PHP Objects, Patterns, and Practice

Fourth Edition

Matt Zandstra

Apress®

PHP Objects, Patterns, and Practice, Fourth Edition

Copyright © 2013 by Matt Zandstra

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN-13 (pbk): 978-1-4302-6031-8

ISBN-13 (electronic): 978-1-4302-6032-5

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

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

President and Publisher: Paul Manning

Lead Editor: Ben Renow-Clarke

Development Editor: James Markham

Technical Reviewer: Wes Hunt

Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Louise Corrigan, James DeWolf,

Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham,

Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft,

Gwenan Spearing, Matt Wade, Steve Weiss, Tom Welsh

Coordinating Editor: Anamika Panchoo

Copy Editor: Laura Lawrie

Compositor: SPi Global

Indexer: SPi Global

Artist: SPi Global

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 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. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@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 www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

For Louise, who is the whole point.

Contents at a Glance

About the Author	xxi
About the Technical Reviewer	xxiii
Acknowledgments	xxv
Introduction	xxvii
■ Chapter 1: PHP: Design and Management	1
■ Chapter 2: Objects	7
■ Chapter 3: Object Basics	11
■ Chapter 4: Advanced Features	37
■ Chapter 5: Object Tools.....	81
■ Chapter 6: Objects and Design	111
■ Chapter 7: What Are Design Patterns? Why Use Them?	131
■ Chapter 8: Some Pattern Principles	139
■ Chapter 9: Generating Objects.....	153
■ Chapter 10: Patterns for Flexible Object Programming.....	175
■ Chapter 11: Performing and Representing Tasks	195
■ Chapter 12: Enterprise Patterns	227
■ Chapter 13: Database Patterns.....	277
■ Chapter 14: Good (and Bad) Practice	321
■ Chapter 15: An Introduction to PEAR and Pyrus.....	327
■ Chapter 16: Generating Documentation with phpDocumentor	351

■ **Chapter 17: Version Control with Git365**

■ **Chapter 18: Testing with PHPUnit.....383**

■ **Chapter 19: Automated Build with Phing409**

■ **Chapter 20: Continuous Integration.....431**

■ **Chapter 21: Objects, Patterns, Practice.....457**

■ **Appendix A: Bibliography465**

■ **Appendix B: A Simple Parser.....467**

Index.....489

Contents

About the Author	xxi
About the Technical Reviewer	xxiii
Acknowledgments	xxv
Introduction	xxvii
■ Chapter 1: PHP: Design and Management	1
The Problem	1
PHP and Other Languages.....	2
About This Book	3
Objects.....	3
Patterns	4
Practice.....	4
What's New in the Fourth Edition	5
Summary.....	5
■ Chapter 2: Objects	7
The Accidental Success of PHP Objects.....	7
In the Beginning: PHP/FI.....	7
Syntactic Sugar: PHP 3.....	7
PHP 4 and the Quiet Revolution.....	8
Change Embraced: PHP 5	9
Advocacy and Agnosticism: The Object Debate.....	10
Summary.....	10

■ Chapter 3: Object Basics	11
Classes and Objects	11
A First Class.....	11
A First Object (or Two)	12
Setting Properties in a Class	13
Working with Methods	15
Creating a Constructor Method.....	16
Arguments and Types	18
Primitive Types	18
Taking the Hint: Object Types.....	21
Inheritance	23
The Inheritance Problem	23
Working with Inheritance.....	27
Public, Private, and Protected: Managing Access to Your Classes	31
Summary	35
■ Chapter 4: Advanced Features	37
Static Methods and Properties	37
Constant Properties.....	40
Abstract Classes.....	41
Interfaces	43
Traits.....	45
A Problem for Traits to Solve	45
Defining and Using a Trait.....	46
Using More than One Trait	46
Combining Traits and Interfaces	47
Managing Method Name Conflicts with <code>insteadof</code>	48
Aliasing overridden trait methods	50
Using static methods in traits.....	51
Accessing Host Class Properties	52
Defining Abstract Methods in Traits.....	52
Changing Access Rights to Trait Methods.....	53

