Beginning CakePHP: From Novice to Professional

Copyright © 2008 by David Golding

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-4302-0977-5

ISBN-13 (electronic): 978-1-4302-0978-2

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editors: Steve Anglin, Tom Welsh
Technical Reviewer: Richard K. Miller
Editorial Board: Clay Andres, Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Sofia Marchant
Copy Editor: Kim Wimpsett
Associate Production Director: Kari Brooks-Copony
Production Editor: Laura Cheu
Compositor: Linda Weidemann, Wolf Creek Press
Proofreader: Nancy Sixsmith, ConText Editorial Services, Inc.
Indexer: Becky Hornyak
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at http://www.apress.com/info/bulksales.

The information in this book is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at http://www.apress.com. You will need to answer questions pertaining to this book in order to successfully download the code.

CHAPTER 1

Introduction

Programmers have used frameworks for years, though for web development the use of frameworks has been more recent. Probably the main advantage of using a framework in any project, be it web-related or not, is explained by the concept of "inversion of control." Many programs operate in such a way that the code is in control. In other words, the code decides when one operation should appear, how it should handle the user's response, and so forth. Imagine if this order of control were inverted. Rather than have a script or library that contains a series of operations, the program has a series of objects that can do nothing until you extend them (even though they may contain tons of tools you could put to use). In this way, the framework calls on you, not the other way around.

For example, let's say you are looking for a way to install a voting program into your web site. You browse the Internet and find a handful of useful PHP scripts that all promise to do that for you. After plugging in some unique settings, you place one of these scripts onto your server and launch the program. The program runs just fine, but if you wanted to change anything, you would have to go into the script, locate where the operation occurs that you want to change, and work the adjustment by hand. The script manages the flow of control in the sense that all of its operations are executed when the program runs, and if you want to control the program, you have to alter the script.

A framework, on the other hand, has an inverted flow of control. To produce a voting application in a framework, you would have to add to the framework those objects that would handle the voting. The framework would automatically pull together several resources to make the voting process happen, and you would have to intercept those resources or extend them to add your own functionality. A library will behave on its own, like the script example, and any changes must be made directly in the code. A framework is different in that it will wait for you to extend or add to it before it can really do anything for you. You will not need to go directly to the framework's code to make changes; instead, the framework will take your extensions and use those instead of its own libraries.

CakePHP (or, for short, Cake) is a framework, not a set of libraries, even though it contains dozens of functions and methods that simplify web development much like libraries do. As such, Cake waits on you to extend its objects and add your own customized resources. With Cake, gone are the days of individually scripting each and every function. Instead, developers are using a bundled package of scripts, libraries, and conventions that are designed specifically for web development.

From Novice to Professional

This guide is for beginners to CakePHP. Whether or not you have much experience with the PHP scripting language, working in Cake will require some new methods you may or may not have tried before. If you don't know what a "has-and-belongs-to-many" relationship is, don't know how to build your own class object, or don't know how to parse an array, then this book is a perfect place to start when getting into Cake.

Most of the available online resources require some sort of prior knowledge of web development to get a grasp on how to install and work in Cake. If you're like me when I started using Cake, you probably just want a handful of tutorials with code samples from square one that can get you up and running quickly and lead you in the right direction for more advanced techniques. In fact, when asking a question on forums or chat rooms, many beginners get little help or confusing leads from the experts. Simple questions can get a response like "Well, just read the online manual and API." Sometimes novices need a very simple approach to the software, and this guide is just that. As you begin to master Cake, this guide will also provide tips and a reference for helping you quickly add more features to your projects and catch errors.

This book will start by showing how to install Cake on a server and your own computer and will provide some detailed code samples and visual snapshots to walk you through the process. In Chapter 2, I'll show how to build a simple Cake application. You'll get used to the Model-View-Controller (MVC) structure and how to organize your Cake applications effectively. In Part 2, you'll build more extensive web applications in Cake, and you'll explore Cake's built-in helpers, including the Ajax helper, and work with more advanced features. By the end of the book, you will be able to create your own helpers, plugins, and other useful features that will reduce the overall amount of code to run your applications, and you'll also have a solid enough foundation to try other advanced features on your own.

Why Cake?

Ever since Ruby on Rails became a popular web-based framework, teams of developers have been creating clones of Rails or Rails-like frameworks for various languages: TurboGears for Python; Zend, Symfony, and many others for PHP; Catalyst for Perl; and on and on. With so many options out there, why choose CakePHP for your web project?

It's PHP!

