



Real-World Solutions for Developing High-Quality PHP Frameworks and Applications

Sebastian Bergmann, Stefan Pribsch

REAL-WORLD SOLUTIONS FOR DEVELOPING HIGH-QUALITY PHP FRAMEWORKS AND APPLICATIONS

FOREWORD	xxi
INTRODUCTION.....	xxiii
 ► PART I FOUNDATIONS	
CHAPTER 1 Software Quality.....	3
CHAPTER 2 Software Testing	15
 ► PART II BEST PRACTICES	
CHAPTER 3 TYPO3: The Agile Future of a Ponderous Project	49
CHAPTER 4 Unit Testing Bad Practices	71
CHAPTER 5 Quality Assurance at Digg Inc.	91
 ► PART III SERVERS AND SERVICES	
CHAPTER 6 Testing Service-Oriented APIs	115
CHAPTER 7 Testing a WebDAV Server	131
 ► PART IV ARCHITECTURE	
CHAPTER 8 Testing symfony and symfony Projects	153
CHAPTER 9 Testing the ezcGraph Component	171
CHAPTER 10 Testing Database Interaction.....	187
 ► PART V Q&A IN THE LARGE	
CHAPTER 11 Quality Assurance at studiVZ	225
CHAPTER 12 Continuous Integration	249
CHAPTER 13 swoodoo: A True Agile Story	281
 ► PART VI NON-FUNCTIONAL ASPECTS	
CHAPTER 14 Usability.....	301
CHAPTER 15 Performance Testing	317

CHAPTER 16 Security 341

CHAPTER 17 Conclusion..... 357

BIBLIOGRAPHY 359

INDEX..... 365

Real-World Solutions for Developing High-Quality PHP Frameworks and Applications

Real-World Solutions for Developing High-Quality PHP Frameworks and Applications

Sebastian Bergmann
Stefan Pribsch



WILEY

Wiley Publishing, Inc.

Real-World Solutions for Developing High-Quality PHP Frameworks and Applications

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2011 by Sebastian Bergmann and Stefan Pribsch

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-87249-9
ISBN: 978-1-118-09822-6
ISBN: 978-1-118-09824-0
ISBN: 978-1-118-09823-3

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Web site may provide or recommendations it may make. Further, readers should be aware that Internet Web sites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Control Number: 2010939958

Trademarks: Wiley, the Wiley logo, Wrox, the Wrox logo, Programmer to Programmer, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

ABOUT THE AUTHORS

SEBASTIAN BERGMANN (thePHP.cc) holds a degree in computer science and is a pioneer in the field of quality assurance in PHP projects. His test framework, PHPUnit, is a de facto standard. He is actively involved in the development of PHP and is the creator of various development tools. Sebastian Bergmann is an internationally sought-after expert. As an author, he shares his long-standing experience in books and articles. He is a frequent speaker at conferences around the world.

STEFAN PRIEBSCH (thePHP.cc) is a co-founder and Principal Consultant with thePHP.cc. He holds a degree in computer science and is the author of various books and technical articles. As a consultant, he helps customers to improve development processes and make better use of PHP, with a focus on software architecture, OOP, design patterns, and tools and methods. Stefan is a frequent speaker at IT conferences around the world.