Late Static Bindings: The static Keyword	54
Handling Errors	57
Exceptions	58
Final Classes and Methods.....	64
Working with Interceptors.....	66
Defining Destructor Methods	71
Copying Objects with __clone()	72
Defining String Values for Your Objects.....	75
Callbacks, Anonymous Functions and Closures	76
Summary	79
■ Chapter 5: Object Tools.....	81
PHP and Packages	81
PHP Packages and Namespaces	81
Autoload	89
The Class and Object Functions	92
Looking for Classes	93
Learning About an Object or Class.....	93
Getting a Fully Qualified String Reference to a Class	94
Learning About Methods.....	95
Learning About Properties	97
Learning About Inheritance	97
Method Invocation	97
The Reflection API	99
Getting Started	99
Time to Roll Up Your Sleeves	99
Examining a Class	101
Examining Methods.....	103
Examining Method Arguments.....	104
Using the Reflection API	105
Summary	109

■ Chapter 6: Objects and Design	111
Defining Code Design	111
Object-Oriented and Procedural Programming	112
Responsibility	115
Cohesion	116
Coupling.....	116
Orthogonality	116
Choosing Your Classes	116
Polymorphism	117
Encapsulation.....	119
Forget How to Do It.....	120
Four Signposts	120
Code Duplication.....	121
The Class Who Knew Too Much	121
The Jack of All Trades	121
Conditional Statements	121
The UML	121
Class Diagrams.....	122
Sequence Diagrams	127
Summary	129
■ Chapter 7: What Are Design Patterns? Why Use Them?	131
What Are Design Patterns?.....	131
A Design Pattern Overview.....	133
Name	133
The Problem	133
The Solution.....	133
Consequences	134
The Gang of Four Format.....	134

Why Use Design Patterns?	135
A Design Pattern Defines a Problem.....	135
A Design Pattern Defines a Solution.....	135
Design Patterns Are Language Independent	135
Patterns Define a Vocabulary	135
Patterns Are Tried and Tested	136
Patterns Are Designed for Collaboration.....	136
Design Patterns Promote Good Design.....	136
Design Patterns are Used By Popular Frameworks	136
PHP and Design Patterns	136
Summary	137
■ Chapter 8: Some Pattern Principles	139
The Pattern Revelation	139
Composition and Inheritance.....	140
The Problem	140
Using Composition	143
Decoupling	145
The Problem	146
Loosening Your Coupling	147
Code to an Interface, Not to an Implementation.....	149
The Concept That Varies.....	149
Patternitis.....	150
The Patterns	150
Patterns for Generating Objects	150
Patterns for Organizing Objects and Classes	151
Task-Oriented Patterns.....	151
Enterprise Patterns.....	151
Database Patterns	151
Summary.....	151

■ Chapter 9: Generating Objects.....	153
Problems and Solutions in Generating Objects	153
The Singleton Pattern.....	157
The Problem	157
Implementation.....	158
Consequences	159
Factory Method Pattern.....	160
The Problem	160
Implementation.....	162
Consequences	165
Abstract Factory Pattern	165
The Problem	165
Implementation.....	166
Consequences	168
Prototype	169
The Problem	169
Implementation.....	170
But That's Cheating!	173
Summary	174
■ Chapter 10: Patterns for Flexible Object Programming.....	175
Structuring Classes to Allow Flexible Objects	175
The Composite Pattern.....	175
The Problem	176
Implementation.....	178
Consequences	181
Composite in Summary	184
The Decorator Pattern	184
The Problem	184
Implementation.....	187
Consequences	190

The Facade Pattern	190
The Problem	190
Implementation.....	192
Consequences	192
Summary.....	193
■ Chapter 11: Performing and Representing Tasks	195
The Interpreter Pattern.....	195
The Problem	195
Implementation.....	196
Interpreter Issues	203
The Strategy Pattern	204
The Problem	204
Implementation.....	205
The Observer Pattern	208
Implementation.....	210
The Visitor Pattern.....	215
The Problem	215
Implementation.....	216
Visitor Issues	221
The Command Pattern.....	221
The Problem	221
Implementation.....	222
Summary.....	226
■ Chapter 12: Enterprise Patterns	227
Architecture Overview.....	227
The Patterns	227
Applications and Layers	228
Cheating Before We Start	230
Registry	230
Implementation.....	231

