

The Definitive Guide to symfony



François Zaninotto and
Fabien Potencier

The Definitive Guide to symfony

Copyright © 2007 by Sensio SA

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and one Back-Cover Text: "Apress (<http://www.apress.com/>) and the authors ask for your support by buying the print edition through any online or retail outlet." A copy of the license is included in the section entitled "GNU Free Documentation License."

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): 978-1-59059-786-6

ISBN-10 (pbk): 1-59059-786-9

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 Editor: Jason Gilmore

Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick,

Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Matt Wade

Project Manager: Kylie Johnston

Copy Edit Manager: Nicole Flores

Copy Editors: Marilyn Smith and Ami Knox

Assistant Production Director: Kari Brooks-Copony

Production Editor: Katie Stence

Compositor: Susan Glinert

Proofreaders: Linda Marousek and April Eddy

Indexer: Toma Mulligan

Artist: April Milne

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 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit <http://www.apress.com>.

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> in the Source Code/Download section.

Contents at a Glance

About the Authors	xxi
About Sensio Labs	xxiii
Acknowledgments	xxv
Introduction	xxvii
License	xxix

PART 1 ■■■ The Basics

■ CHAPTER 1	Introducing Symfony	3
■ CHAPTER 2	Exploring Symfony's Code	13
■ CHAPTER 3	Running Symfony	35
■ CHAPTER 4	The Basics of Page Creation	49
■ CHAPTER 5	Configuring Symfony	61

PART 2 ■■■ The Core Architecture

■ CHAPTER 6	Inside the Controller Layer	83
■ CHAPTER 7	Inside the View Layer	113
■ CHAPTER 8	Inside the Model Layer	141

PART 3 ■■■ Special Features

■ CHAPTER 9	Links and the Routing System	171
■ CHAPTER 10	Forms	191
■ CHAPTER 11	Ajax Integration	221
■ CHAPTER 12	Caching	245
■ CHAPTER 13	I18N and L10N	265

PART 4 ■■■ Development Tools

■ CHAPTER 14	Generators	281
■ CHAPTER 15	Unit and Functional Testing	317
■ CHAPTER 16	Application Management Tools	345
■ CHAPTER 17	Extending Symfony	367

PART 5 ■■■ Becoming a Symfony Expert

■ CHAPTER 18	Performance	397
■ CHAPTER 19	Mastering Symfony's Configuration Files	417
■ APPENDIX	GNU Free Documentation License	437
■ INDEX	445

About the Authors

■ **FRANÇOIS ZANINOTTO** is a consultant and project manager for Internet application projects. He graduated from the French business school Ecole des Mines in 1997 with a specialization in computer science. He tried quite a few jobs before settling on the Internet business: social worker in a children's facility, manager of a bike rental shop, web project manager for a tire manufacturer, writer of a travel guide on Germany for the same tire manufacturer, logistician for Médecins Sans Frontières, and IT architect for a consumer credit company. He joined the Sensio web agency in 2003, and since then has managed many Internet and intranet web application projects, dealing with complex usability issues, agile development methodologies, and cutting-edge web techniques. When the symfony project started, he took responsibility for the documentation, and wrote the symfony online book and tutorials.

■ **FABIEN POTENCIER** is a serial entrepreneur. Since he was ten, he always dreamed of creating and running companies. He started his career with an engineering degree from the French business school Ecole des Mines and an MBA in entrepreneurship from HEC Paris. In 1998, right after graduation, Fabien founded his very first company with a fellow student. The company was a web agency focused on simplicity and open source technologies, and was called Sensio. His acute technical knowledge and his endless curiosity won him the confidence of many French big corporate companies. While Sensio kept growing (at the time of writing, it has more than 30 employees), Fabien started other businesses: an indoor go-kart circuit in Lille (France), an auto spare parts e-commerce shop, and an autopilot training business riding on the most famous French racetracks. Fabien is the main developer of the symfony framework and is responsible for 95% of its code. Today, Fabien spends most of his time as Sensio's CEO and as the symfony project leader.



Introducing Symfony

What can symfony do for you? What's required to use it? This chapter answers these questions.

Symfony in Brief

A *framework* streamlines application development by automating many of the patterns employed for a given purpose. A framework also adds structure to the code, prompting the developer to write better, more readable, and more maintainable code. Ultimately, a framework makes programming easier, since it packages complex operations into simple statements.

