**CakePHP: the rapid development PHP framework.**

DIEGO CASTILLO, Harding University.

The development of web applications is continuously been redefined by new technologies and paradigms. Every day, to define a modern web application becomes more complicated. PHP (PHP: Hypertext Preprocessor) is a scripting language for general-purpose which is particularly suited for developing web applications and it can also be embedded in HTML (PHP Documentation Group). It is the web's most popular server-side language (Diaz, 2013). The use of frameworks to develop powerful web applications aids towards alleviating the overhead of performing common tasks, improve security features, and work in a structured way (Cake Software Foundation, 2014). Choosing the right tool for the right job is one of the most important decisions which have to be made in order to deliver an exceptional product. The purpose of this paper is to explore the CakePHP framework, the implementation of the MVC software design pattern of it, what it is built for, and some of its best features to help developers accomplish different type of tasks in an easier way.

Categories and Subject Descriptors: D.3.3 [Language Constructs and Features]: Frameworks; D.2.11

[Software Architectures]: Domain-specific architectures

General Terms: PHP, framework, MVC, web.

Additional Key Words and Phrases: open-source, rapid development, CRUD, server-side, client-side, JSON, XML, PDF.

## 1. INTRODUCTION.

### 1.1. What is CakePHP?

CakePHP is a free, open-source, rapid development framework for PHP (Cake Software Foundation, 2014). It is an underlying well-formed structure for software developers which allow them to easily create robust web applications without having to reinvent the wheel every time a new project is started. According to the Cake Software Foundation, the primary goal of CakePHP is to enable software developers to work in a structured and rapid manner without the loss of flexibility in an application.

### 1.2. MVC in CakePHP.

In object-oriented programming, the Model-View-Controller (MVC) is a software design pattern for developing user interfaces and efficiently relating the user interface to underlying data models of an application (Rouse, 2011). CakePHP follows the MVC design pattern because it aids in the creation of maintainable, modular and rapid developed applications. The following graphic explains a typical MVC request of a CakePHP application:

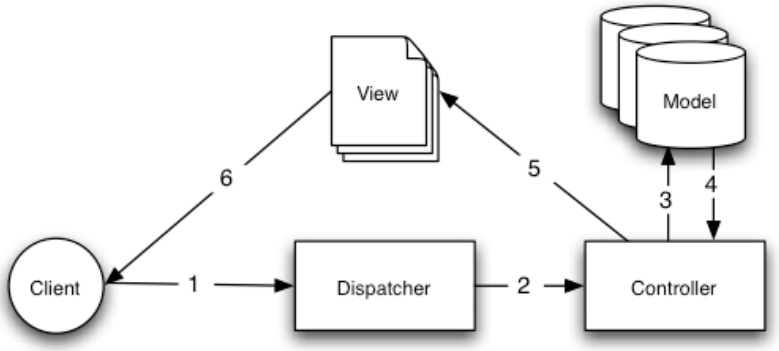


Figure 1: A typical MVC Request in CakePHP (Cake Software Foundation, 2014).

A CakePHP request starts with a client requesting a resource of the application. This request object is then processed by a dispatcher. The dispatcher's job is to convert a request object into a controller's action. It uses the dispatched request object and default route configurations to locate and load the correct controller associated with this request. If found, the requested action or method is called on the controller which was previously loaded (CakePHP, 2014). After the correct controller has been found, it will call the model layer to perform any processes which it was commanded to do by the controller's action. Finally, when this operation has been completed the controller will go ahead and delegate to the correct view associated with this controller's action the task of producing the output of the data supplied by the model.

### 1.3. **CakePHP Conventions.**

The CakePHP framework uses the software development approach of convention over configuration which seeks to develop applications following typical programming conventions, rather than developers defining their own configurations (Janssen). As a result, convention over configuration enables simplicity and fast software development, without necessarily minimizing the application's flexibility.