The Presentation Layer	240
Front Controller.....	240
Application Controller	249
Page Controller	260
Template View and View Helper	265
The Business Logic Layer.....	267
Transaction Script.....	268
Domain Model.....	272
Summary	275
■ Chapter 13: Database Patterns	277
The Data Layer	277
Data Mapper.....	277
The Problem	278
Implementation.....	278
Consequences	292
Identity Map	292
The Problem	293
Implementation.....	293
Consequences	296
Unit of Work.....	296
The Problem	296
Implementation.....	296
Consequences	300
Lazy Load.....	301
The Problem	301
Implementation.....	301
Consequences	303
Domain Object Factory	303
The Problem	303
Implementation.....	303
Consequences	304

The Identity Object	306
The Problem	306
Implementation.....	307
Consequences	312
The Selection Factory and Update Factory Patterns	312
The Problem	312
Implementation.....	313
Consequences	317
What's Left of Data Mapper Now?	317
Summary	319
■ Chapter 14: Good (and Bad) Practice	321
Beyond Code	321
Borrowing a Wheel	322
Playing Nice	323
Giving Your Code Wings.....	323
Documentation	324
Testing.....	325
Continuous Integration	326
Summary	326
■ Chapter 15: An Introduction to PEAR and Pyrus.....	327
What Is PEAR?	327
Phar Out with Pyrus.....	328
Installing a Package	329
PEAR Channels	331
Using a PEAR Package	333
Handling PEAR Errors	335
Creating Your Own PEAR Package.....	338
package.xml	338
Package Elements	338

The contents Element	340
Dependencies	343
Tweaking Installation with phprelease	344
Preparing a Package for Shipment	345
Setting Up Your Own Channel	345
Summary	349
■ Chapter 16: Generating Documentation with phpDocumentor	351
Why Document?	351
Installation	352
Generating Documentation	352
DocBlock Comments	354
Documenting Classes	356
File-Level Documentation	357
Documenting Properties	358
Documenting Methods	359
Namespace support	360
Creating Links in Documentation	361
Summary	364
■ Chapter 17: Version Control with Git	365
Why Use Version Control?	365
Getting Git	366
Configuring a Git Server	366
Creating the Remote Repository	367
Beginning a Project	369
Cloning the Repository	372

Updating and Committing.....	372
Adding and Removing Files and Directories.....	376
Adding a File.....	376
Removing a File	376
Adding a Directory	377
Removing Directories	377
Tagging a Release	378
Branching a Project.....	378
Summary.....	382
■ Chapter 18: Testing with PHPUnit.....	383
Functional Tests and Unit Tests	383
Testing by Hand	384
Introducing PHPUnit	386
Creating a Test Case	386
Assertion Methods.....	387
Testing Exceptions.....	388
Running Test Suites.....	389
Constraints	390
Mocks and Stubs	392
Tests Succeed When They Fail	394
Writing Web Tests	397
Refactoring a Web Application for Testing	397
Simple Web Testing	399
Introducing Selenium	400
A Note of Caution	406
Summary.....	407

■ Chapter 19: Automated Build with Phing	409
What Is Phing?	409
Getting and Installing Phing	410
Composing the Build Document	410
Targets	412
Properties	414
Types	421
Tasks.....	425
Summary.....	429
■ Chapter 20: Continuous Integration	431
What Is Continuous Integration?	431
Preparing a Project for CI	432
Jenkins.....	444
Installing Jenkins.....	444
Installing Jenkins Plugins.....	446
Setting up the Git Public Key	447
Installing a Project.....	447
Running the First Build	450
Configuring the Reports.....	451
Triggering Builds.....	454
Summary	456
■ Chapter 21: Objects, Patterns, Practice	457
Objects	457
Choice.....	458
Encapsulation and Delegation	458
Decoupling.....	458
Reusability	459
Aesthetics.....	459

Patterns	459
What Patterns Buy Us	460
Patterns and Principles of Design	460
Practice	462
Testing	462
Documentation	462
Version Control	463
Automated Build	463
Continuous Integration	463
What I Missed	463
Summary	464
■ Appendix A: Bibliography	465
Books	465
Articles	465
Sites	466
■ Appendix B: A Simple Parser	467
The Scanner	467
The Parser	475
Index	489

About the Author

Matt Zandstra has worked as a web programmer, consultant, and writer for nearly two decades. He was a senior developer at Yahoo! working both in London and Silicon Valley. He now earns his living as a freelance consultant and writer.