CREDITS

EXECUTIVE EDITOR

Carol Long

PROJECT EDITOR

Tom Dinse

CONSULTING AND TECHNICAL EDITOR

Elizabeth Naramore

PRODUCTION EDITOR

Daniel Scribner

COPY EDITOR

Gwenette Gaddis

EDITORIAL DIRECTOR

Robyn B. Siesky

EDITORIAL MANAGER

Mary Beth Wakefield

FREELANCER EDITORIAL MANAGER

Rosemarie Graham

ASSOCIATE DIRECTOR OF MARKETING

Ashley Zurcher

PRODUCTION MANAGER

Tim Tate

VICE PRESIDENT AND EXECUTIVE GROUP**PUBLISHER**

Richard Swadley

VICE PRESIDENT AND EXECUTIVE PUBLISHER

Barry Pruett

ASSOCIATE PUBLISHER

Jim Minatel

PROJECT COORDINATOR, COVER

Katherine Crocker

PROOFREADER

Louise Watson, Paul Sagan,
Word One New York

INDEXER

Ron Strauss

COVER DESIGN

Michael E. Trent

COVER IMAGE

© istockphoto.com/Dmitry Mordvintsev

CONTENTS

FOREWORD *xxi*

INTRODUCTION *xxiii*

PART I: FOUNDATIONS

CHAPTER 1: SOFTWARE QUALITY **3**

External Quality **4**

Internal Quality **5**

Technical Debt **5**

Constructive Quality Assurance **7**

Clean Code **8**

 Explicit and Minimal Dependencies 9

 Clear Responsibilities 9

 No Duplication 9

 Short Methods with Few Execution Branches 9

Software Metrics **10**

 Cyclomatic Complexity and npath Complexity 10

 Change Risk Anti-Patterns (CRAP) Index 11

 Non-Mockable Total Recursive Cyclomatic Complexity 11

 Global Mutable State 11

 Cohesion and Coupling 12

Tools **12**

 PHPUnit 12

 phploc 12

 PHP Copy-Paste-Detector (phpcpd) 12

 PHP Dead Code Detector (phpdcd) 13

 PHP_Depend (pdepend) 13

 PHP Mess Detector (phpmd) 13

 PHP_CodeSniffer (phpcs) 13

 bytekit-cli 13

 PHP_CodeBrowser (phpcb) 13

 CruiseControl and phpUnderControl 13

 Hudson 14

 Arbit 14

Conclusion **14**

CHAPTER 2: SOFTWARE TESTING	15
Black Box and White Box Tests	15
How Many Tests Are Needed?	16
System Tests	17
Browser Testing	17
Automated Tests	18
Test Isolation	19
Acceptance Tests	20
Limits of System Tests	20
Unit Tests	21
Return Values	23
Dependencies	24
Side Effects	25
Real-Life Example	25
Analyzing the Code to Test	28
Setting Up a Test Environment	29
Avoid Global Dependencies	31
Test Independently from Data Sources	32
Testing Asynchronous Events	37
Storing Changes in the Database	41
Unpredictable Results	42
Encapsulating Input Data	44
Further Reflections	45
Conclusion	46

PART II: BEST PRACTICES

CHAPTER 3: TYPO3: THE AGILE FUTURE OF A PONDEROUS PROJECT	49
Introduction	49
The History of TYPO3: Thirteen Years in Thirteen Paragraphs	49
Daring to Start Over!	51
Our Experience with Testing	51
Policies and Techniques	52
Bittersweet Elephant Pieces	53
Test-Driven Development	53
Tests as Documentation	54
Continuous Integration	55

Clean Code	56
Refactoring	57
Programming Guidelines	58
Domain-Driven Design	59
Course of Action in Development	60
Developing New Code	60
Extending and Modifying Code	61
Optimizing Code	61
Speed	61
Readability	63
Finding and Fixing Bugs	63
Disposing of Old Code	63
Test Recipes	64
Inadvertently Functional Unit Test	64
Access to the File System	64
Constructors in Interfaces	65
Testing Abstract Classes	66
Testing Protected Methods	66
Use of Callbacks	68
Into the Future	69
CHAPTER 4: UNIT TESTING BAD PRACTICES	71
Why Test Quality Matters	71
Bad Practices and Test Smells	72
Duplication in Test Code	73
Assertion Roulette and Eager Test	74
Fragile Test	76
Obscure Test	78
Problems with Global State	78
Indirect Testing	80
Obscure Test Names	82
Lying Test	83
Slow Test	84
Conditional Logic in Tests	85
Self-validating Tests	87
Web-surfing Tests	87
Mock Overkill	88
Skip Epidemic	90
Conclusion	90