Symfony is a complete framework designed to optimize the development of web applications by way of several key features. For starters, it separates a web application's business rules, server logic, and presentation views. It contains numerous tools and classes aimed at shortening the development time of a complex web application. Additionally, it automates common tasks so that the developer can focus entirely on the specifics of an application. The end result of these advantages means there is no need to reinvent the wheel every time a new web application is built!

Symfony was written entirely in PHP 5. It has been thoroughly tested in various real-world projects, and is actually in use for high-demand e-business websites. It is compatible with most of the available databases engines, including MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. It runs on *nix and Windows platforms. Let's begin with a closer look at its features.

Symfony Features

Symfony was built in order to fulfill the following requirements:

- Easy to install and configure on most platforms (and guaranteed to work on standard *nix and Windows platforms)
- Database engine-independent
- Simple to use, in most cases, but still flexible enough to adapt to complex cases
- Based on the premise of convention over configuration—the developer needs to configure only the unconventional
- Compliant with most web best practices and design patterns
- Enterprise-ready—adaptable to existing information technology (IT) policies and architectures, and stable enough for long-term projects

- Very readable code, with phpDocumentor comments, for easy maintenance
- Easy to extend, allowing for integration with other vendor libraries

Automated Web Project Features

Most of the common features of web projects are automated within symfony, as follows:

- The built-in internationalization layer allows for both data and interface translation, as well as content localization.
- The presentation uses templates and layouts that can be built by HTML designers without any knowledge of the framework. Helpers reduce the amount of presentation code to write by encapsulating large portions of code in simple function calls.
- Forms support automated validation and repopulation, and this ensures a good quality of data in the database and a better user experience.
- Output escaping protects applications from attacks via corrupted data.
- The cache management features reduce bandwidth usage and server load.
- Authentication and credential features facilitate the creation of restricted sections and user security management.
- Routing and smart URLs make the page address part of the interface and search-engine friendly.
- Built-in e-mail and API management features allow web applications to go beyond the classic browser interactions.
- Lists are more user-friendly thanks to automated pagination, sorting, and filtering.
- Factories, plug-ins, and mixins provide a high level of extensibility.
- Ajax interactions are easy to implement thanks to one-line helpers that encapsulate cross-browser-compatible JavaScript effects.

Development Environment and Tools

To fulfill the requirements of enterprises having their own coding guidelines and project management rules, symfony can be entirely customized. It provides, by default, several development environments and is bundled with multiple tools that automate common software-engineering tasks:

- The code-generation tools are great for prototyping and one-click back-end administration.
- The built-in unit and functional testing framework provides the perfect tools to allow test-driven development.
- The debug panel accelerates debugging by displaying all the information the developer needs on the page he's working on.

- The command-line interface automates application deployment between two servers.
- Live configuration changes are possible and effective.
- The logging features give administrators full details about an application's activities.

Who Made Symfony and Why?

The first version of symfony was released in October 2005 by project founder Fabien Potencier, coauthor of this book. Fabien is the CEO of Sensio (<http://www.sensio.com/>), a French web agency well known for its innovative views on web development.

Back in 2003, Fabien spent some time inquiring about the existing open source development tools for web applications in PHP. He found that none fulfilled the previously described requirements. When PHP 5 was released, he decided that the available tools had reached a mature enough stage to be integrated into a full-featured framework. He subsequently spent a year developing the symfony core, basing his work on the Mojavi Model-View-Controller (MVC) framework, the Propel object-relational mapping (ORM), and the Ruby on Rails templating helpers.

Fabien originally built symfony for Sensio's projects, because having an effective framework at your disposal presents an ideal way to develop applications faster and more efficiently. It also makes web development more intuitive, and the resulting applications are more robust and easier to maintain. The framework entered the proving grounds when it was employed to build an e-commerce website for a lingerie retailer, and subsequently was applied to other projects.

After successfully using symfony for a few projects, Fabien decided to release it under an open source license. He did so to donate this work to the community, to benefit from user feedback, to showcase Sensio's experience, and because it's fun.

Note Why “symfony” and not “FooBarFramework”? Because Fabien wanted a short name containing an *s*, as in Sensio, and an *f*, as in framework—easy to remember and not associated with another development tool. Also, he doesn't like capital letters. *symfony* was close enough, even if not completely English, and it was also available as a project name. The other alternative was “baguette.”

