Community Experience Distilled

PHPUnit Essentials

Get started with PHPUnit and learn how to write and test code using advanced technologies

Zdenek Machek

[PACKT] PUBLISHING open source* community experience distilled

PHPUnit Essentials

Get started with PHPUnit and learn how to write and test code using advanced technologies

Zdenek Machek

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

PHPUnit Essentials

First published: May 2014

Production Reference: 1190514

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78328-343-9

www.packtpub.com

Cover Image by Asher Wishkerman (wishkerman@hotmail.com)

Credits

Author
Zdenek Machek

Reviewers
R. Rajesh Jeba Anbiah
Azizur Rahman
Mauro Takeda
Jakub Tománek
M. A. Hossain Tonu
Aaron Saray
Dmytro Zavalkin

Commissioning Editors
Amarabha Banerjee
Usha Iyer

Acquisition Editor
Amarabha Banerjee

Content Development Editor
Arvind Koul

Technical Editors
Shweta S. Pant
Ritika Singh

Copy Editors
Sarang Chari
Janbal Dharmaraj
Mradula Hegde
Deepa Nambiar
Alfida Paiva
Adithi Shetty

Project Coordinator
Kranti Berde

Proofreaders
Simran Bhogal
Maria Gould
Linda Morris

Indexer
Mehreen Deshmukh

Production Coordinator
Manu Joseph

Cover Work
Manu Joseph

About the Author

Zdenek Machek is an experienced PHP developer, who has been working with PHP since the year 2000 and PHP3 days. He introduced software testing and PHPUnit to various companies, and used them on various small as well as large scale projects.

Zdenek wrote several articles and blog posts focused on Continuous Integration processes during PHP application development.

Currently, Zdenek leads technology standards and values across the organization, and also handles analysis, planning, and technical delivery of large scale, critical, and high performance systems for our most complex projects.

First and foremost, I would like to thank all the people who invest their time and energy in open source software—people writing code, documentation, reporting and fixing bugs, and organizing meetings. They allowed PHP to become so popular and widely used. The same can be said about PHPUnit and all other related projects, which really transformed the way developers work and probably how the Internet looks and works today.

I would like to thank all the people who helped me and supported me through all my life. Especially, I would like to thank my family for their patience for all the hours, days, and years that I have spent in front of a computer screen.

I would also like to thank the people at Packt Publishing, who have been working on this project, for their valuable feedback and professional approach, as well as the reviewers who helped to improve the book with their excellent comments and suggestions.

About the Reviewers

R. Rajesh Jeba Anbiah has been into programming since 1998 with C, Visual Basic, Delphi, Perl, PHP, ASP.NET MVC, and Play. He currently heads the project division in Agriya (`http://www.agriya.com/`), where he oversees challenging web and mobile development projects, web software products, and lab initiatives. These days, he's more passionate about topics related to machine learning and REST-based SPA using Node.js, AngularJS, and upcoming frameworks.

So far he has authored two books:

- *A to Z of C, Packt Publishing*
- *PHP Ajax Cookbook, Packt Publishing*

When he's not in the office, he is mostly available at home nagging his wife and two children.

Azizur Rahman is a senior web developer at the BBC. Currently, he's working on the BBC Homepage project. He has a BSc (Hons) Artificial Intelligence degree from the University of Westminster, UK.

He joined BBC in late 2011 as a web application developer within the Future Media Knowledge and Learning department. While working on the Knowledge & Learning project (`http://www.bbc.co.uk/education`), he put his knowledge of Test-Driven Development (TDD) using PHPUnit into practice. In this large scale project, the development team also used Ruby and Cucumber for fully automated acceptance testing.

The Knowledge & Learning project's purpose was to bring together factual and learning content from over 100 existing BBC websites, from Bitesize, Food, Science to History, and place them into a single consistent user experience designed to make learning feel effortless and delightful.

After the successful launch of the Knowledge & Learning project, he moved to the Interactive Guides project (`http://www.bbc.co.uk/guides`). While developing Interactive Guides, he solidified and advanced his knowledge of TDD and Behavior-Driven Development (BDD).