CHAPTER 5: QUALITY ASSURANCE AT DIGG INC.	91
Problems We Are Facing	91
Legacy Code Base	92
How Do We Solve These Problems?	93
Size Does Matter	93
Team Size	94
Project Size	94
Code Size	94
Unit Testing and You	94
Choosing a Testing Framework	95
Working with an Expert	95
One Week in a Room	95
Training Our Team	95
Writing Testable Code	98
Avoid Static Methods	98
Dependency Injection	100
Mock Objects	100
Overview	100
Database	101
Loosely Coupled Dependencies	101
Subject/Observer for Testing Class Internals	102
Memcached	103
Mocking a Service-Oriented Architecture	104
Model	104
Service Query	105
Service Endpoint	105
The Base Classes	105
Digg's Quality Assurance Process	107
Testing	108
Planning the Testing Effort	108
Tasks	108
Automation	108
Benefits	109
Testing Early	109
Testing Often	109
Challenges	110
Conclusion	111

PART III: SERVERS AND SERVICES**CHAPTER 6: TESTING SERVICE-ORIENTED APIS 115**

The Problems	117
Solutions	118
API Credentials	118
API Limits	121
Offline Testing of Service Protocols	122
Offline Testing of Concrete Services	126
Conclusion	130

CHAPTER 7: TESTING A WEBDAV SERVER 131

About the eZ WebDAV Component	131
WebDAV	131
Architecture	133
Development Challenges	135
Requirements Analysis	135
TDD after RFC	136
Testing a Server	137
Automated Acceptance Tests with PHPUnit	139
Capturing Test Trails	140
Test Recipe	141
Integration into PHPUnit	142
A Custom Test Case	142
The Acceptance Test Suite	146
Acceptance Tests by Example	147
Conclusion	149

PART IV: ARCHITECTURE**CHAPTER 8: TESTING SYMFONY AND SYMFONY PROJECTS 153**

Testing a Framework	154
The symfony Release Management Process	154
Long-term Support	154
Code Coverage	155
Tests versus Real Code	155

Running the Test Suite	156
Main Lessons Learned	156
Never Use the Singleton Design Pattern in PHP	156
Decouple Your Code with Dependency Injection	158
Lower the Number of Dependencies between Objects with an Event Dispatcher	159
Testing Web Applications	161
Lowering the Barrier of Entry of Testing	161
Unit Tests	162
Easy to Install	162
Easy to Learn	163
Fun to Use	165
Functional Tests	165
The Browser Simulator	166
The Fixtures	168
The CSS3 Selectors	168
Testing Forms	169
Debugging	169
Conclusion	170
CHAPTER 9: TESTING THE EZCGRAPH COMPONENT	171
Development Philosophy	172
Graph Component	172
Architecture	173
Test Requirements	174
Driver Mocking	175
Mock the Driver	175
Multiple Assertions	176
Structs	177
Expectation Generation	178
Conclusion	178
Testing Binary Data	179
The Drivers	179
Expectation Generation	179
SVG	180
XML Comparison	180
Floating-point Problems	181
Bitmap Creation	181
Bitmap Comparison	182
GD Version Differences	183

Flash	183
The Assertion	184
Conclusion	185
CHAPTER 10: TESTING DATABASE INTERACTION	187
Introduction	187
Reasons Not to Write Database Tests	188
Why We Should Write Database Tests	189
What We Should Test	190
Writing Tests: Mocking Database Connections	191
Writing Tests: PHPUnit Database Extension	191
The Database Test Case Class	192
Establishing the Test Database Connection	193
Creating Data Sets	196
XML Data Sets	197
Flat XML Data Sets	199
CSV Data Sets	200
YAML Data Sets	201
Database Data Sets	203
Data Set Decorators	204
Generating Data Sets	209
Data Operations	209
Creating Tests	211
Testing the Loading of Data	211
Testing the Modification of Data	215
Using the Database Tester	218
Applying Test-Driven Design to Database Testing	220
Using Database Tests for Regression Testing	220
Testing Problems with Data	221
Testing Problems Revealed by Data	222
Conclusion	222
PART V: Q&A IN THE LARGE	
CHAPTER 11: QUALITY ASSURANCE AT STUDIVZ	225
Introduction	225
About studiVZ	226
Acceptance Tests	227
Acceptance Tests in Agile Environments	227

