



F r o m T e c h n o l o g i e s t o S o l u t i o n s

Symfony 1.3

Web Application Development

Design, develop, and deploy feature-rich, high-performance PHP web applications using the Symfony framework

Tim Bowler

Wojciech Bancer

[PACKT]
PUBLISHING

Symfony 1.3

Web Application Development

Design, develop, and deploy feature-rich,
high-performance PHP web applications using
the Symfony framework

Tim Bowler

Wojciech Bancer



BIRMINGHAM - MUMBAI

Symfony 1.3 Web Application Development

Copyright © 2009 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: September 2009

Production Reference: 1150909

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 978-1-847194-56-5

www.packtpub.com

Cover Image by Vinayak Chittar (vinayak.chittar@gmail.com)

Credits

Authors

Tim Bowler
Wojciech Bancer

Reviewer

Jose Argudo Blanco

Acquisition Editor

David Barnes

Development Editor

Ved Prakash Jha

Technical Editors

Kartik Thakkar
Reshma Sundaresan

Copy Editor

Sneha Kulkarni

Indexer

Rekha Nair

Editorial Team Leader

Abhijeet Deobhakta

Project Coordinator

Srimoyee Ghoshal
Neelkanth Mehta

Proofreader

Lynda Sliwoski

Graphics

Nilesh Mohite

Production Coordinator

Shantanu Zagade

Cover Work

Shantanu Zagade

About the Authors

Tim Bowler has a Bachelor's Degree in Computer Science, a Masters Degree in Internet Technologies and e-commerce, and is currently studying for his Ph.D. in Near Field Communication. With over ten years of experience in web application development and agile project management, he has gained an MIET membership at the Institute of Engineering and Technology (IET) and Chartered I.T. Professional membership at the British Computer Society (BCS).

Tim started his career developing web applications in PHP for a digital media agency in London. As client expectations and delivery times became more and more demanding, he introduced agile and scrum into the development process along with the Symfony framework, in order to effectuate rapid application development.

Tim is currently the Managing Director at Agile Labs (<http://www.agilelabs.co.uk>), which specialize in web application development and agile coaching.

I would like to thank all of my editors at Packt Publishing – Ved Prakash Jha, Srimoyee Ghoshal, Neelkanth Mehta and David Barnes – for this book.

I would also like to thank my parents, Marian and Michael Bowler, for inspiring me to do well and to achieve everything possible.

And finally, I would like to thank all of my friends for their patience and understanding that books don't write themselves.

Wojciech Bancer has a Master's Degree in Computer Science. He has over eight years of experience in web application development. In 2007, after passing the Zend exam, he gained a Zend Certified Engineer for PHP5 certificate. Wojciech started his career developing web applications in PHP4 and PHP5, as a freelancer. Later he started working for a digital media agency in London, where he was introduced to Symfony and the scrum process. Currently he is a Lead Developer at Agile Labs.

I thank all of the Symfony developers for their great work of creating the Symfony framework, making a PHP developer's life much easier.

I also thank Packt Publishing and my editors – Neelkanth Mehta, Ved Prakash Jha, Srimoyee Ghoshal, Reshma Sundaresan – for their kindness and support.

A special thanks to my wife, Kate, and my friends for their support and patience during long evenings spent on writing this book.

About the Reviewer

Jose Argudo is a web developer from Valencia, Spain. After finishing his studies he started working for a web design company. After six years of working for that company, and others, he decided to start working as a freelancer.

Now, after some years have passed, he thinks it's the best decision he has ever made, a decision that lets him work with the tools he likes, such as Joomla!, CodeIgniter, CakePHP, JQuery, and other known open source technologies.

For the last months he has also been reviewing Packt Publishing books, such as Magento Theme Design, Magento Beginners Guide, and others about to be published, such as Magento Development with PHP, Joomla! SEO, Joomla! and Flash, and a book on Symfony framework.

If that isn't enough, he is also writing a book on CodeIgniter for Packt Publishing, something that he is putting all his effort into.

To my brother, I wish him the best.

