: [singular table name]_id

Models and Association

- User Model: */app/models/user.php*

```php
<?
class User extends AppModel {
	var $name = 'User';
	var $hasMany = array('Post');
}
?>
```

- Post Model: */app/models/post.php*

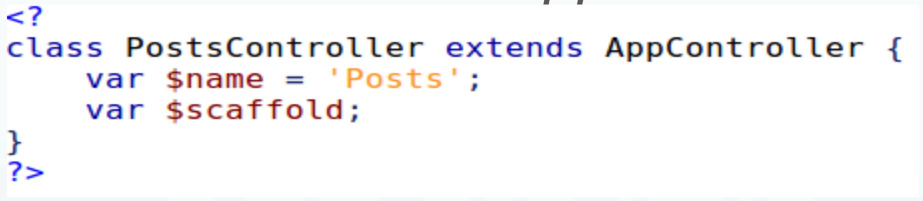
```php
<?
class Post extends AppModel {
    var $name = 'Post';
    var $belongsTo = array('User');
    var $hasAndBelongsToMany = array('Tag');
}
?>
```

- Tag Model: */app/models/tag.php*

```php
<?
class Tag extends AppModel {
	var $name = 'Tag';
	var $hasAndBelongsToMany = array('Post');
}
?>
```

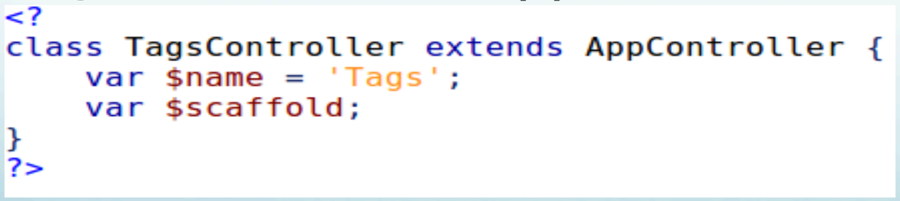
Controllers and Scaffolding

- ❑ Scaffolding is a technique for testing table associations without needing to code any HTML.
- ❑ To add scaffolding to the application, define the $scaffold variable in the controller.
  - ❑ Posts Controller: */app/controllers/posts_controller.php*



```
<?
class PostsController extends AppController {
    var $name = 'Posts';
    var $scaffold;
}
?>
```