Selenium	228
The Selenium Extension of PHPUnit	229
The Technical Setup of studiVZ	230
Development Environment	230
Test Environment	231
Best Practices	232
Sins of Our Youth	232
Monolithic Tests	232
Static Users	233
Strategy Change	234
Atomic Tests with Dynamic Test Data	234
Robust Selenium Tests	235
Test Scope Must Be Clear	235
Common Functionality or Browser Compatibility as Well?	236
Fix Tests Right Away!	236
Stabilize Locators, and Use IDs	237
Speed, the Sore Subject	238
Recipes for Last-Minute Features	239
Tests Are Software Too	240
Capture and Replay versus Programming Tests	240
The Team: A Good Mix	242
We Need a DSL	242
Internal DSL	243
Testing_SeleniumDSL 1.0	243
Problem: Context Sensitivity	245
Testing_SeleniumDSL 2.0 — A Draft	245
State and Outlook on Version 2.0	246
Conclusion	246
 CHAPTER 12: CONTINUOUS INTEGRATION	 249
Introduction	249
Continuous Integration	251
Configuration	251
Build Management and Automated Tests	251
Version Management	252
Continuous Integration	252
Static Analysis	253
Code Clones	253
Refactoring	253
Software Metrics	254

Classic Metrics	255
Object-Oriented Metrics	259
RATS	262
Installation	263
Configuration	264
Static Tests	266
Programming Conventions	266
Coding Guidelines	268
Gradual Introduction into Legacy Projects	269
Coding Standards in the Daily Work	270
Syntax Analysis	271
Dynamic Tests	272
Reporting	272
Notification in the Case of Errors	272
Statistics	272
PHP_CodeBrowser	273
Deliverables	274
Operations	275
Advanced Topics	276
Continuous Deployment	276
Using a Reverse Proxy	277
Continuous Integration and Agile Paradigms	278
Conclusion	278
CHAPTER 13: SWOODOO: A TRUE AGILE STORY	281
Introduction	281
Evolution: Only the Strong Survive	282
How We Reached the eXtreme Side	285
And While We Are Working...	288
The Art of Evolution	292
Lack of Experience	293
The Java-developer-coding-in-PHP Phenomenon	294
The Nobody-but-me-understands-my-code Developer	296
Conclusion	298
PART VI: NON-FUNCTIONAL ASPECTS	
CHAPTER 14: USABILITY	301
Anything Goes, But What Is the Price?	303
Design Aspects	304

Accessibility	304
Readability	304
Labels for Form Elements	305
Navigating by Keyboard	305
Effective Contrast	306
Logo Links to Home Page	307
Alternative Texts for Images	307
Background Image in Background Color	307
Usable Print Version	307
Visible Links	307
Good Bookmarks	307
No Frames	308
Scalable Fonts	308
Technical Aspects	308
Performance	308
Semantic Code	309
Fewer Requests	309
CSS Sprites	309
JavaScript on Bottom, CSS on Top	310
Link CSS Instead of Importing	310
JavaScript	310
User Guidance	310
The “Fold” Myth	311
Feedback on Interaction	311
Navigation	312
Pop-ups and Other Annoyances	312
Habits and Expectations	313
Fault Tolerance and Feedback	313
Testing Usability	313
Conclusion	315

CHAPTER 15: PERFORMANCE TESTING 317

Introduction	317
Tools	318
Environmental Considerations	319
Load Testing	320
Apache Bench	321
Pylot	322
Other Load Testing Tools	324