The Interactive Guides project took a different approach to presenting content compared to traditional web articles of TV and radio programs online. They organize video and audio, rich infographics, written summaries, and activities into stories that make the most out of BBC's interactive medium. Interactive Guides takes the audience through a series of steps that ask them to look at multiple perspectives of intriguing questions, always with the chance to reflect on the significance of the story at the end.

A firm believer in philanthropy, Azizur spends his spare time supporting philanthropic causes using his knowledge and expertise of open source technologies.

He serves as a senior web developer / IT advisor to ProductiveMuslim.com—a website dedicated to Islam and productivity. A global team of volunteers with the sole aim to help the Ummah ("nation" or "community") become productive again!

The following is a quote by Prophet Muhammad (PBUH) taken from `http://sunnah.com/urn/1252420`:

> "*The best of charity is when a [Muslim] man gains knowledge, then he teaches it to his [Muslim] brother.*"

In April 2014, he became an ambassador of STEMNET. As an ambassador, he uses his enthusiasm and commitment to encourage young people to enjoy science, technology, engineering, and mathematics. You can read more about STEMNET at `http://www.stemnet.org.uk`.

Along with technically reviewing this book, he has also technically reviewed *PHP Application Development with NetBeans: Beginner's Guide*, *Packt Publishing*.

His keen interest in open source software makes him a regular attendee at local technology user groups and wider open source community events.

Mauro Takeda has been working in the IT industry since 1999 when he faced his first legacy problem: the Y2K bug. Since then, he has worked with several programming languages, such as COBOL, Dataflex, C, Visual Basic, Delphi, Pascal, Lisp, Prolog, and Java with whom he has a relationship of over 10 years. His newer passion is functional programming.

In the last five years, he has worked in CI&T (`www.ciandt.com`), a global IT company headquartered in Campinas, Brazil (a prominent tech center considered as the Brazilian Silicon Valley) with strategic locations across Latin America, North America, Europe, and the Asia-Pacific. Nowadays, as a systems architect, he is responsible for people and software development, mainly in PHP and Drupal.

For Helena, Henrique, and Márcia, whose smiles make me happy even when nothing else does.

Jakub Tománek is a seasoned (more than eight years' experience) PHP developer. After having started working on regular websites, he quickly focused on complex web applications, including creating APIs, background jobs, vast databases, and fixing untraceable bugs.

He currently works as a Senior Software Development Engineer in Microsoft, where his team maintains and improves Skype's website.

I'd like to thank my parents for all they've done for me in the past. I wouldn't be where I am without their support. I also want to thank all my current and past colleagues from whom I had the opportunity to learn so much.

M. A. Hossain Tonu, the author of the book *PHP Application Development with NetBeans: Beginner's Guide*, *Packt Publishing*, graduated in Computer Science and Engineering from the Dhaka University of Engineering and Technology (DUET) in Bangladesh.

He is working at Vantage, Dhaka (`http://www.vantage.com/`), where he is leading and maintaining a highly available SAAS platform Vantage CRM that is the single most intuitive and easy-to-use Customer Relationship Management system (`http://www.vantageip.com/products-services/vantage-crm/`) on the market.

He has been a passionate developer for the past eight years, and has developed a series of web applications, services, and solutions for leading software companies in the country, such as somewherein and Improsys.

You can reach Tonu at `mahtonu@vantage.com` and at his tech blog at `http://mahtonu.wordpress.com`.

Aaron Saray knows exactly what it's like to have code under the microscope. PHP conference presentations, open source contributions, and managing a team that now works in his old code, are things that have made him used to the constant code reviews. While he enjoys learning the newest in web technologies, more than a decade later, his true passion is still the core and basics of PHP programming, testing, and best practices. He reflects this in his WROX book *Professional PHP Design Patterns* and his technical blog at `aaronsaray.com/blog`. Additionally, he can be found as a technical editor of many books encompassing PHP, JavaScript, and Internet technologies.