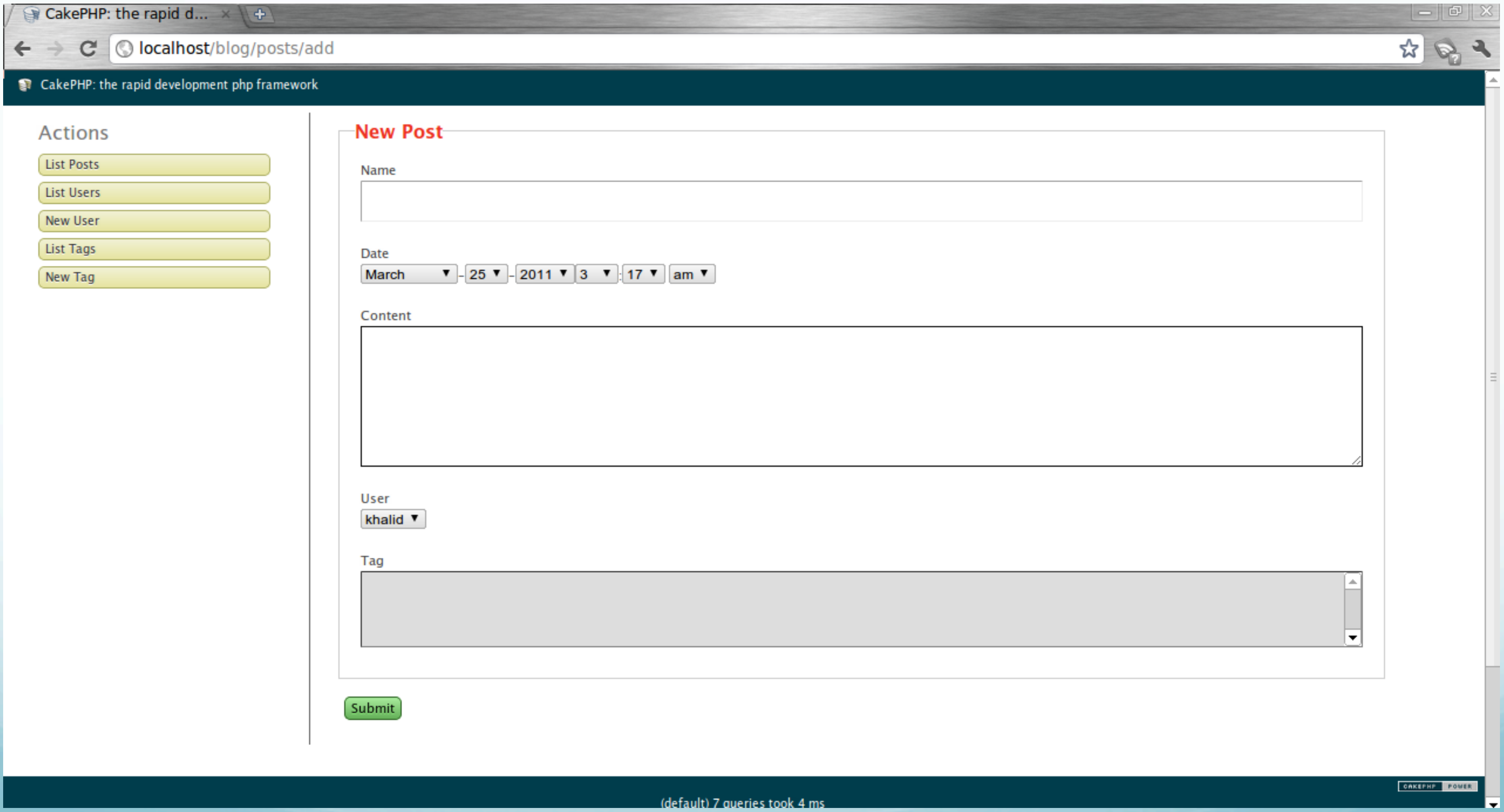
  - ❑ Tags Controller: */app/controllers/posts_controller.php*



```
<?
class TagsController extends AppController {
    var $name = 'Tags';
    var $scaffold;
}
?>
```

The Resulting Application (I)

