

PHP Object-Oriented Solutions

David Powers



PHP Object-Oriented Solutions

Copyright © 2008 by David Powers

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): 978-1-4302-1011-5

ISBN-13 (electronic): 978-1-4302-1012-2

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.

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 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 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 freely available to readers at www.friendsofed.com in the Downloads section.

Credits

Lead Editor	Production Editor
Ben Renow-Clarke	Laura Esterman

Technical Reviewer	Composer
Seungyeob Choi	Molly Sharp

Editorial Board	Proofreader
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	Patrick Vincent
	Indexer
	Toma Mulligan

Project Manager	Artist
Beth Christmas	April Milne

Copy Editors	Interior and Cover Designer
Heather Lang and Damon Larson	Kurt Krames

Associate Production Director	Manufacturing Director
Kari Brooks-Copony	Tom Debolski

CONTENTS AT A GLANCE

About the Author	xi
About the Technical Reviewer	xiii
Acknowledgments	xv
Introduction	xvii
Chapter 1: Why Object-Oriented PHP?	3
Chapter 2: Writing PHP Classes	23
Chapter 3: Taking the Pain Out of Working with Dates	77
Chapter 4: Using PHP Filters to Validate User Input	121
Chapter 5: Building a Versatile Remote File Connector	169
Chapter 6: SimpleXML—Couldn't Be Simpler	207
Chapter 7: Supercharged Looping with SPL	251
Chapter 8: Generating XML from a Database	289
Chapter 9: Case Study: Creating Your Own RSS Feed	321
Index	355

CONTENTS

About the Author	xi
About the Technical Reviewer	xiii
Acknowledgments	xv
Introduction	xvii
Chapter 1: Why Object-Oriented PHP?	3
Understanding basic OOP concepts	4
How OOP evolved	5
Using classes and objects	6
Protecting data integrity with encapsulation	8
Polymorphism is the name of the game	10
Extending classes through inheritance	10
Deciding on a class hierarchy	11
Using best practice	12
How OOP has evolved in PHP	13
OOP since PHP 5	13
Preparing for PHP 6	14
Choosing the right tools to work with PHP classes	16
Using a specialized script editor	16
Chapter review	19
Chapter 2: Writing PHP Classes	23
Formatting code for readability	25
Using the Zend Framework PHP Coding Standard	25
Choosing descriptive names for clarity	26
Creating classes and objects	26
Defining a class	27
Controlling access to properties and methods	27
Quick review	32
Setting default values with a constructor method	33

CONTENTS

Using inheritance to extend a class	36
Defining a child class	37
Accessing a parent class's methods and properties	39
Using the scope resolution operator	39
Controlling changes to methods and properties	44
Preventing a class or method from being overridden	44
Using class constants for properties	46
Creating static properties and methods	47
Quick review	49
Loading classes automatically	50
Exploring advanced OOP features	51
Creating abstract classes and methods	52
Simulating multiple inheritance with interfaces	54
Understanding which class an object is an instance of	55
Restricting acceptable data with type hinting	56
Using magic methods	59
Converting an object to a string	60
Cloning an object	60
Accessing properties automatically	64
Accessing methods automatically	65
Cleaning up with a destructor method	66
Handling errors with exceptions	67
Throwing an exception	67
Catching an exception	67
Extracting information from an exception	68
Extending the Exception class	72
Using comments to generate code hints	73
Writing PHPDoc comments	74
Chapter review	75

Chapter 3: Taking the Pain Out of Working with Dates 77

Designing the class	78
Examining the built-in date-related classes	79
Using the DateTime class	81
Setting the default time zone in PHP	83
Examining the DateTimeZone class	85
Using the DateTimeZone class	87
Deciding how to extend the existing classes	89
Building the class	91
Creating the class file and constructor	91
Resetting the time and date	95
Accepting dates in common formats	98
Accepting a date in MM/DD/YYYY format	98
Accepting a date in DD/MM/YYYY format	99
Accepting a date in MySQL format	99
Outputting dates in common formats	100
Outputting date parts	101
Performing date-related calculations	103