#### 1.3.1. **Model and Database Conventions.**

A model class name should be CamelCased and singular. In a blog application, the model which handles the posts made by a user should be named Post.php. The tables of the database in an application should be plural and underscored if it consists of more than one word. The table name for our example should be "posts." Field names of the tables with more than one word are underscored. All tables with which a CakePHP model interacts (except for join tables) must have a singular primary key to uniquely identify each row (Cake Software Foundation, 2014). When creating associations among models in an application, a foreign key is recognized by default as the singular name of the related table in the association followed by "_id."

#### 1.3.2. **Controller Conventions.**

A controller class name should be CamelCased, plural, and end with the word Controller. In the blog example from above, the controller associated with the posts of a user should be called PostsController.php. It is a good practice to have an index() action for every controller since if no action is specified by a request made to a controller, the default behavior of the framework will execute the index() action.

### 1.3.3. **View Conventions.**

A view file is named after the controller's action it displays with underscores if it consists of more than one word. For instance, if PostsController.php has an action called index(), the view template where this action will be rendered should be named index.ctp. The extension of view files in CakePHP stand for CakePHP template (Cake Software Foundation, 2014).

## 2. Models in CakePHP

A model represents the unchanging underlying structure of an application. It consists of the set of classes which model the heart of a project and tend to be stable and as long-lived as the problem itself (Deacon, 1995). The model layer will implement the business logic, validation, association with other models, among other tasks related to manipulating data in the application. In a CakePHP project models will usually represent a table of a database but they are not limited to this (Cake Software Foundation, 2014). The following code declares a model in CakePHP:

```
App::uses('AppModel', 'Model');
class Post extends AppModel {
    public $name = 'Post';
}
```

CakePHP model inheritance uses lazy loading which is a programming pattern to load resources only if these resources have been requested, this in turn helps applications to have better performance (Sironi, 2010). A model class inherits the AppModel.php class which extends from the core application Model.php class. The Model.php class contains the implementation for creating queries, deleting, and saving data to the model. As a result, the above declaration of the Post.php model is already packed with CRUD and other querying functionality. The AppModel.php class is initially empty and it is intended to be used for the declaration and implementation of methods which need to be shared and accessed from multiple models in an application. After a model has been defined it can be automatically accessed from within its respective controller using the following syntax:

```
$this->Post->someFunction();
```

In order to deal with accessing relational databases CakePHP uses ORM and the Active Record Pattern. ORM helps to transform data between two conflicting systems (Hibernate, 2011), and the Active Record Pattern facilitates objects an interface which allows software developers to execute functions similar to those found in SQL queries, such as create, read, update, or delete to an object which represents a database record (Fowler, 2002). This object also has similar properties to those stored in the database table which it represents.

### 2.1. **Model Associations.**

Models of an application commonly have different type of relations among them. For instance, a post by a user has many comments, while a comment belongs to a post. CakePHP provides a powerful an easy way to set up associations among models in an application.

| Relationship | Association Type | Example |
|---|---|---|
| one to one | hasOne | A user has one profile. |
| one to many | hasMany | A user can have multiple recipes. |

| Relationship | Association Type | Example |
|---|---|---|
| many to one | belongsTo | Many recipes belong to a user. |
| many to many | hasAndBelongsToMany | Recipes have, and belong to, many ingredients. |

Figure 2: The four association types in CakePHP (Cake Software Foundation, 2014).

Associations are declared as class variables and are named according to the type of association which is been created. An association class variable might be a multidimensional array or just a simple string. A class variable association needs to have an "Alias," which is an identifier for the relationship. Aliases for each model must be unique across the application (Cake Software Foundation, 2014). Continuing with the blog example, a post has many comments and this association between the Post.php model and the Comment.php model is declared in the following way:

```
class Post extends AppModel {
    public $hasMany = array(
        'Comment' => array(
            'className' => 'Comment',
            'foreignKey' => 'post_id'
        )
    );
}
```

Since the Post.php model now has an association with the Comment.php model, CakePHP automatically creates links between them, and as a result, the Comment.php model can be accessed from the Post.php model. The remaining associations provided by CakePHP can be similarly implemented to other models in an application.