Many web developers complain about switching to Ruby on Rails simply because the framework is built on the Ruby language. PHP, they say, is one of the more widely supported web programming languages and is standard with most web services providers, so why give that up for Ruby? For those who learned web development on PHP or those who have made PHP their primary development tool, the idea of ditching PHP for something else may seem daunting or time-consuming. For companies, switching to another language can require reallocating resources, changing web service providers, or reworking an expensive server configuration. Whatever the case, leaving PHP for another development framework can be costly and time-consuming. With Cake, you can enjoy the benefits of framework-based development without learning another language.

One of the difficulties in using some PHP frameworks has been their compatibility with PHP 4 and 5. Symfony, for example, requires PHP 5 and is not backward compatible with PHP 4. Cake, on the other hand, is compatible with both versions of PHP, a necessary feature for many developers with long-term projects that go back a couple of years.

Many PHP developers overlook the benefits of a framework and simply look for premade functions or classes to be used as includes in their scripts or, as with Perl, pullin modules that chew up lots of time on the server and provide little customization. Cake, however, is thoroughly object-oriented in its scope. It supplies objects that can be implemented and modified to your liking and is not just some module or set of includes that give you little control.

Rapid Development

Getting a web project off the ground can be cumbersome and technically demanding, especially when using older methods of development. Cake, however, makes the initial steps of building a web application easy. Rather than run installation scripts from the command line, Cake comes prepackaged as a folder you simply drop onto a server and is ready to run.

The command line does come in handy once you begin building onto the framework. Later, I'll discuss Cake's scaffolding features that cut down on routine development tasks. With Cake, creating user flows in the application early on is simple and can improve communication with clients. In some cases, a run-through of the application can be developed in minutes, allowing the client to get an idea of the project's architecture.

Once a project is fleshed out and launched, site maintenance is also improved thanks to Cake. Because of its hierarchy and organization, as well as its effectiveness at limiting redundancy, Cake helps developers adjust a web application on the fly. Cake also supports test databases and URL routes for testing new features or versions of web applications on the live setup.

Model-View-Controller

Cake enforces an MVC structure for your web applications. Basically, it effectively separates typical operations into specific areas: models for all your database interaction, views for all your output and displays, and controllers for all your commands/scripts for input and program flow. The typical PHP application mixes each of these three functions in the same code, making it difficult to maintain and debug.

This is the typical flow for PHP scripting (see Figure 1-1):

1. The client sends a request to a PHP script by typing a URL or clicking a link of some kind.
2. The script processes the data and then sends the database requests directly to the database.
3. The script receives any database output and processes the data.
4. The script generates output and forwards it to the client's browser.

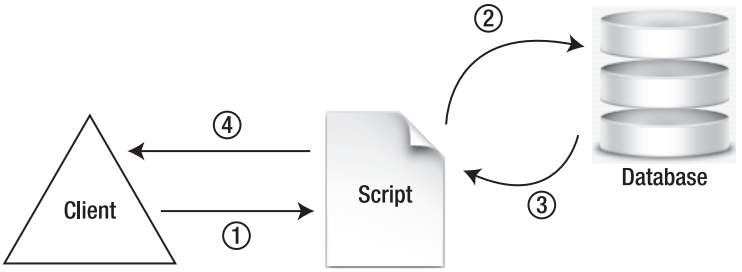

Figure 1-1. *The typical flow for PHP scripting*

In short, everything is contained in one PHP script. By using the `include()` function, developers strip out common functions into other external files, which makes it possible to reduce redundancy. The most complex PHP applications use objects that can be called anywhere in the application and modified depending on the variables and settings passed to them. Developers, when using objects and classes, can structure the application in numerous ways.

MVC improves upon the typical PHP flow and is an effective technique for making class objects available over the whole application. The main goal behind MVC is to make sure that each function of the application is written once and only once, thus streamlining code by reducing redundancy. Cake accomplishes this goal by not only providing the resources to make MVC possible but also by using a consistent method for where to store operations in the application. Simply naming your own files a certain way allows Cake to piece together the various resources without using any code specifications.

MVC can vary depending on the framework with which you're working, but generally it works as follows (see Figure 1-2):

1. The client sends a page request to the application, either by typing a URL or by clicking a link of some kind. By convention, a typical URL is usually structured like this:

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

2. The dispatcher script parses the URL structure and determines which controller to execute. It also passes along any actions and parameters to the controller.

3. The function in the controller may need to handle more data than just the parameters forwarded by the dispatcher. It will send database requests to the model script.

4. The model script determines how to interact with the database using the requests submitted by the controller. It may run queries with the database and do all sorts of handy data-sorting instructions.