Adding and subtracting days or weeks	105
Adding months	106
Subtracting months	110
Adding and subtracting years	112
Calculating the number of days between two dates	113
Creating a default date format	114
Creating read-only properties	115
Organizing and commenting the class file	117
Chapter review	118
Chapter 4: Using PHP Filters to Validate User Input	121
Validating input with the filter functions	122
Understanding how the filter functions work	123
filter_has_var()	125
filter_list()	126
filter_id()	126
Setting filter options	127
Filtering single variables	130
Setting flags and options when filtering a single variable	134
Filtering multiple variables	136
Setting a default filter	137
Building the validation class	138
Deciding what the class will do	138
Planning how the class will work	139
Coding the validation class properties and methods	140
Naming properties and defining the constructor	140
Setting the input type and checking required fields	142
Preventing duplicate filters from being applied to a field	147
Creating the validation methods	147
Creating the methods to process the tests and get the results	157
Using the validation class	159
Sticking to your design decisions	165
Chapter review	166
Chapter 5: Building a Versatile Remote File Connector	169
Designing the class	171
Building the class	172
Defining the constructor	172
Checking the URL	174
Retrieving the remote file	180
Defining the accessDirect() method	180
Using cURL to retrieve the remote file	186
Using a socket connection to retrieve the remote file	190
Handling the response headers from a socket connection	196
Generating error messages based on the status code	202
Final testing	204
Ideas for improving the class	204
Chapter review	205

Chapter 6: SimpleXML—Couldn't Be Simpler 207

A quick XML primer	208
What is XML?	208
How XML documents are structured.	210
The rules of writing XML	212
Using HTML entities in XML	213
Inserting HTML and other code in XML	213
Using SimpleXML.	214
Loading an XML document with SimpleXML	217
Loading XML from a file	217
Loading XML from a string	218
Extracting data with SimpleXML	220
Accessing text nodes	221
Accessing attributes.	221
Accessing unknown nodes	222
Saving and modifying XML with SimpleXML.	228
Outputting and saving SimpleXMLElement objects.	228
Modifying SimpleXMLElement objects.	231
Changing the values of text and attributes	231
Removing nodes and values	232
Adding attributes	233
Adding new elements.	234
Using SimpleXML with namespaces	235
How namespaces are used in XML	236
Handling namespace prefixes in SimpleXML	236
Handling namespaced attributes.	241
Finding out which namespaces a document uses.	242
Using SimpleXML with XPath	244
A quick introduction to XPath	244
Using XPath to drill down into XML	245
Using XPath expressions for finer control.	246
Using XPath with namespaces	247
Registering namespaces to work with XPath	247
Chapter review	248

Chapter 7: Supercharged Looping with SPL 251

Introducing iterators	252
Using an array with SPL iterators.	253
Limiting the number of loops with the LimitIterator.	253
Using SimpleXML with an iterator	255
Filtering.	256
Setting options for RegexIterator	259
Looping sequentially through more than one set of data	263
Looking ahead with the CachingIterator.	265
Using anonymous iterators as shorthand	268
Examining files and directories	269
Using DirectoryIterator	270
Including subdirectories in a single operation	271

Extracting file information with SplFileInfo	273
Finding files of a particular type	274
Reading and writing files with SplFileObject	275
Extending iterators	281
Understanding the Iterator interface	282
Extending the FilterIterator class	283
Chapter review	285
Chapter 8: Generating XML from a Database	289
Designing the application	290
Defining the application's purpose.	290
Setting the requirements	292
Building the application	292
Creating the database connection	293
Getting the database result	294
Defining the properties and constructor	295
Implementing the Iterator interface.	296
Implementing the Countable interface	298
Generating the XML output.	302
Defining the properties and constructor	303
Setting the SQL query.	305
Setting the root and top-level node names.	305
Obtaining the primary key	306
Setting output file options	307
Using XMLWriter to generate the output	307
Chapter review	317
Chapter 9: Case Study: Creating Your Own RSS Feed	321
Understanding the RSS 2.0 format	322
The structure of an RSS 2.0 feed	322
What the <channel> element contains	323
What the <item> elements contain	325
Deciding what the feed will contain	326
Building the class.	327
Populating the elements that describe the feed	328
Populating the <item> elements	333
Building the SQL query	334
Creating the <pubDate> element	338
Creating the <link> elements	340
Creating helper methods to format <item> child elements.	344
Generating the XML for the <item> elements	346
Where to go from here	352
Index.	355