Profiling	324
Callgrind	325
KCachegrind	328
APD	329
Xdebug	330
XHProf	331
OPProfile	333
System Metrics	334
strace	334
Sysstat	335
Custom Instrumentation	337
Common Pitfalls	338
Development versus Production Environments	338
CPU Time	338
Micro-Optimizations	338
PHP as the Glue	339
Priority of Optimization	339
Conclusion	340
CHAPTER 16: SECURITY	341
What Is Security?	341
Secure by Design	342
Operations	342
Physical Access	343
Software Development	344
No Security by Obscurity	344
Separation of Concerns	344
A Matter of Rights	345
Error Handling	345
Basic Settings	346
What Does Security Cost?	346
The Most Common Problems	347
A10: Unvalidated Redirects and Forwards	347
A9: Insufficient Transport Layer Protection	348
A8: Failure to Restrict URL Access	349
A7: Insecure Cryptographic Storage	349
A6: Security Misconfiguration	350
A5: Cross Site Request Forgery (CSRF/XSRF)	351
A4: Insecure Direct Object References	351

A3: Broken Authentication and Session Management	352
A2: Cross-Site Scripting (XSS)	353
A1: Injection	354
Conclusion	355
CHAPTER 17: CONCLUSION	357
Bibliography	359
<i>INDEX</i>	365

FOREWORD

Building and assuring quality software is not a new concept, and few will argue it is not important. I have had the privilege of building truly mission-critical operational software for many years—the kind where people’s lives can be at stake. During that time, I learned lots about how to implement and drive a quality process from project inception to mission-critical use. Creating a high-quality process is not trivial, requires the support and commitment of the organization’s leadership, and can impact choice of people, systems, processes, communications, and even organizational structures.

In my opinion, the challenges of the Internet’s broad reach and pace dwarf the challenges of the mission-critical systems I was building. While many of these new systems are “only” business-critical, the truth is that they are no less critical and are dealing with additional layers of complexity such as more distributed development teams, well-known and evolving security attacks on web standards and software, internationalization challenges, shorter release cycles in SaaS settings, and more. In addition, in e-commerce applications, where downtime directly equates to money, the requirement for a strong quality-assurance program is even more critical and requires a special emphasis on compliance, quick time to fix (and time to verify and deploy that fix), and ability to run real-time end-to-end transactions to ensure not only that the application is up, but also that transactions can actually occur. In addition, the increasing emphasis on user experience also means that perceived quality has become increasingly critical and a functioning system that is not delivering the desired user experience has to be enhanced within a very short time frame without compromising on quality. The quality process and systems have to support such rapid turnaround of changes at high quality.

Suffice to say that these challenges have led to significant innovation and changes in approach when it comes to quality assurance compared with the well-established past practices of building mission-critical software. Software development has made huge strides over the past years in establishing strong best practices and awareness around quality assurance. Some key advances include a recognition that the developers must be a huge part of the quality process and cannot defer sole responsibility to the quality team. Continuous integration methodology mitigates one of the biggest challenges and bottlenecks in pushing out quality software—the integration stage. A more strategic focus on automated testing enables pushing out fixes faster, and not only meeting agreed-upon SLAs but exceeding SLAs to deliver end-user satisfaction.

This book puts a strong focus on many of the required quality-assurance disciplines with a very practical focus on PHP and covers people, systems, processes, and tools. The authors of this book have a great mix of both practical and theoretical knowledge of the subject, and I cannot think of better authors to write such a book. Not only do they have practical experience from a diverse set of real-life projects, but they also have created and contributed to quality tools within the PHP eco-system. In addition, the focus on case studies is invaluable as a way to learn from others and understand how everyone tailors his best practices to his situation.

I am sure this book will serve you well in taking the quality of your projects to the next level and help you instill an even stronger sense of pride in the software you are creating within your teams and your management.

—ANDI GUTMANS
MENLO PARK, CA
February 2010

INTRODUCTION

Experience: that most brutal of teachers. But you learn, my God do you learn.

— C.S. LEWIS

ABOUT THIS BOOK

According to the TIOBE Programming Community Index, PHP is the most popular programming language after C/C++ and Java.¹ Gartner predicts that dynamic programming languages will be critical to the success of many next-generation application development efforts and sees PHP as one of the strongest representatives of this type of programming language.² Since the beginning, PHP was designed for web application development and was likely one of the driving forces behind the dot-com boom at the turn of the millennium. Since then, PHP has matured to a general-purpose programming language that supports both procedural and object-oriented programming. In the past, subjects such as performance, scalability, and security were hot in the PHP community. In recent years, however, architecture and quality are getting more attention. In our consulting practice, we see more enterprises that want to modernize their PHP-based software and to base their development processes on agile values. The modernization of a code base is usually driven by a migration from PHP4 to PHP5 or by the introduction of a framework to standardize development.