Table of Contents

Preface	1
Chapter 1: Getting Started with Symfony	7
Exploring Symfony	7
The framework	7
The Model-View-Controller pattern	8
Taking a look at the key features	10
Forms and validation	10
Plugins	10
Internationalization and localization	10
Generators	11
Cache	11
Testing	11
Configuration files	11
Coding guidelines	12
Symfony-specific guidelines	12
Installing Symfony	12
Summary	13
Chapter 2: Developing Our Application	15
The milkshake shop	15
Creating the skeleton folder structure	16
Creating our database schema	21
Configuring the ORM layer	25
Configuring the database connection	26
Generating the models, forms, and filters	27
Building the database	28
Creating the application modules	29
Handling the routing	31
The application logic	32
Rendering the template	34
Adding our routing rules	35

Configuring template parameters	36
Styling the pages	37
Common installation problems	39
Web server 404 error page	39
A symfony 'Oops! An error occurred'	39
Pages and debug bar do not render correctly	40
Summary	41
Chapter 3: Adding the Business Logic and Complex Application Logic	43
The generated models	43
Populating the database	44
Retrieving data using the models	46
Defining the criteria	46
Hydration	47
Retrieving the result set from the action	48
The template logic	49
Returned results	49
Using the DAOs	50
Displaying the results	50
Helpers	52
Paginating our menu	54
Adding the pager business logic	54
Setting a configuration option	55
Amending the action	55
Accessing the \$_POST, \$_GET, and \$_REQUEST variables	56
Accessing the application and module configuration file	56
Building up the routing	57
Organizing a template with partials	58
Creating the milkshake page	62
Routing with an object	64
Adding the route to the template	65
Retrieving many-to-many results	65
Accessing related objects in the action	66
Accessing related objects in the templates	67
Plugins	68
DbFinderPlugin	69
Finishing off the location page	71
Summary	73
Chapter 4: User Interaction and Email Automation	75
The signup module	75
Binding a form to a database table	77
A look at the generated base class	78
Rendering the form	80
Customizing form widgets and validators	82

Removing unneeded fields	83
Modifying the form widgets	83
Adding form validators	84
Form naming convention and setting its style	85
Submitting the form	85
Changing the global rendering of forms	87
Customizing the rendering of the form	89
Form security for the user	91
Creating a simple form	92
Automated email responses	94
Adding the mailer settings to the application	94
Creating the application logic	94
The partial email template	96
Flashing temporary values	97
Creating a plugin	99
Packaging a plugin	107
Summary	108
Chapter 5: Generating the Admin Area	109
How Symfony can help us	109
Initializing generator	110
Creating application and module	110
Exploring list view	111
Looking into the generated list view code	113
Customizing the admin generator	116
Customizing the edit view	118
Handling foreign keys using admin generator	119
Accessing application settings from generator.yml	122
Using partials in the generated views	123
Customizing the layout	124
Securing the application	125
Setting up credentials (permissions)	130
Handling credentials in templates	131
Tidying up the backend	132
Summary	132
Chapter 6: Advanced Forms and JavaScript	133
Adding JavaScript code into the Symfony project	133
JavaScript frameworks	134
Using JavaScript helpers	134
Adding JavaScript files into the header section	135
Creating more advanced admin modules	136
Installing the required plugins and libraries	137

Creating an advanced admin module	138
Adding file upload and thumbnails	141
Handling many-to-many relations	143
Adding jQuery calendar and TinyMCE widget	144
Autocompleting the search	145
Other JavaScript helpers	149
Summary	149
Chapter 7: Internationalizing our Global Positions	151
Internationalization and localization	151
Refactoring the schema	152
Rebuilding with test data	154
Setting and getting the culture and language	155
Preferred culture and language	156
The action	156
Adding culture to the routing	157
Localizing the template	158
Translating interface text	159
Configuring i18n for the templates	159
Dictionary files	160
Translating the interface	161
Adding the culture links	161
Translating the static text	162
Summary	165
Chapter 8: Extending Symfony	167
Bridging to other frameworks	167
Bridging with eZ Components	167
Configuring the component with Symfony	168
Using the component	169
Bridging with the Zend Framework	172
Extending the core classes with your own	172
Multiple inheritance	173
Summary	
Chapter 9: Optimizing for Performance	175
HTTP compression	175
Caching	177
Cache settings	178
Caching globally	178
Caching page-by-page	179
Caching without the layout	180
Caching with the layout	181