- Navigate to: http://localhost/blog/posts/add



New Post

Name

Date
March - 25 - 2011 3 : 17 am

Content

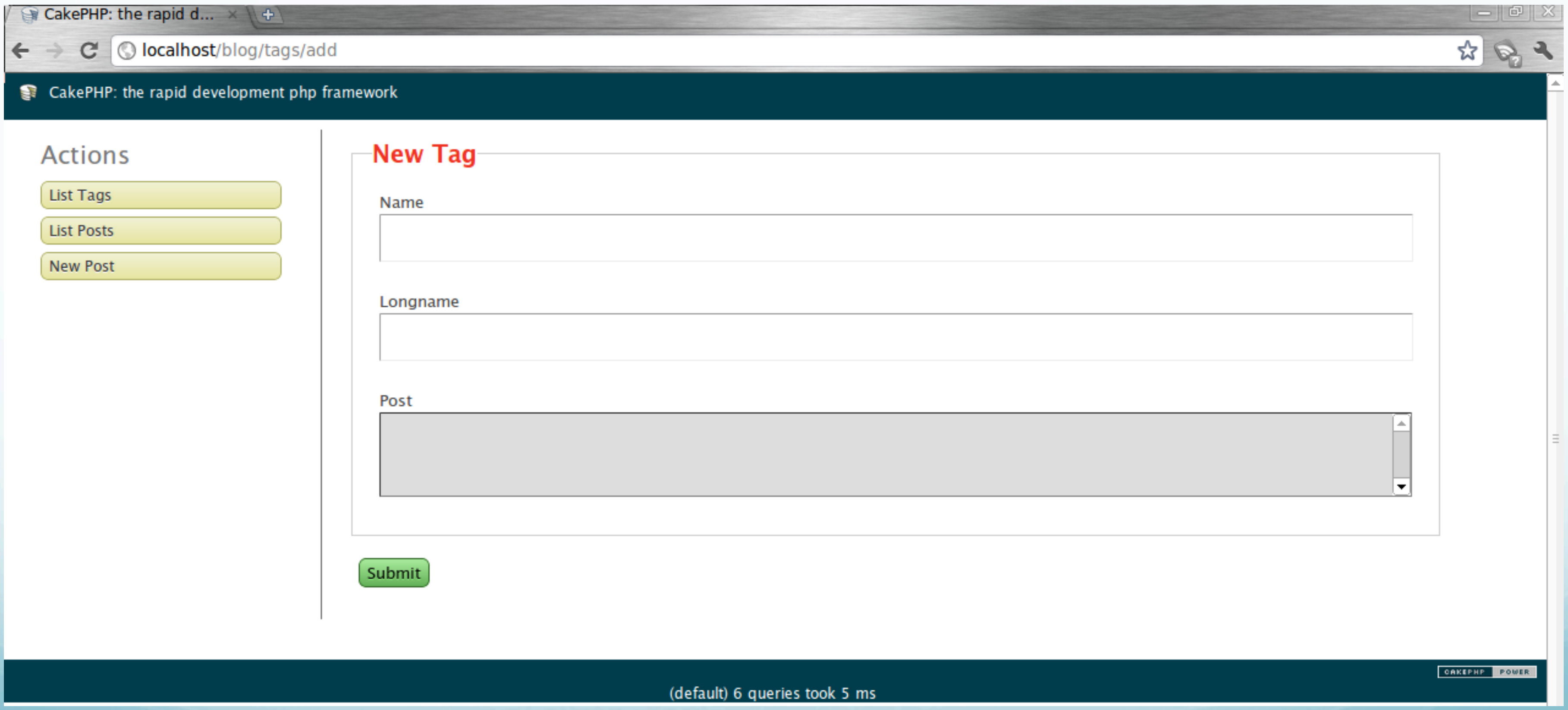
User
khalid

Tag

(default) 7 queries took 4 ms

The Resulting Application (II)

- Navigate to: http://localhost/blog/tags/add



CakePHP: the rapid d...

localhost/blog/tags/add

CakePHP: the rapid development php framework

Actions

- List Tags
- List Posts
- New Post

New Tag

Name

Longname

Post

Submit

(default) 6 queries took 5 ms

The Bake Script

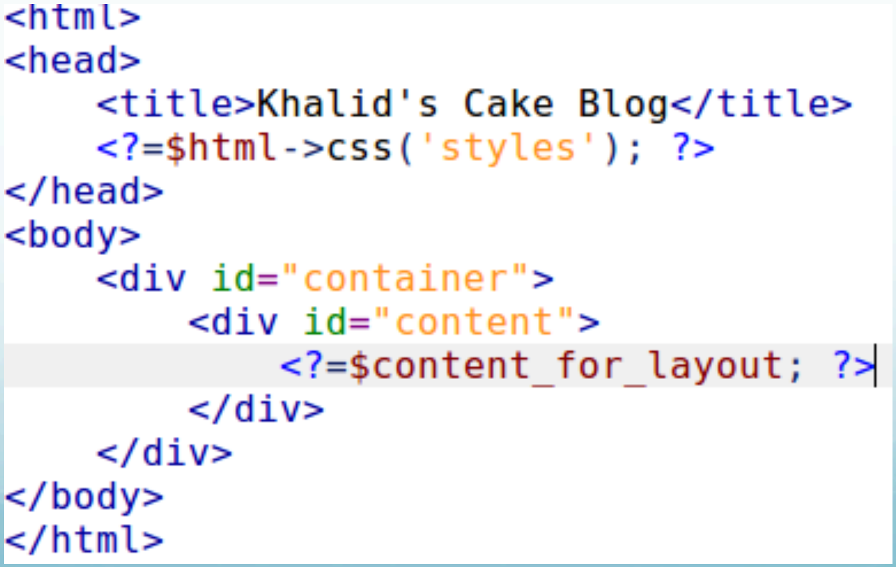
- The bake script is a shell script that automatically generates the basic elements of CakePHP.
- It saves a lot of time generating the needed controllers, models, views, and unit tests.
- It helps understand how MVC works in Cake.
- It is located in *cake/console/libs/bake.php*
- Configuring and using the bake script is illustrated in the following link: http://www.screencast.com/t/Lyr5li6btec

Callback methods

- Helps add special logic before or after actions are executed.
- Cake supports callback methods for both controllers and models.
  - Controller callback methods:
    - **`beforeFilter(), beforeRender(), afterFilter(), etc.`**
  - Model callback methods:
    - **`beforeFind(), afterFind(), beforeValidate(), beforeSave(), beforeDelete(), etc.`**