Against this backdrop, it is hardly surprising that a plethora of PHP frameworks exists. All these frameworks want to help with solving recurring use cases and the standardization of application development. Dynamic and static testing techniques as well as automated builds and continuous integration are no longer alien concepts to PHP developers. Especially in enterprise-critical applications, simple PHP programming has evolved into software engineering with PHP.

Is This a PHP Book?

Based on examples from the PHP world, this book teaches the planning, execution, and automation of tests for the different software layers, the measuring of software quality using software metrics, and the application of appropriate practices such as continuous integration. We assume the reader is either an experienced PHP developer and interested in quality assurance for PHP projects or a developer who is proficient enough with another programming language to follow the examples.

¹TIOBE Software BV, “TIOBE Programming Community Index for December 2010,” accessed December, 2010, <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

²Gartner Inc., “Dynamic Programming Languages Will Be Critical to the Success of Many Next-Generation AD Efforts,” 2008, accessed April 10, 2010, http://www.gartner.com/DisplayDocument?ref=g_search&id=832417.

This book cannot and does not want to be an introduction to (object-oriented) programming with PHP5. And although many companies think about quality-assurance measures for the first time while migrating from PHP4 to PHP5, topics related to migration of PHP applications and environments are not covered in this book. For those topics, refer to *Professionelle Softwareentwicklung mit PHP 5: Objektorientierung Entwurfsmuster, Modellierung und fortgeschrittene Datenbankprogrammierung* by Sebastian Bergmann (dpunkt.verlag, 2005, ISBN 978-3-89864-229-3) and *PHP migrieren: Konzepte und Lösungen zur Migration von PHPAnwendungen und -Umgebungen* by Stefan Pribsch (Carl Hanser Verlag, 2008, ISBN 978-3-446-41394-8).

In addition to developers, project leaders and quality-assurance engineers should be concerned with the topic of quality assurance. We hope this book fosters mutual trust among all stakeholders of a software project and motivates all readers to improve the internal quality of their software (see the “Internal Quality” section in Chapter 1).

Structure of the Book

Following the idea that we learn best through experience—and especially from the experience of others—this book brings together case studies that allow a look behind the scenes of well-known enterprises and projects and imparts valuable practical experience.

The first part, “Foundations,” explains how we define and understand software quality and how you can test the different layers of software. Part II, “Best Practices,” shows tried and true approaches and strategies (for instance, with regard to the writing of unit tests) and how the developers of Digg Inc. and the TYPO3 project implement them. The “Unit Testing Bad Practices” chapter captures the same topic from a different angle and shows the pitfalls you should avoid while writing unit tests.

Part III, “Servers and Services,” discusses testing service-oriented APIs and server components. Part IV, “Architecture,” uses Symfony as an example to show how both a framework itself and applications built using it can be tested. Using the Graph component from eZComponents, we discuss how a good architecture of loosely coupled objects can enable even the testing of binary image data. Testing database interaction is a topic that affects multiple layers of an application’s architecture and thus is worth its own chapter.

In Part V, “Q&A in the Large,” the developers of studiVZ and swoodoo report on their experience with quality assurance in large projects and teams. Chapter 12, “Continuous Integration,” brings together dynamic and static testing techniques and shows how the various quality-assurance tools can be effectively used together.

The last part, “Non-Functional Aspects,” tops off the book with chapters covering usability, performance, and security.

ABOUT THE CASE STUDY AUTHORS

Robert Lemke, TYPO3 Association, and Karsten Dambekalns, TYPO3 Association

Robert Lemke is co-founder of the TYPO3 Association and leads the development of TYPO3 v5 (code name Phoenix) and FLOW3. He's passionate about agile development methods and clean code. Robert lives in Lübeck, with his wife Heike, his daughter Smilla, and Vibiemme, their espresso machine.