ABOUT THE AUTHOR



David Powers is the author of a series of highly successful books on PHP, including *PHP Solutions: Dynamic Web Design Made Easy* (friends of ED, ISBN: 978-1-59059-731-6) and *The Essential Guide to Dreamweaver CS3 with CSS, Ajax, and PHP* (friends of ED, ISBN: 978-1-59059-859-7). As a professional writer, he has been involved in electronic media for more than 30 years, first with BBC radio and television, both in front of the microphone (he was a BBC correspondent in Tokyo from 1987 to 1992) and in senior editorial positions. His clear writing style is valued not only in the English-speaking world—several of his books have been translated into Spanish and Polish.

Since leaving the BBC to work independently, David has devoted most of his time to web development, writing books, and teaching. He is active in several online forums, giving advice and troubleshooting PHP problems. David's expertise was recognized by his designation as an Adobe Community Expert in 2006.

When not pounding the keyboard writing books or dreaming of new ways of using PHP and other programming languages, David enjoys nothing better than visiting his favorite sushi restaurant. He has also translated several plays from Japanese.

ABOUT THE TECHNICAL REVIEWER

Seungyeob Choi is the lead developer and technology manager at Abraham Lincoln University in Los Angeles, where he has been developing various systems for online education. He built the university's learning platform and has been working on a development project for Student Lifecycle Management. Seungyeob has a PhD in computer science from the University of Birmingham, England.

ACKNOWLEDGMENTS

The book you're holding in your hand (or reading on the screen) owes its genesis to a tongue-in-cheek exchange with Steve Fleischer of Flying Tiger Web Design (www.flyingtigerwebdesign.com), who suggested I should write *Powers Object-Oriented PHP*. Actually, he phrased it rather differently. If you take the initial letters of the suggested title, you'll get the drift . . . But Steve had an important point: he felt that books on object-oriented programming (OOP) frequently assumed too much prior knowledge or weren't easily adaptable to PHP in a practical way. If you like what you find in this book, thank Steve for planting the idea in my brain. If you don't like it, blame me, because I'm the one responsible for writing it the way it is.

Thanks must also go to everyone at Apress/friends of ED for helping bring “my baby” into the world. Books are uncannily like real babies. This one took exactly nine months from conception to birth with the expert help of editor Ben Renow-Clarke, project manager Beth Christmas, and many other “midwives.” I owe a particular debt of gratitude to Seungyeob Choi for his perceptive technical review. Seungyeob's eagle eye and deep knowledge of PHP and OOP saved me from several embarrassing mistakes. Any remaining errors are my responsibility alone.

I would also like to thank everyone who has supported me by buying this or any of my previous books. I realize not everyone can afford to buy books, but the royalties from new—not second-hand—books ensure that authors get some reward for all the hard effort that goes into writing. Even the most successful computer books can never aspire to the stratospheric heights of Harry Potter, so every little bit helps—and is much appreciated.

The biggest thanks of all must undoubtedly go to the developers of PHP, who have given the rest of the world a superb programming language that continues to go from strength to strength.

INTRODUCTION

My first experiments with object-oriented programming in PHP took place about six years ago. Unfortunately, the book that introduced me to the subject concentrated on the mechanics of writing classes and paid little heed to principles underlying OOP. As a result, I wrote classes that were closely intertwined with a specific project (“tightly coupled,” to use the OOP terminology). Everything worked exactly the way I wanted, but the design had a fundamental flaw: the classes couldn’t be used for any other project. Worse still, it was a large project—a bilingual, searchable database with more than 15,000 records—so any changes I wanted to make to it involved revising the whole code base.