Layouts

- Contain the overall design that wraps a view.
- Layouts files must be placed in */app/views/layouts* directory and named lowercase with .ctp extension.
- The variable `$content_for_layout` is used to place the code for the views.



```
<html>
<head>
    <title>Khalid's Cake Blog</title>
    <?=$html->css('styles'); ?>
</head>
<body>
    <div id="container">
        <div id="content">
            <?=$content_for_layout; ?>
        </div>
    </div>
</body>
</html>
```

Routes

- Mechanism that maps URLs to controller actions.
- Helps customize URLs and maintain the MVC structure.
- Helps generate non-HTML files such as PDFs, RSS feeds, etc.
- Cake comes configured with a default set of routes:
  - Ex: `Router::connect('/', array('controller' => 'pages', 'action' => 'display', 'home'));`
- Routing configuration file is located in: */app/config/routes.php.*

Extending MVC Structure

- Cake contains resources that extends the MVC structure while maintaining its lightweight organization.
- Cake has the following extensions:
  - Helper: extends the View's functionality.
  - Component: extends the Controller's functionality.
  - Data Source: extends the Model's functionality.
  - Behavior: extends the Model's functionality.

Helpers

- Classes for the presentation layer to extend the functionality of the views.
- Contain the presentation logic shared between views, elements, or layouts.
- Simplify rendering HTML and processing forms.
- CakePHP has a set of built in helpers:
  - HTML, Form, AJAX, Cache, JavaScript, RSS, XML, sessions, Time, Text, etc.
- Ex:

```
<?=$html->link('Add a Post','/posts/add');?>
```

- Ability to create custom helpers.

Components

- Used to extend the functionality of the controllers.
- Components are specific purpose shared actions that can be used across all controllers such as sending email, etc.
- Components are stored in the */app/controllers/components* directory.
- Cake comes with built-in components:
  - Auth, Session, Cookie, Email, Security, etc.
- Ability to create custom controllers.

Data Sources

- Connect to extra data-sources and provide data to the models.
- Models can reference functions in the DataSource without performing extra connections and data handling operations.
- Cake comes with built-in DataSources that support only relational database systems.
- Ability to create custom data sources and connect to REST APIs.

Behaviors

- Extend the functionality of models without writing a set of functions in the models.
- Keep repetitive tasks in separate resource file and allow models to access it.
- Example: when updating a record in the database, perform many updates across the database.
- Cake comes with built-in behaviors: ACL, Containable, Translate, and Tree.
- Ability to create custom behaviors.

Utilities

- General-purpose libraries that can be called from anywhere in the app through the `App::import()` function.
- Cake comes with built-in utilities:
  - Configure: storing global variables.
  - File and Folder: managing file and folder operations.
  - HTTP Socket: managing requests to web services.

Plugins

- Mini-cake application inside other Cake app.
- Plugins are useful as third-party resources.
- Plugins contain models, views, and controllers just like any Cake application.
- Plugins are droped into the *app/plugins* directory.
- No built-in plugins in Cake only third-party plugins.
- Ability to create custom plugins so they can be used by other applications.

Cake Community (I)

- Cookbook – http://book.cakephp.org
  - Official CakePHP documentation.
- Bakery - http://bakery.cakephp.org
  - Official CakePHP community portal.
- API - http://api.cakephp.org
- All questions at CakePHP- http://ask.cakephp.org
  - Official questions and answers website.
- CakePHP TV - http://tv.cakephp.org
  - Free video tutorials.

Cake Community (II)

- CakeFest - http://cakefest.org
  - The CakePHP conference.
- CakeForge - http://cakeforge.org
  - Hosts open source CakePHP projects.
- CakePackages - http://www.cakepackages.com
  - Free CakePHP code to download.
- GitHub – https://github.com/cakephp/cakephp
  - Official CakePHP repository.

References

- Beginning CakePHP: From Novice to Professional By David Golding, ISBN: 1-4302-0977-1
- CakePHP Application Development By Anupom Syam and Ahsanul Bari, ISBN: 9781847193896
- CakePHP 1.3 Manual Book, http://book.cakephp.org
- CakePHP after 3 years, looking back and moving ahead by Gwoo http://bakery.cakephp.org/articles/gwoo/2008/04/22/after-3-years-looking-back-and-moving-ahead

Thank You!