Before writing *PHP Objects Patterns and Practice* he was the author of *SAMS Teach Yourself PHP in 24 Hours* (three editions) and a contributor to *DHTML Unleashed*. He has written articles for *Linux Magazine*, Zend.com, *IBM Developer Works*, and *php|architect Magazine*, among others.

Matt also studies literature and writes fiction. He holds MA degrees in Creative Writing from both Manchester University and the University of East Anglia.

When he's not studying or freelancing at various locations around the UK, Matt lives in Liverpool with his wife, Louise, and two children, Holly and Jake.

About the Technical Reviewer

Wes Hunt has worked as an application developer and UX lead since the late 90s. He is now the founder and lead developer for Armigent, a web service startup in the real estate investment industry. He uses PHP and Scala for rapid development of web and REST services in highly scalable enterprise and startup environments. When not working he's working towards getting a thousand kids coding in Montana this year through CodeMontana.org and advocating programming with the user group MontanaProgrammers.org.

Acknowledgments

As always I have benefited from the support of many people whilst working on this edition. But as always, I must also look back to the book's origins.

I tried out some of this book's underlying concepts in a talk in Brighton, back when we were all first marvelling at the shiny possibilities of PHP 5. Thanks to Andy Budd who hosted the talk, and to the vibrant Brighton developer community. Thanks also to Jessey White-Cinis who was at that meeting and who put me in touch with Martin Streicher at Apress.

Once again the Apress team has provided enormous support, feedback and encouragement. I am lucky to have benefited from such professionalism.

Thanks and love to my wife, Louise, and to our children Holly and Jake for many much needed distractions.

Thanks to Steven Metsker for his kind permission to reimplement in PHP a simplified version of the parser API he presented in his book *Building Parsers in Java*.

I write to music and, in the previous three editions of this book, I remembered the great DJ John Peel, champion of the underground and the eclectic. I must also thank the broadcasters who have taken up his mantle, especially those in the weirder corners of BBC 6 Music and Dandelion Radio.

Introduction

When I first conceived of this book, object-oriented design in PHP was an esoteric topic. The intervening years have not only seen the inexorable rise of PHP as an object-oriented language, but also the march of the framework. Frameworks are incredibly useful, of course. They manage the guts and the glue of many (perhaps, these days, most) Web applications. What's more, they often exemplify precisely the principles of design that this book explores.

There is, though, a danger for developers here, as there is in all useful APIs. This is the fear that one might find oneself relegated to *userland*, forced to wait for remote gurus to fix bugs or add features at their whim. It's a short step from this standpoint to a kind of exile in which one is left regarding the innards of a framework as advanced magic, and one's own work as not much more than a minor adornment stuck up on top of a mighty unknowable infrastructure.

Although I'm an inveterate reinventor of wheels, the thrust of my argument is not that we should all throw away our frameworks and build MVC applications from scratch (at least not always). It is rather that, as developers, we should understand the problems that frameworks solve, and the strategies they use to solve them. We should be able to evaluate frameworks not only functionally but in terms of the design decisions their creators have made, and to judge the quality of their implementations. And yes, when the conditions are right, we should go ahead and build our own spare and focused applications, and, over time, compile our own libraries of reusable code.

I hope this book goes some way toward helping PHP developers maintain design-oriented insight into their platforms and libraries, and provides some of the conceptual tools needed when it's time to go it alone.

I recently took a year or so out to return to school. This is something I recommend for all sorts of reasons. One benefit is the new perspective you gain afterwards when you revisit a once-familiar world. I came back to consulting and found that most of my old clients and contacts had taken the plunge and migrated to Git (I follow suit in this edition). And everyone, but everyone, was calling their development methodology *agile*. However, of my first four new clients, three asked me to help evaluate hastily-built and inflexible codebases. Each project required the retrofitting of unit tests, the creation of basic documentation, and the design of automated build mechanisms before refactoring could safely commence. I found myself drawing heavily on the tools and practices I discuss in the last section of this book. I very much hope you find them as useful and powerful as I do, and they help you to create robust and flexible systems.