Caching parts of a template	183
Dynamic cache	183
Cache storage	183
Caching dynamic pages	184
Looking at the database	184
Setting limits and columns in the criteria	185
Creating your own SQL statements	185
Limit your queries	186
Caching your queries	187
ETags	187
Less requests	187
Stylesheets	188
JavaScripts	188
Other tools to aid you	189
Firefox developer tools	189
Database tools	189
Deciding on your table types	190
Accelerators	191
memcached	191
Caching database calls	194
Summary	195
Chapter 10: Final Tweaks and Deployment	197
Editing the default pages	197
Disabling the application	199
Symfony on the server or not?	200
Transferring your application to the server	201
rsync	201
Symfony and rsync	202
Summary	204
Index	205

Preface

Back in the days, PHP developers developed web sites using a mixture of PHP functional code and HTML, with no separation between the two. The problem with this is that larger sites lost scalability and maintainability. Not to mention that there was vast amount of code duplication. The increasing demand for web applications sparked a need for a better way of rapid application development.

A framework helps a developer to create code that is readable as well as maintainable. Further more, it helps to alleviate repetitive tasks by automating them and provides additional classes and tools to aid in rapid application development. The Symfony framework is one of the best frameworks available today. It contains all of the features mentioned in the previous sentences and even more. If Symfony doesn't have something you need, then by integrating external components you can achieve it quiet easily. By using the Symfony framework for your projects, you will be able to develop web applications quickly and more easily.

What this book covers

Chapter 1: Getting Started with Symfony gives an overview of the MVC framework and covers the key features of Symfony framework, such as plugins, generators, internationalization, forms, and validation that help to save time on development of an application.

Chapter 2: Developing our Application shows how to start developing an application with less effort by using the Command Line Interface (CLI). In this chapter, you will learn the basic activities, such as creating the folder structure and database schema, configuring the ORM layer, and generating models, forms, and filters. Finally, we will see how to build the database and handle the routing. We also learn to add styling to the pages and cover some common installation problems.

Chapter 3: Adding the Business Logic and Complex Application Logic shows how we can add business and application logic to make the prototype (created in Chapter 2) to interact with the database. In this chapter, become familiar with the flow of the MVC pattern in Symfony. You will see how a request is handled and passed to the application logic, which in turn will retrieve data using models before passing the results to the view. This chapter also illustrates how to add plugins with an example of adding the DbFinderPlugin plugin to the application.

Chapter 4: User interaction and email automation introduces the Symfony subframework that handles forms. Here we will see how Symfony can generate nice looking forms for us, before creating our own formatting class. We then progress to create a fully customized form. We will also learn about how Symfony can be expanded to use the other third-party libraries, and how can we convert a module into a fully working plugin that can be packaged up and reused in other projects.

Chapter 5: Generating the Admin Area explains how we can build a backend admin area application without having to code much. In this chapter, we will initialize the Propel admin generator and customize it. Then we will see how to handle Foreign Keys using the admin generator. We will customize the layout and then secure the application by setting permissions for the user, and look at how we can handle credentials from the template.

Chapter 6: Advanced Forms and JavaScript contains examples on how to add JavaScript into Symfony, how to use more advanced widgets in forms, and how to handle M-N database relations. Finally we look at how to add AJAX support into your application.

Chapter 7: Internationalizing our Global Positions introduces internationalization and localization to parts of our application. In this chapter you will learn how to automatically set user language and how to allow the user to change their language. You will learn to create the XLIFF dictionary files using the Symfony tasks, which will help in internationalization and localization of the application. We will also see how to create the database to accommodate a multilingual site, and how Symfony handles the data retrieval for us.