For symfony to be a successful open source project, it needed to have extensive documentation, in English, to increase the adoption rate. Fabien asked fellow Sensio employee François Zaninotto, the other author of this book, to dig into the code and write an online book about it. It took quite a while, but when the project was made public, it was documented well enough to appeal to numerous developers. The rest is history.

The Symfony Community

As soon as the symfony website (<http://www.symfony-project.com/>) was launched, numerous developers from around the world downloaded and installed the framework, read the online documentation, and built their first application with symfony, and the buzz began to mount.

Web application frameworks were getting popular at that time, and the need for a full-featured framework in PHP was high. Symfony offered a compelling solution due to its impressive code quality and significant amount of documentation—two major advantages over the other players in the framework category. Contributors soon began to surface, proposing patches and enhancements, proofreading the documentation, and performing other much-needed roles.

The public source repository and ticketing system offer a variety of ways to contribute, and all volunteers are welcome. Fabien is still the main committer in the trunk of the source code repository, and guarantees the quality of the code.

Today, the symfony forum, mailing lists, and Internet Relay Chat (IRC) channel offer ideal support outlets, with seemingly each question getting an average of four answers. Newcomers install symfony every day, and the wiki and code snippets sections host a lot of user-contributed documentation. The number of known symfony applications increases by an average of five per week, and counting.

The symfony community is the third strength of the framework, and we hope that you will join it after reading this book.

Is Symfony for Me?

Whether you are a PHP 5 expert or a newcomer to web application programming, you will be able to use symfony. The main factor in deciding whether or not to do so is the size of your project.

If you want to develop a simple website with five to ten pages, limited access to a database, and no obligations to ensuring its performance or providing documentation, then you should stick with PHP alone. You wouldn't gain much from a web application framework, and using object orientation or an MVC model would likely only slow down your development process. As a side note, symfony is not optimized to run efficiently on a shared server where PHP scripts can run only in Common Gateway Interface (CGI) mode.

On the other hand, if you develop more complex web applications, with heavy business logic, PHP alone is not enough. If you plan on maintaining or extending your application in the future, you will need your code to be lightweight, readable, and effective. If you want to use the latest advances in user interaction (like Ajax) in an intuitive way, you can't just write hundreds of lines of JavaScript. If you want to have fun and develop fast, then PHP alone will probably be disappointing. In all these cases, symfony is for you.

And, of course, if you are a professional web developer, you already know all the benefits of web application frameworks, and you need one that is mature, well documented, and has a large community. Search no more, for symfony is your solution.

Tip If you would like a visual demonstration, take a look at the screencasts available from the symfony website. You will see how fast and fun it is to develop applications with symfony.

Fundamental Concepts

Before you get started with symfony, you should understand a few basic concepts. Feel free to skip ahead if you already know the meaning of OOP, ORM, RAD, DRY, KISS, TDD, YAML, and PEAR.

PHP 5

Symfony is developed in PHP 5 (<http://www.php.net/>) and dedicated to building web applications with the same language. Therefore, a solid understanding of PHP 5 is required to get the most out of the framework.

Developers who already know PHP 4 but not PHP 5 should mainly focus on the language's new object-oriented model.

Object-Oriented Programming (OOP)

Object-oriented programming (OOP) will not be explained in this chapter. It needs a whole book itself! Because symfony makes extensive use of the object-oriented mechanisms available as of PHP 5, OOP is a prerequisite to learning symfony.

Wikipedia explains OOP as follows:

The idea behind object-oriented programming is that a computer program may be seen as comprising a collection of individual units, or objects, that act on each other, as opposed to a traditional view in which a program may be seen as a collection of functions, or simply as a list of instructions to the computer.

PHP 5 implements the object-oriented paradigms of class, object, method, inheritance, and much more. Those who are not familiar with these concepts are advised to read the related PHP documentation, available at <http://www.php.net/manual/en/language.oop5.basic.php>.

Magic Methods

One of the strengths of PHP's object capabilities is the use of *magic methods*. These are methods that can be used to override the default behavior of classes without modifying the outside code. They make the PHP syntax less verbose and more extensible. They are easy to recognize, because the names of the magic methods start with two underscores (`__`).