### 2.2. **Model Data Validation.**

In a robust application, before any type of data is saved or updated, it needs to be analyzed and filtered to verify it conforms to the logic and business rules which were established for the application. Data validation in a model will essentially validate the data which has been passed to it after the save() method has been called. Nevertheless, data validation must be performed at different levels in an application.

To create the data validation rules in a model, the Model::validate array needs to be declared and populated with the validation rules. CakePHP comes with many built-in validation rules ready to be applied to the fields in the model such as URLs, formatting emails, credit card numbers, among others (Cake Software Foundation, 2014). The general pattern to follow in order to populate the Model::validate array is:

```
public $validate = array(
    'fieldName1' => array(
        'rule'          => 'ruleName',
        'required'      => true,
        'allowEmpty'    => false,
        'on'            => 'create',
        'message'       => 'Your Custom Error Message'
    )
);
```

CakePHP provides many different keys to add to a field validation such as “required”, “allowEmpty”, among others. Most of these rules are self-explanatory and easy to understand. In a similar fashion, it is also possible to add more than one validation rule to a single field in a model.

3. **Controllers in CakePHP.**
A controller's job is to analyze commands from the user and order the model or view to change as needed (Burbeck, 1992). A controller supplies different type of methods or actions which handle user's requests. A controller is created using the following syntax:

```
App::uses('AppController', 'Controller');
class PostsController extends AppController {
}
```

Controllers extend the AppController.php class which in turn extends from the main core controller of the framework Controller.php. The AppController.php class is initially empty and it is intended to be used to declare and implement methods which are going to be available in all of the application controllers.

3.1. **Controller Actions.**
An action is assigned the job of interpreting a request made by a user and providing a response for the browser or user making such request. Any public action inside a controller is available to be accessed from a URL by a user. By convention, CakePHP will render the controller's response in a view with a lowercased and underscored version of the action's name (Cake Software Foundation, 2014). For instance, if the view() action inside the PostsController.php file is called, by default the response of the view() action will be rendered in the view.ctp view file.

3.2. **Controller Methods.**
A controller comes loaded with many methods which it can use to be able to interact with the view. Controller methods will aid towards the communication of the controller and the view by passing information to it, deciding which view layout to render on, or which view file should be the response of the controller be loaded in. CakePHP provides a vast number of methods, but the following are the most commonly used.

3.2.1. **Controller::set(string $var, mixed $value)**
The set() method is the primary way to create a context variable to pass information from the controllers’ action to the corresponding view. This context variable can be later accessed from the view and rendered appropriately.

3.2.2. **Controller::render(string $view, string $layout)**

By default CakePHP will render the view which corresponds to the name of the controller. To override this behavior, the render() method can be used to render a controllers' action in a different view other than the view by convention, and even a different layout.

### 3.3. **Request Life-cycle Callbacks.**

CakePHP comes with useful callbacks which can be used to perform different type of tasks around the life-cycle of a request in an application.

#### 3.3.1. **Controller::beforeFilter()**

This function is called before any action inside the controller is executed. This method is commonly used to perform authentication and to check for active user sessions.

#### 3.3.2. **Controller::beforeRender()**

The beforeRender() method is called after the action inside a controller has finished processing the request, but before the response is rendered on a view.

#### 3.3.3. **Controller::afterFilter()**

This is the last controller method to run. It is called after both the controller action and the rendering are done.

## 4. **Views in CakePHP.**

In MVC, views are responsible for the rendering of the model information in the structure which the users desire (Yiiframework). Most of the time a view will be in the form of (X)HTML, JSON, or XML, but other types of files such as PDF's which users can download are also within the scope of responsibilities of the view layer (Cake Software Foundation, 2014). CakePHP view files are written by default in plain PHP and have a .ctp (CakePHP Template) extension. View templates contain all the logic which is needed in order to render the presentation of the information for the users the application is serving to. A view layer in a CakePHP project can be made up of many different parts, each of these parts have different uses and serve different purposes.

### 4.1. **Views.**

A view is unique to an action of a controller. This is the file where the response created by the action executed in a controller will be rendered appropriately. A view will most of the time be rendered inside of a layout.