In *Chapter 8: Extending Symfony*, you will learn to integrate components from other frameworks, such as eZ Components and the Zend Framework.

Chapter 9: Optimizing for Performance is all about optimizing our site by introducing compression and caching. We will start by looking at and using Symfony's caching framework. To take things a little further we then introduce a caching server. We will also look at several other useful tools that aid in speeding up web applications.

Chapter 10: Final Tweaks and Deployment introduces some of the ways to deploy web applications. Here we take a look at a better way to transfer applications than using FTP. We will also learn to customize the default error 404 and error 500 Symfony pages to match our site.

What you need for this book

LAMP or WAMP stack plus memcached installed. You will also need PEAR installed if you wish to install Symfony via pear.

Basic knowledge of object-oriented design and ORM will be quite helpful.

Who this book is for

This book is for PHP web developers who want to get started with Symfony 1.3. If you are already using Symfony 1.0 or are new to Symfony, you will learn how to use it in the best way to produce better applications faster.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "Open the `settings.yml` file, and then look for the compressed parameter key halfway down."

A block of code is set as follows:

```
<?php use_helper('JavascriptBase'); ?>
<?php echo javascript_tag("
    function name()
    {
        //Code
    }
") ?>
```


When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:


```
dev:
  .settings:
    error_reporting:      <?PHP echo (E_ALL | E_STRICT)."\\n" ?>
    web_debug:            on
    cache:              on
    no_script_name:       off
    etag:                 off
```