Dmytro Zavalkin has around seven years of experience in the field of Web Development using LAMP stack. For the last three years, he has been using PHPUnit in his everyday work. Currently, he works as a PHP/Magento developer at AOE GmbH, Wiesbaden, Germany. Before relocating to Germany, he worked at Magento, an eBay Inc. company in Donetsk, Ukraine.

www.PacktPub.com

Support files, eBooks, discount offers, and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface **1**

Chapter 1: Installing PHPUnit **9**

Requirements **10**

Running PHP from the command line 11

Composer – the dependency manager for PHP **12**

Installing Composer 13

Installation 13

Local installation 14

System-wide installation 14

Installing PEAR **15**

PHPUnit installation 16

Other installation methods **17**

Installing the Linux package 17

Manual installation 18

Testing the installation **18**

Xdebug 21

Installing Xdebug 21

Summary **23**

Chapter 2: PHPUnit Support in IDEs **25**

IDEs and PHPUnit **26**

NetBeans 27

Zend Studio 31

Eclipse PDT 34

Installing MakeGood 35

Creating your FirstTest 39

PhpStorm 41

Summary **45**

Chapter 3: Tests and What They're All About 47
Understanding unit testing 48
What is a unit test? 48
Assertions 48
The importance of unit testing 49
Testing all possible scenarios 49
What makes a good test? 50
When to write tests 50
Anatomy of a PHPUnit test 51
Defining test methods 53
Testing functions 53
Testing methods 56
The MVC application architecture and tests 61
Testing controllers 62
Summary 64
Chapter 4: Testing Dependencies and Exceptions 67
Detecting dependencies 68
Handling dependencies 72
Exceptions are expected 80
Testing errors and exceptions 81
Summary 83
Chapter 5: Running Tests from the Command Line 85
Running tests 85
Processing test results 86
Test statuses 87
Command-line switches 90
Logging results 91
Code coverage 91
Including and excluding tests from the execution 91
When to stop the test execution 92
Configuration options 92
Code coverage analysis 93
Summary 96
Chapter 6: Test Isolation and Interaction 97
Test fixtures 97
Before and after each test method 98
Before and after each test suite class 99
Global state 100
Test dependencies 103
Data providers 104
Summary 106

Chapter 7: Organizing Tests 107
The PHPUnit XML configuration file 107
Test listeners 109
Configuring the code coverage 110
Where to store tests 110
Test suites 111
Groups 112
Using the bootstrap file 113
Summary 114
Chapter 8: Using Test Doubles 115
Creating test doubles 116
Test doubles in action 119
Using fake 122
Using stubs 123
Using mocks and expectations 126
Test proxies 131
Understanding Mockery 132
Installation methods 132
Comparing Mockery to PHPUnit 133
How to use Mockery 135
Summary 137
Chapter 9: Database Testing 139
Which database to use 140
Tests for a database 141
DBUnit 152
Installing DBUnit 152
Database test cases 152
Datasets 153
Using DBUnit 154
Doctrine 2 ORM and database testing 159
Summary 166
Chapter 10: Testing APIs 167
An example of integration testing 168
Testing the PayPal API 177
Testing the Facebook API 184
Testing the Twitter API 189
Testing the service-oriented architecture 193
Summary 194

Chapter 11: Testing Legacy Code 195
Testing spaghetti code 197
Black box testing 197
Using Reflection 198
Handling dependencies 201
The Patchwork library 201
The vfsStream library 204
The runkit PHP extension 207
Summary 213
Chapter 12: Functional Tests in the Web Browser Using Selenium 215
Installing Selenium 216
The Selenium IDE 216
The Selenium Server 219
Installing drivers 221
The PHPUnit Selenium extension 221
Testing in the browser 222
Recording Selenium tests 222
PHPUnit Selenium2TestCase 226
Writing Selenium tests 229
PHP-SeleniumClient 243
Organizing Selenium tests 245
Summary 246
Chapter 13: Continuous Integration 247
Using a Travis CI hosted service 248
Setting up Travis CI 249
Using Travis CI 251
Using the Jenkins CI server 253
Installation 253
Usage 254
Creating a job 255
Results 257
Using the Xinc PHP CI server 258
Installation 258
Usage 260
Summary 265