Karsten Dambekalns, learned the basics of web technology the hard way—by looking at other websites' HTML source code. After using OS/2, Windows, and Linux, he now uses a Mac. All this happened after he learned BASIC and Assembler on a good old Commodore C128. Using PHP since 1999, Karsten discovered TYPO3 in 2002 and got caught by its immense possibilities. Now he is part of the TYPO3 Phoenix and FLOW3 core development teams and is a Steering Committee member of the TYPO3 Association.

In their chapter, “TYPO3: The Agile Future of a Ponderous Project,” Robert Lemke and Karsten Dambekalns show foundations and techniques that the TYPO3 project uses to improve the quality of their software in a sustainable way.

Benjamin Eberlei, direkt:effekt GmbH

Benjamin Eberlei is software developer at direkt:effekt GmbH. In his spare time, he is part of the Doctrine team, maintains some components for the Zend Framework, and contributes to a handful of other small open-source projects.

In his “Unit Testing Bad Practices” chapter, Benjamin Eberlei shows common mistakes you should avoid while writing unit tests in order to maximize the return on investment from automated testing of your software.

Matthew Weier O'Phinney, Zend Technologies Ltd.

Matthew Weier O'Phinney is Project Lead for Zend Framework and has been contributing to the project since before the initial preview release. He is an advocate for open-source software and regularly blogs and speaks on PHP best practice topics. You'll find him most often in Vermont, where he resides with his wife, daughter, son, and aging basset hound.

In his “Testing Service-Oriented APIs” chapter, Matthew Weier O'Phinney highlights the challenges of testing web services and presents solutions that proved successful in the Zend Framework project.

Tobias Schlitt

Tobias Schlitt has a degree in computer science and has worked for more than 10 years on professional web projects using PHP. As an open-source enthusiast, he contributes to various community projects. Tobias is a co-founder of Qafoo GmbH, which provides services for high-quality PHP development. This includes consulting and training for quality assurance and better programming, as well as technical support for several PHP QA tools.

In his “Testing a WebDAV Server” chapter, Tobias Schlitt shows that you sometimes have to resort to unusual methods to reach your goals when writing automated tests.

Fabien Potencier, Sensio Labs

Fabien Potencier³ discovered the Web in 1994, at a time when connecting to the Internet was still associated with the harmful, strident sounds of a modem. Being a developer by passion, he immediately started to build websites with Perl. But with the release of PHP5, he decided to switch focus to PHP and created the Symfony framework⁴ project in 2004 to help his company leverage the power of PHP for its customers. Fabien is a serial entrepreneur, and among other companies, he created Sensio, a services and consulting company specializing in web technologies and Internet marketing, in 1998. Fabien is also the creator of several other open-source projects, a writer, blogger, speaker at international conferences, and happy father of two wonderful kids.

In his “Testing Symfony and Symfony Projects” chapter, Fabien Potencier reports his experience from the Symfony project and shows how the testing of Symfony improved the framework’s programming interfaces.

Kore Nordmann

Kore Nordmann has a degree in computer science. During his studies, he worked as a developer and software architect at eZ Systems, the manufacturer of the leading enterprise open-source CMS. In addition, he is developing and/or maintaining various open-source projects, such as eZComponents, Arbit, WCV, Image 3D, PHPUnit, and others. For several years, Kore has been a regular speaker at national and international conferences and has published several books and articles. He offers consulting as a partner in Qafoo GmbH.

In his “Testing the ezcGraph Component” chapter, Kore Nordmann describes how a well-designed architecture, together with the use of mock objects, enables even the testing of a component that produces binary output.

Michael Lively Jr., Selling Source LLC

Michael Lively has been working with PHP since 2001 and has been involved to some degree with the PHP testing community since 2005. He is the creator of the database testing extension for PHPUnit and makes other small contributions to PHPUnit. He is now an Application Architect for

³<http://fabien.potencier.org>

⁴<http://www.symfony-project.org>

Las Vegas-based Selling Source LLC. While working with Selling Source, he has worked on several projects, including enterprise-level loan management and processing platforms written in PHP that service millions of customers and hundreds of agents.