The purpose of this book is to help you avoid the same mistake. Although most chapters revolve around mini-projects, the classes they use are project-neutral. Rather than being a “how to” cookbook, the aim is to help developers with a solid knowledge of PHP basics add OOP to their skill set.

So, what is OOP? To oversimplify, OOP groups together functions (known in OOP-speak as “methods”) in classes. In effect, a class can be regarded as a function library. What makes OOP more powerful is the fact that classes can be extended to add new functionality. Since many of the new features added to PHP 5 are object-oriented, this means you can easily extend core PHP classes to add new functionality or simply make them work the way you want them to. In fact, Chapter 3 does precisely that: it extends the PHP `DateTime` class to make it easier to use. The project in Chapter 4 takes the PHP filter functions and hides them behind a much more user-friendly interface.

Chapter 5 shows how to create a class that retrieves a text file from a remote server by automatically detecting the most efficient available method. Chapters 6 and 7 cover two of the most important OOP features added to core PHP in version 5: SimpleXML and the Standard PHP Library (SPL). The XML theme continues in the final two chapters, which use the PHP `XMLWriter` class to generate XML on the fly from a database and show you how to create a news feed from your site.

The need for OOP has come about because PHP is being used increasingly for large-scale web applications. Object-oriented practices break down complex operations into simple units, each responsible for a defined task. This makes code much easier to test and maintain. However, ease of maintenance is just as important in small-scale projects, so OOP can play a

role in projects of any size. This is an introductory book, so the object-oriented solutions it contains are designed for use in small projects, but the principles they demonstrate apply equally to large-scale projects.

By the time you have finished this book, you should understand what OOP is and how to write PHP classes that conform to current best practices, making your code easier to maintain and deploy across multiple projects. The information contained in this book will also provide a solid foundation for anyone planning to use an object-oriented framework, such as the Zend Framework (www.zend.com/en/community/framework).

Although everything in this book is devoted to OOP, it's important to emphasize that OOP is only *part* of PHP. OOP helps you create portable, reusable code. Use it where appropriate, but there's no need to throw out all of your existing PHP skills or code.

Another important thing to emphasize is that all the code in this book requires a minimum of PHP 5, and preferably PHP 5.2 or 5.3. It has also been designed to work in PHP 6. *The code will not work in PHP 4*, nor will any support be provided for converting it to PHP 4. Even though at the time of publication, it's estimated that more than half of all PHP-driven websites still run on PHP 4, all support for PHP 4 officially ended on August 8, 2008. PHP 4 is dead. Long live PHP 5 (and PHP 6 when it's released). If you haven't yet made the switch from PHP 4, now is the time to do it.

Who should read this book

If you develop in PHP, but haven't yet got your feet wet with OOP, this is the book for you. No previous knowledge of OOP is necessary: Chapter 1 covers the basic theory and explains how OOP fits into PHP; Chapter 2 then goes into the mechanics of writing object-oriented code in PHP. The remaining seven chapters put all the theory into practice, showing you how to create and use your own classes and objects, as well as covering object-oriented features that have been built into core PHP since version 5.

You don't need to be a PHP expert to follow this book, but you do need to know the basics of writing your own PHP scripts. So, if you're comfortable with concepts such as variables, loops, and arrays, and have ever created a function, you should be fine. Throughout the book, I make extensive use of core PHP functions. In some cases, such as with the filter functions in Chapter 4, I go into considerable detail about how they work, because that knowledge is essential to understanding the chapter. Most of the time, though, I explain what the function is for and why I'm using it. If you want a more in-depth explanation, I expect you to look it up for yourself in the PHP online documentation at <http://docs.php.net/manual/en/>.

The book aims to be a gentle introduction to OOP in PHP, but it moves at a fairly fast pace. The code involved isn't particularly difficult, but it might take a little more time for some of the concepts to sink in. The best way to achieve this is to roll up your sleeves and start coding. Exercises at strategic points demonstrate what a particular section of code does and help reinforce understanding.