Chapter 14: PHPUnit Alternatives, Extensions, Relatives, and BDD **267**

Unit testing alternatives **268**

PHPUnit extensions **268**

Behavior-driven development **269**

Understanding BDD 269

Testing with PHPSpec 270

Installing PHPSpec 271

Using PHPSpec 272

Functional testing with Behat 277

Installing Behat 278

Using Behat 279

Summary **284**

Index **287**

Preface

In the last ten years, PHP as a programming language has made great progress. It has evolved from a simple scripting language to a powerful platform, powering some of the busiest websites on the Internet, such as Wikipedia and Facebook. It has grown from an ugly duckling to a beautiful swan, and some statistics say that 75 percent of the Internet these days is powered by PHP.

Many sites began as simple PHP scripts, which were successful and grew—more users and transactions. Yet a simple scripting language wasn't enough. For example, object-oriented programming was introduced in PHP 4, which was completely rewritten in PHP 5 with the usage of design patterns such as MVC (Model-View-Controller), event-driven programming, and much more. Things are not simple anymore; everything became more complex and also more difficult to understand. However, applications have become quicker to build by reusing already existing components without the need to reinvent the wheel again and again. Also, the developers are able to work with the code that they haven't seen before, and they are able to extend and modify this code.

What seems to be simple on paper is not simple in the real world. There exists complexity and dependency in the code, and this is exactly where the problem lies. The more complex the code, the bigger the chance it will break. This even has a name CRAP (Change Risk Analysis and Predictions) index. The CRAP index shows how difficult it will be to maintain and extend a certain code and how likely it is that new bugs can occur.

So what can you do to minimize the possible problems? The answer is unit testing. Simply, try to split your code into units (such as cells), which you can test independently, then test that every small bit for its desired actions. In the PHP world, it has become standard to have a PHPUnit, library, and framework written by Sebastian Bergmann for PHP code automated testing. The API is very similar to any other xUnit framework, such as the JUnit in Java world or NUnit in the .NET world.

What this book tries to show and teach you is how to write better and more predictable code, and PHPUnit and automated testing is great way to start.

But this book is not theoretical. The focus is on practical PHPUnit usage. While the PHPUnit is a great cornerstone, a cornerstone is not enough to build a house. This book helps you learn what a PHPUnit is and to write unit tests, but with real-world examples and not just isolated theoretical ones. Automated testing is really useful when it works, and this is something that is missing in many similar books—a style where the topic is in the real world and in the everyday developer's work context. Thus, the book covers all aspects, from the very beginning on how to install and use the PHPUnit, to how to run tests and, of course, how to write tests. But the question is why should we write tests?

As mentioned earlier, it's a way to write better code. It's a way to guarantee the quality of produced code, thus minimizing the number of bugs. Every developer wants to minimize these mistakes. But then question arises, who will pay for the tests? It's extra code that needs to be written, which will cost us extra time. The answer to this question is simple. Do you care about the work that you are doing, the code that you write, or don't you?

Writing tests and setting up automated testing should be a part of any bigger project and also part of the budget.

Quickly developing a project, sticking to a very tight budget and deadline, and sacrificing all procedures, including testing is a straight way to hell. When the project is delivered, more and more bugs will be found, and any refactoring or changes in application will be problematic and expensive. These extra costs will very quickly become much more than the initial investment in automated testing and writing testable code. But the worst case is that these costs could be completely unpredictable because even a simple change in an application might return as a boomerang again and again as more and more problems are reported by customers.

Simply writing code without testing is like borrowing money on a credit card—one day you have to pay it back, and you pay it back with interest. Not investing in development procedures and quality assurance is very shortsighted and sooner or later will lead to disaster. Writing PHPUnit tests and better code is the way to make your projects and business successful.