### 4.2. **Elements.**

Elements are small blocks of presentation logic which are usually rendered inside of another view (Cake Software Foundation, 2014). These are often reusable code. Elements are useful for inserting "mini" views which are needed in many places in an application, such as menus, login forms, help boxes, among others. As a result, an application becomes easier to maintain since if a menu changes, only the root element where it was defined needs to be changed, instead of changing it in every place it is included.

### 4.3. **Layouts.**

A layout is a file which holds presentation code and it is used by many interfaces of the application. Layout content is the rendered version of a controller's response (a view) plus what is already contained within the layout. CakePHP default layout is called default.ctp. It is possible to create a custom layout and tell the controller's action to render in this custom layout instead.

### 4.4. **Helpers.**

Helper classes encapsulate logic which will be needed in other places in an application. Helpers will assist in the creation of well formatted markup using PHP.

#### 4.4.1. **Form Helper.**

The form helper assists developers to quickly and easily create HTML forms. Some of the advantages of using the CakePHP form helper rather than writing plain HTML text include the framework taking advantage of the conventions and displaying the right input type according to what is stored in the model business rules, re-population, and streamline validation (Cake Software Foundation, 2014).

#### 4.4.2. **HTML Helper.**

The goal of the HTML helper is to make HTML element attributes more flexible, easier, and faster to maintain (Cake Software Foundation, 2014). The use of this helper will assist developers to be more resilient on where it is placed in relation to the root domain of the application.

## 5. **Conclusion.**

This paper has been a brief overview of a very capable and complete framework for developing web applications. CakePHP allows modern server-side and client-side developers to deliver high end products to the market. It provides a useful set of tools which assist developers to code less and do more, while not compromising the application's security or performance.

## Works Cited

Burbeck, S. (1992). *How to use Model-View-Controller (MVC)*. Retrieved April 7, 2014, from Computer Science Department of Illinois: http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

***Cake Software Foundation. (2014, March 21). *CakePHP Cookbook Documentation*. Retrieved April 2, 2014, from CakePHP: http://book.cakephp.org/2.0/_downloads/en/CakePHPCookbook.pdf**

CakePHP. (2014, March 2). *CakePHP API*. Retrieved April 04, 2014, from CakePHP API: http://api.cakephp.org/2.3/class-Dispatcher.html

Deacon, J. (1995, August 1). *Model-View-Controller (MVC) Architecture*. Retrieved April 4, 2014, from Geographical Association of MYANMAR: http://geographymyanmar.org/sites/default/files/gam/publication_file/MVC.pdf

Diaz, C. (2013, March 6). *Server-side programming language statistics.* Retrieved April 4, 2014, from Websites Frameworks: http://blog.websitesframeworks.com/2013/03/programming-language-statistics-in-server-side-161/

Fowler, M. (2002). *Patterns of Enterprise Application Architecture.* Amsterdam: Addison-Wesley Longman.

Hibernate. (2011, April 19). *What is Object/Relational Mapping?* Retrieved April 9, 2014, from Hibernate: http://hibernate.org/orm/what-is-an-orm/

Janssen, C. (n.d.). *Convention Over Configuration*. Retrieved April 4, 2014, from Techopedia: http://www.techopedia.com/definition/27478/convention-over-configuration

PHP Documentation Group. (n.d.). *What is PHP?* Retrieved April 5, 2014, from PHP: Hypertext Preprocessor: http://www.php.net/manual/en/intro-whatis.php

Rouse, C. (2011, March). *Model-View-Controller (MVC).* Retrieved April 4, 2014, from Techtarget: http://whatis.techtarget.com/definition/model-view-controller-MVC

Sironi, G. (2010, April 5). *Practical PHP Patterns: Lazy Loading.* Retrieved April 7, 2014, from Dzone: http://css.dzone.com/books/practical-php-patterns/practical-php-patterns-lazy

Yiiframework. (n.d.). *Best MVC Practices.* Retrieved April 7, 2014, from Yiiframework: http://www.yiiframework.com/doc/guide/1.1/en/basics.best-practices#view

***Main source of information for topic.**