For instance, when displaying an object, PHP implicitly looks for a `__toString()` method for this object to see if a custom display format was defined by the developer:

```
$myObject = new myClass();  
echo $myObject;  
// Will look for a magic method  
echo $myObject->__toString();
```

Symfony uses magic methods, so you should have a thorough understanding of them. They are described in the PHP documentation (<http://www.php.net/manual/en/language.oop5.magic.php>).

PHP Extension and Application Repository (PEAR)

PEAR is “a framework and distribution system for reusable PHP components.” PEAR allows you to download, install, upgrade, and uninstall PHP scripts. When using a PEAR package, you don't need to worry about where to put scripts, how to make them available, or how to extend the command-line interface (CLI).

PEAR is a community-driven project written in PHP and shipped with standard PHP distributions.

■ **Tip** The PEAR website, <http://pear.php.net/>, provides documentation and packages grouped by categories.

PEAR is the most professional way to install vendor libraries in PHP. Symfony advises the use of PEAR to keep a central installation point for use across multiple projects. The symfony plug-ins are PEAR packages with a special configuration. The symfony framework itself is available as a PEAR package.

You don't need to know all about the PEAR syntax to use symfony. You just need to understand what it does and have it installed. You can check that PEAR is installed in your computer by typing the following in a CLI:

```
> pear info pear
```

This command will return the version number of your PEAR installation.

The symfony project has its own PEAR repository, or channel. Note that channels are available only since version 1.4.0 of PEAR, so you should upgrade if your version is older. To upgrade your version of PEAR, issue the following command:

```
> pear upgrade PEAR
```

Object-Relational Mapping (ORM)

Databases are *relational*. PHP 5 and symfony are *object-oriented*. In order to access the database in an object-oriented way, an interface translating the object logic to the relational logic is required. This interface is called an object-relational mapping, or ORM.

An ORM is made up of objects that give access to data and keep business rules within themselves.

One benefit of an object/relational abstraction layer is that it prevents you from using a syntax that is specific to a given database. It automatically translates calls to the model objects to SQL queries optimized for the current database.

This means that switching to another database system in the middle of a project is easy. Imagine that you have to write a quick prototype for an application, but the client has not decided yet which database system would best suit his needs. You can start building your application with SQLite, for instance, and switch to MySQL, PostgreSQL, or Oracle when the client is ready to decide. Just change one line in a configuration file, and it works.

An abstraction layer encapsulates the data logic. The rest of the application does not need to know about the SQL queries, and the SQL that accesses the database is easy to find. Developers who specialize in database programming also know clearly where to go.

Using objects instead of records, and classes instead of tables, has another benefit: you can add new accessors to your tables. For instance, if you have a table called `Client` with two fields, `FirstName` and `LastName`, you might like to be able to require just a `Name`. In an object-oriented world, this is as easy as adding a new accessor method to the `Client` class, like this:

```
public function getName()
{
    return $this->getFirstName.' '.$this->getLastName();
}
```

All the repeated data-access functions and the business logic of the data can be maintained within such objects. For instance, consider a class `ShoppingCart` in which you keep items (which are objects). To retrieve the full amount of the shopping cart for the checkout, you can add a `getTotal()` method, like this:

```
public function getTotal()
{
    $total = 0;
    foreach ($this->getItems() as $item)
    {
        $total += $item->getPrice() * $item->getQuantity();
    }
    return $total;
}
```

And that's it. Imagine how long it would have required to write a SQL query doing the same thing!

Propel, another open source project, is currently one of the best object/relational abstraction layers for PHP 5. Symfony integrates Propel seamlessly into the framework, so most of the data manipulation described in this book follows the Propel syntax. This book will describe how to use the Propel objects, but for a more complete reference, a visit to the Propel website (<http://propel.phpdb.org/trac/>) is recommended.

Rapid Application Development (RAD)

Programming web applications has long been a tedious and slow job. Following the usual software engineering life cycles (like the one proposed by the Rational Unified Process, for instance), the development of web applications could not start before a complete set of requirements was written, a lot of Unified Modeling Language (UML) diagrams were drawn, and tons of preliminary documentation were produced. This was due to the general speed of development, to the lack of versatility of programming languages (you had to build, compile, restart, and who knows what else before actually seeing your program run), and most of all, to the fact that clients were quite reasonable and didn't change their minds constantly.