In his “Testing Database Interaction” chapter, Michael Lively Jr. documents the functionality of DbUnit, PHPUnit’s extension for database testing, and shows how this powerful tool can be leveraged effectively.

Christiane Philipps, Rebate Networks GmbH, and Max Horváth, Vodafone GmbH

Christiane Philipps is CTO and Agile Enthusiast with Rebate Networks. She regularly writes about agile testing and agile leadership, topics that are dear to her heart, in her blog.⁵

Max Horváth is Lead Software Engineer with Vodafone Internet Services and has more than 10 years of experience with web development. At the time of this writing, he was the Team Lead Mobile Development at VZnet Netzwerke.

In their “Quality Assurance at studiVZ” chapter, Christiane Philipps and Max Horváth report how they successfully introduced PHPUnit and Selenium RC for one of Europe’s largest social networking platforms.

Manuel Pichler, OnVista Media GmbH, and Sebastian Nohn, Ligatus GmbH

Manuel Pichler is the creator of PHP quality assurance tools such as PHP Depend,⁶ PHPMD,⁷ and phpUnderControl.⁸ He is a co-founder of Qafoo GmbH, which provides services for high-quality PHP development.

Sebastian Nohn started developing dynamic websites in 1996 and has handled quality assurance in commercial and open-source projects since 2002. He was one of the first to use CruiseControl for the continuous integration of PHP projects.

In their chapter, “Continuous Integration with phpUnderControl,” Manuel Pichler and Sebastian Nohn report how continuous integration, retroactively written unit tests, software metrics, and other static testing techniques helped improve the software quality of a legacy application.

Lars Jankowsky, swoodo AG

Lars Jankowsky is the CTO of swoodo AG and is responsible for the PHP-based flight and hotel price comparison service. He has been developing web applications for over 15 years and has used PHP since its early versions. Another passion of his is leading eXtreme Programming teams.

⁵<http://agile-qa.de>

⁶<http://pdepend.org/>

⁷<http://phpmd.org/>

⁸<http://phpUnderControl.org/>

In his chapter, “swoodoo: A True Agile Story,” Lars Jankowsky tells how swoodoo introduced agile methods and a service-oriented architecture to allow for the smooth and continuous evolution of their product.

Jens Grochtdreis

Jens Grochtdreis⁹ is a self-employed web developer and consultant who specializes in front-end development and accessibility.

In his “Usability” chapter, Jens Grochtdreis shows how to develop websites that are comprehensible and easy to use, as well as how to test for usability.

Brian Shire

Brian Shire started programming around age eight on an Apple IIe. When he wasn’t playing games, he was learning the Basic programming language. At the time of this writing, Brian was working for Facebook, Inc., where he focused on scaling their PHP infrastructure. In his four years with Facebook, the site grew from 5 million to over 175 million users. During this time, Brian became a primary contributor to APC, an opcode and user variable cache for PHP. He also made contributions to PHP itself and other PECL extensions. Brian has shared some of this knowledge and experience as a speaker at various conferences around the world. He currently resides in San Francisco, California, and maintains a personal and technical blog.¹⁰

In his “Performance Testing” chapter, Brian Shire provides motivation for performance testing of web applications and introduces the reader to the appropriate tools and processes for performance testing.

Arne Blankerts, thePHP.cc

Arne Blankerts has long-standing experience as Head of IT. His software fCMS makes innovative use of XML technologies and is vital to business-critical applications in international corporations. He is actively involved with the documentation of PHP. Arne Blankerts is an expert on IT security and writes about this in a magazine column. He is a sought-after speaker at international conferences and a book author, and he publishes articles in IT magazines.

In his “Security” chapter, Arne Blankerts shows how easy the development of fundamentally secure applications is if you know the common attack vectors and follow some important rules.

ERRATA

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, like a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata, you might save another reader hours of frustration, and at the same time, you will be helping us provide even higher-quality information.

⁹<http://grochtdreis.de>

¹⁰<http://tekrat.com>