Any command-line input or output is written as follows:

```
$/home/timmy/workspace/milkshake>Symfony generate:module frontend best
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "As you can see in the following screenshot, the total page size is **113 KB**".

 Warnings or important notes appear in a box like this.

 Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an email to feedback@packtpub.com, and mention the book title via the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on www.packtpub.com or email suggest@packtpub.com.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book on, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code for the book

Visit http://www.packtpub.com/files/code/4565_Code.zip to directly download the example code.

The downloadable files contain instructions on how to use them.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration, and help us to improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/support>, selecting your book, clicking on the **let us know** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata added to any list of existing errata. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or web site name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Getting Started with Symfony

This chapter is an overview of the Symfony framework and how good it is to develop with. It will cover how Symfony conforms to the MVC pattern, the main features, general coding guidelines, and how to install it.

By the end of this chapter you will know:

- About the MVC pattern
- How Symfony incorporates the MVC pattern
- How to install Symfony

Exploring Symfony

Symfony was released in October 2005 by Fabien Potencier who is the CEO of Sensio, which is a French web agency (<http://www.sensio.com>). After Fabien used the framework on several projects successfully, he decided to release the project under an open source license. Ever since its first release, the Symfony community has increased dramatically and continues to do so.

The community can be found at <http://www.symfony-project.org/>.

The framework

A framework is aimed at reducing the development time without the need to sacrifice maintainability, scalability, or quality. Symfony can take less than a day to learn, comes with many tools and classes, and is easy to install. This means the developer can spend more time developing the application. All of these reasons and many more are why Symfony has come about, and why it has maintained its place as one of the best PHP5 frameworks.

The current trends at the moment seem to revolve around agile development methodologies with groups of developers working on the same web application. Using the Symfony framework, developers are aided in writing structured and maintainable code. This is all down to the framework's strict implementation of the **Model-View-Controller (MVC)** paradigm and modulization.

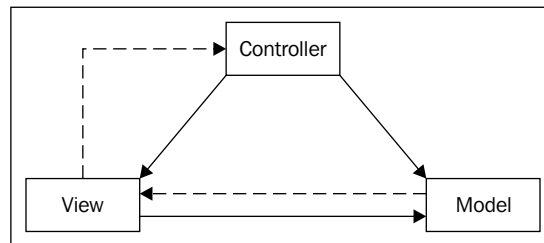
"It aims to speed up the creation and maintenance of web applications, and to replace the repetitive coding tasks by power, control and pleasure."

More information about this project can be found at
<http://www.symfony-project.org/about>.

The Model-View-Controller pattern

Many books go into the details of what the MVC pattern is and how it works. However, we will just look at the basic overview and how Symfony incorporates the pattern.

The MVC pattern is designed to split the presentation and business logic, and has a controller that manages the user's interactions between the two.



When you first use Symfony to generate the skeleton code for a new application and module, you can see exactly how Symfony strictly abides by the MVC pattern.

Controller

The **controller** is responsible for processing user events. The controllers in Symfony are split into several components.

1. It is the entry point into the application.
2. It determines what action is required to execute.
3. Loads the configurations.
4. Executes the filters.

One great feature about the controller being the entry point is that any time a site needs to go down for maintenance, the controller can simply be disabled. Creation of a new application in Symfony creates two controllers:

- A controller for the production environment
- A controller for the development environment

The difference between the two is the debug information and error displaying.

The controller calls an action, which is what drives the application. The action contains all of the application logic and has the ability to access everything from the request, sessions, authentication, and core Symfony objects.

Model

The **model layer** represents the applications data and the business rules used to manipulate and access it.

Symfony's model layer is split into two separate layers—an **Object Relational Mapping (ORM)** layer and a data abstraction layer. Of course, there are a few good PHP5 ORM and database abstraction libraries that already exist. Therefore, rather than reinventing the wheel, the framework incorporates the Doctrine ORM (<http://www.doctrine-project.org/>) which is the default ORM layer, with the option of using the Propel ORM (<http://propel.phpdb.org>). The second layer, being the data abstraction layer is handled by PHP Data Objects (PDO).

Database abstraction means database portability. Every database vendor will have a slight variant in their SQL syntax. Therefore, by moving your application to another RDBMS, a developer would have to amend certain queries. But with a database abstraction layer, this portability becomes transparent.

Object relational mapping turns database tables, rows, and different variable types into objects. As Symfony is written using OOP, it makes sense that the data is returned as an object.

At the moment, Symfony comes shipped with Propel 1.2 as it's default ORM. However, this whole ORM layer can be easily changed. For example, the ORM layer can be changed to Doctrine (<http://www.phpdoctrine.org/>).

Views

A **view**, which is commonly referred to as a template, is displayed to the user. These templates are completely separated from controllers and models. They mainly comprise of XHTML markup and presentation logic in the form of PHP tags. Although Symfony's template system has matured, the view layer can be replaced with another template engine, such as Smarty (<https://smarty.php.net>) through a plugin, for example.

Taking a look at the key features

We have looked at Symfony's implementation of the MVC pattern. Next, let's go over some of the features that Symfony has to offer in order to cut down development time.

Forms and validation

This is one of those repetitive requirements that a developer always has to face. Using Symfony, the development time is decreased due to the form subframework. There are two types of form:

- **Propel form** is a form that is based on a database table(s). These forms persist the submitted data to the table(s) that they are based on. As part of the generation task(s), these forms are automatically created along with validation. Although we can easily customize both form and validation, the default forms are a great way to display an initial prototype.
- **Simple form** is a form that doesn't persist data to the database. Although they are not generated, they follow the same approach as the Propel-based forms.

Plugins

One of Symfony's best features is its plugin architecture. So, many units of functionality can be written as a plugin and used time and again. The available plugins either help a developer in some way, or provide full, feature-rich applications. Looking at the plugin repository, numerous plugins have been submitted by the community and it continues to grow. You can visit <http://trac.symfony-project.com/wiki/SymfonyPlugins> to know more about Symfony Plugins. A few of the main plugins are:

- `sfGuardPlugin`: Web asset management
- `sfSimpleBlog`: Simple blog for your site
- `sfSimpleCMSPlugin`: Create a CMS
- `sfLucenePlugin`: Integrates the Zend framework's search engine

Internationalization and localization

Many web applications offer locale translations and services based on your locale. Symfony provides interfaces, standards, and localized helpers to make internationalization (i18N) and localization (l10N) simple.