Today, business moves faster, and clients tend to constantly change their minds in the course of the project development. Of course, they expect the development team to adapt to their needs and modify the structure of an application quickly. Fortunately, the use of scripting languages like Perl and PHP makes it easy to apply other programming strategies, such as rapid application development (RAD) or agile software development.

One of the ideas of these methodologies is to start developing as soon as possible so that the client can review a working prototype and offer additional direction. Then the application gets built in an iterative process, releasing increasingly feature-rich versions in short development cycles.

The consequences for the developer are numerous. A developer doesn't need to think about the future when implementing a feature. The method used should be as simple and

straightforward as possible. This is well illustrated by the maxim of the KISS principle: Keep It Simple, Stupid.

When the requirements evolve or when a feature is added, existing code usually has to be partly rewritten. This process is called *refactoring*, and happens a lot in the course of a web application development. Code is moved to other places according to its nature. Duplicated portions of code are refactored to a single place, thus applying the Don't Repeat Yourself (DRY) principle.

And to make sure that the application still runs when it changes constantly, it needs a full set of unit tests that can be automated. If well written, unit tests are a solid way to ensure that nothing is broken by adding or refactoring code. Some development methodologies even stipulate writing tests before coding—that's called test-driven development (TDD).

Note There are many other principles and good habits related to agile development. One of the most effective agile development methodologies is called *Extreme Programming* (abbreviated as XP), and the XP literature will teach you a lot about how to develop an application in a fast and effective way. A good starting place is the XP series books by Kent Beck (Addison-Wesley).

Symfony is the perfect tool for RAD. As a matter of fact, the framework was built by a web agency applying the RAD principle for its own projects. This means that learning to use symfony is not about learning a new language, but more about applying the right reflexes and the best judgment in order to build applications in a more effective way.

The symfony project website proposes a step-by-step tutorial illustrating the development of an application in an agile way. It is called *askeet* (<http://www.symfony-project.com/askeet>), and is recommended reading for those who want to learn more about agile development.

YAML

According to the official YAML website (<http://www.yaml.org/>), YAML is “a straightforward machine parsable data serialization format designed for human readability and interaction with scripting languages.” Put another way, YAML is a very simple language used to describe data in an XML-like way but with a much simpler syntax. It is especially useful to describe data that can be translated into arrays and hashes, like this:

```
$house = array(  
  'family' => array(  
    'name'      => 'Doe',  
    'parents'   => array('John', 'Jane'),  
    'children'  => array('Paul', 'Mark', 'Simone')  
  ),  
  'address' => array(  
    'number'   => 34,  
    'street'   => 'Main Street',  
    'city'     => 'Nowheretown',  
    'zipcode'  => '12345'  
  )  
);
```

This PHP array can be automatically created by parsing the YAML string:

```
house:
  family:
    name: Doe
    parents:
      - John
      - Jane
    children:
      - Paul
      - Mark
      - Simone
  address:
    number: 34
    street: Main Street
    city: Nowheretown
    zipcode: 12345
```

In YAML, structure is shown through indentation, sequence items are denoted by a dash, and key/value pairs within a map are separated by a colon. YAML also has a shorthand syntax to describe the same structure with fewer lines, where arrays are explicitly shown with `[]` and hashes with `{}`. Therefore, the previous YAML data can be written in a shorter way, as follows:

```
house:
  family: { name: Doe, parents: [John, Jane], children: [Paul, Mark, Simone] }
  address: { number: 34, street: Main Street, city: Nowheretown, zipcode: 12345 }
```

YAML is an acronym for Yet Another Markup Language and pronounced “yamel.” The format has been around since 2001, and YAML parsers exist for a large variety of languages.

Tip The specifications of the YAML format are available at <http://www.yaml.org/>.

As you can see, YAML is much faster to write than XML (no more closing tags or explicit quotes), and it is more powerful than `.ini` files (which don’t support hierarchy). That is why symfony uses YAML as the preferred language to store configuration. You will see a lot of YAML files in this book, but it is so straightforward that you probably don’t need to learn more about it.

Summary

Symfony is a PHP 5 web application framework. It adds a new layer on top of the PHP language, providing tools that speed up the development of complex web applications. This book will tell you all about it, and you just need to be familiar with the basic concepts of modern programming to understand it—namely object-oriented programming (OOP), object-relational mapping (ORM), and rapid application development (RAD). The only required technical background is knowledge of PHP 5.