Using the download code

All the files necessary to work with this book can be downloaded from the friends of ED website by going to www.friendsofed.com/downloads.html and scrolling down to the link for *PHP Object-Oriented Solutions*. Download the ZIP file, and unzip its contents into a new folder inside your web server document root. I named the folder `OopSolutions`, but you can call it whatever you want. In addition to a series of folders named `ch2_exercises` through `ch9_exercises`, the folder should contain the following:

- `Ch2`: This contains example class definitions for use with `ch2_exercises`.
- `class_docs`: This contains full documentation in HTML format for all the classes developed in the book. Double-click `index.html` to view them in your browser.
- `finished_classes`: This contains a full set of completed class definitions.
- `Pos`: *This folder is empty*. It is where you should create your own versions of the class definitions as you work through each chapter. If you don't want to type out everything yourself, you need to copy each class definition from `finished_classes` to this folder for the files in the exercise folders for each chapter to work.

Understanding the file numbering system

Most download files have a filename ending in an underscore and a number before the `.php` filename extension (e.g., `Book_01.php`, `Book_02.php`). This is because the files represent a class definition or exercise at a particular stage of development.

If you are typing out the exercises and class definitions yourself, leave out the underscore and number (e.g., use `Book.php` instead of `Book_01.php`). Throughout the text, I indicate the number of the current version so you can compare the appropriate supplied version with your own, or simply use it directly if you don't want to type everything yourself.

To get the best out of this book, I strongly urge you to type out all the exercises and class definitions yourself. It's a lot of work, but hands-on practice really does reinforce the learning process.

What to do if things go wrong

Every effort has been made to ensure accuracy, but mistakes do slip through. If something doesn't work the way you expect, your first port of call should be www.friendsofed.com/book.html?isbn=9781430210115. A link to any known corrections since publication will be posted there. If you think you have found a mistake that's not listed, please submit an error report to www.friendsofed.com/errataSubmission.html. When friends of ED has finished with the thumbscrews and forced me to admit I'm wrong, we'll post the details for everyone's benefit on the friends of ED site.

If the answer isn't on the corrections page, scan the chapter subheadings in the table of contents, and try looking up a few related expressions in the index. Also try a quick search

INTRODUCTION

through Google or one of the other large search engines. My apologies if all this sounds obvious, but an amazing number of people spend more time waiting for an answer in an online forum than it would take to go through these simple steps.

If you're still stuck, visit www.friendsofed.com/forums/. Use the following guidelines to help others help you:

- Always check the book's corrections page first. The answer may already be there.
- Search the forum to see if your question has already been answered.
- Give your message a meaningful subject line. It's likely to get a swifter response and may help others with a similar problem.
- Give the name of the book and a page reference to the point that's giving you difficulty.
- "It doesn't work" gives no clue as to the cause. "When I do so and so, x happens" is a lot more informative.
- If you get an error message, say what it contains.
- Be brief and to the point. Don't ask half a dozen questions at once.
- It's often helpful to know your operating system and which version of PHP you're using.
- Don't post the same question simultaneously in several forums. If you find the answer elsewhere, have the courtesy to close the forum thread and post a link to the answer.

Please be realistic in your expectations when asking for help in a free online forum. I'm delighted if you have bought one of my books and will try to help you if you run into problems; but I'm not always available and can't offer unlimited help. If you post hundreds of lines of code, and expect someone else to scour it for mistakes, don't be surprised if you get a rather curt answer or none at all. And if you do get the help that you need, keep the community spirit alive by answering questions that you know the answer to.

Layout conventions

To keep this book as clear and easy to follow as possible, the following text conventions are used throughout.

Important words or concepts are normally highlighted on the first appearance in **bold type**.

Code is presented in `fixed-width font`.

New or changed code is normally presented in **`bold fixed-width font`**.

Pseudocode and variable input are written in *italic fixed-width font*.

Menu commands are written in the form Menu ► Submenu ► Submenu.

Where I want to draw your attention to something, I've highlighted it like this:

Ahem, don't say I didn't warn you.

Sometimes code won't fit on a single line in a book. Where this happens, I use an arrow like this: ➡.

This is a very, very long section of code that should be written all ➡
on the same line without a break.