5. Once the model has pulled any data from or sent data to the database, it returns its output to the controller.

6. The controller processes the data and outputs to the view file.

7. The view adds any design or display data to the controller output and sends its output to the client's browser.

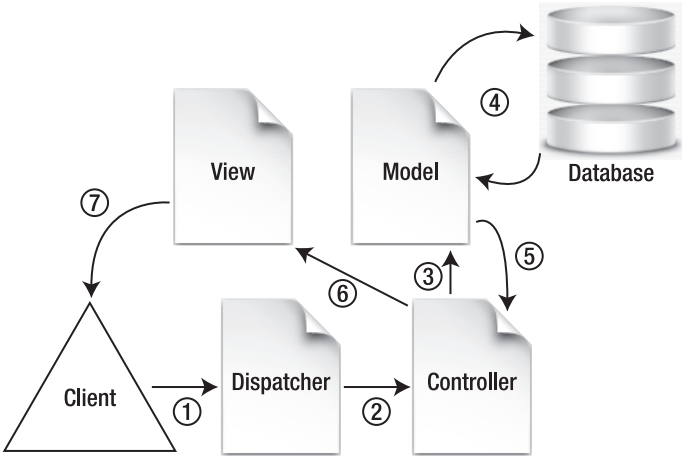

Figure 1-2. *How Cake makes use of the MVC structure*

The benefit of using MVC to develop web sites is that repeated functions or tasks can be separated, thus allowing for quicker edits. It can even help in debugging. Say an error keeps occurring during the interaction with the database. Usually the problem will be somewhere in a model. Knowing that all database interactions occur in just one place makes it easier to solve problems.

CRUD Operations and the Bake Script

Almost all web sites use CRUD operations: create, read, update, and delete. A blog, for example, will need to create posts; users will need to be able to read each post; the author will likely want the ability to edit the post in the future or update the post; and the author will also want access for deleting posts.

Cake makes these operations a breeze with its automated CRUD functions. Instead of writing each CRUD operation by hand, it has prebuilt classes that do it for you. Cake includes the Bake script, a handy command-line tool that generates editable CRUD code based on your database schema and customized parameters.

Scaffolding

Getting a web site off the ground is much easier with Cake's scaffolding abilities. With just one simple line of code, you can call out Cake's prebuilt scaffold to render views based on the database. In other words, it figures out how some standard interface views should work with your database and outputs the HTML forms, all without you having to write one bit of HTML. Although the scaffolding feature is not intended for full production views, it lets you begin testing logic and functions without wasting time building views or HTML output.

Helpers

Cake comes with standard HTML, Ajax, and JavaScript helpers that make creating views much easier. Your HTML output will be greatly facilitated by intuitive strings of helper code that render

the markup for you. And getting Ajax to work, although a little tricky at first, is much easier and far more efficient than if you had to worry about DOM peculiarities. What's more, you can download other helpers written by fellow Cake developers to boost the strength of the framework or even write your own to cut down on repetitive or clumsy markup.

Customizable Elements

You can customize each of Cake's features to fit your application. For example, you can bring FCKeditor, the popular WYSIWYG editor for web browsers, into Cake as a plugin. Using customized helpers, you can bring all the functionality of FCKeditor into your Cake application and actually trim out extra lines of PHP code to get it working. Later, I'll discuss other Cake elements such as components, helpers, and plugins, all of which can be customized by you for your specific needs or brought into your application as third-party resources from other developers.

Large Community

Should you need help down the road, a massive online community exists to provide it. In reality, the PHP community is the largest open source programming group on the Web, so if you need a quick workaround for a problem in Cake, someone somewhere will have some help for you, usually within minutes. Cake specialists have also established online forums, chat rooms, and blogs to help others improve and learn the framework. Compared to other PHP frameworks, this community is one of the largest on the Web.

Code samples are a must for anyone getting involved in web development. PHP dominates this field, and Cake has a growing repository of code samples as well. If you are considering another framework, this fact just may tip the scales in favor of Cake if you are wanting to piggyback on someone else's work.

More Features

Cake aims to simplify the development process for building web applications by providing an overall method for organizing the database and other resource files that cuts down on code. Although this general approach to web programming is itself a major feature Cake offers, its repository of other powerful resources such as built-in validation, access control lists (ACLs), data sanitization, security and session handling components, and view caching make Cake worth any serious developer's time.

Summary

As a framework, Cake inverts the flow of control and provides you, the developer, with an effective method for extending your web application in less time. Cake is built in PHP and therefore allows you to take advantage of a web-based framework without having to learn or use another programming language. Other benefits to using Cake include its MVC structure, which separates resources and functions of the application; the Bake script, which automates CRUD scripting; the scaffolding features, which reduce basic view rendering into one line of code; the built-in helpers, which reduce HTML output operations into single-line call-outs; and the large and still growing Cake community.