What this book covers

Chapter 1, *Installing PHPUnit*, introduces you to various PHPUnit installation methods. A simple thing such as PHPUnit installation is something where many users really struggle. It is easy, but it's worth considering different options and installation methods.

Chapter 2, *PHPUnit Support in IDEs*, shows how to configure and use four of the most popular PHP IDEs used for writing and running PHPUnit tests.

Chapter 3, *Tests and What They're All About*, gives a gentle introduction to unit testing and why and when to write tests.

Chapter 4, *Testing Dependencies and Exceptions*, will demonstrate how to write the code in a way that can be tested because one of the biggest nightmares in testing is to test dependencies.

Chapter 5, *Running Tests from the Command Line*, explains how to execute tests by using the command-line test runner. Running tests from the command line is a basic yet powerful way to run tests and get good test results.

Chapter 6, *Test Isolation and Interaction*, describes fixtures of the tests, which are steps to create a consistent known environment where tests will behave in an expected way or will highlight any lapses from expected behavior.

Chapter 7, *Organizing Tests*, will describe how to organize tests, where and how to store tests, and group tests to test suites.

Chapter 8, *Using Test Doubles*, explains how test doubles are a really useful way to replace dependencies. There are several techniques on how to achieve an expected behavior. You can easily replace a tested object or its part through doubles.

Chapter 9, *Database Testing*, explains techniques that allow to test code using a database.

Chapter 10, *Testing APIs*, describes that the best way to implement a third-party API into your own application is to first write integration tests to see whether the API works and how.

Chapter 11, *Testing Legacy Code*, will show several techniques on how to test a legacy code to be able maintain and extend it. But without tests, there is no refactoring.

Chapter 12, *Functional Tests in the Web Browser Using Selenium*, shows how to write PHPUnit Selenium functional tests. Over the years, Selenium has become standard in functional tests, running in a web browser by imitating user's behavior. Selenium tests are just extended PHPUnit tests and definitely are an important jigsaw in the testing puzzle.

Chapter 13, *Continuous Integration*, describes three popular open source projects that can host and run PHPUnit tests.

Chapter 14, *PHPUnit Alternatives, Extensions, Relatives, and BDD*, describes several interesting projects and alternatives of PHPUnit, their differences, and advantages in comparison to PHPUnit.

What you need for this book

PHP CLI (PHP Command Line Interface) is required, which can be installed as a package or which comes with packages such as XAMP or WAMP. An installed web server such as Apache or NGINX is also recommended. PHPUnit can be installed, and it works on Linux, Mac OS X, or Windows.

The first chapter covers various PHPUnit installation methods and the reader can choose, if he or she prefers, PEAR, Composer, GIT, or manual installation.

Who this book is for

PHPUnit Essentials is a book for PHP developers. The book helps with the first steps of the unit-testing world, and teaches readers how to install and use PHPUnit. It also gives developers, who have previous experience with PHPUnit, a detailed overview about testing in the context of the continuous integration process.

Testing is definitely a skill that every good developer should master, and the book helps you understand the point of testing by using simple everyday examples and by offering practical solutions to problems that developers face every day.

Conventions

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

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "CLI may use a different `php.ini` configuration file than the one your web server uses."

The book uses a straightforward approach. Usually the problem is described and then presented in a code similar to the following:

```
<?php

class FistTest extends PHPUnit_Framework_TestCase {

    public function testAddition(){
        $this->assertEquals(2, 1 + 1);
    }

    public function testSubtraction(){
```

```
            $this->assertTrue(1 == (2 - 1));
        }
```

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

```
> php -r "echo 'Hello, World!';"
```

Other code like Composer JSON configuration files are displayed in the same way as code:

```
{
    "require": {
        "phpunit/phpunit": "3.7.*"
    },
    "config": {
        "bin-dir": "/usr/local/bin/"
    }
}
```

New terms and **important** words are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "As a first step, create a new project as PHP application by navigating to the **File | New Project** menu."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

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

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

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

Customer support

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

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Downloading the color images of this book

We also provide you a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from `https://www.packtpub.com/sites/default/files/downloads/3439OS_ColoredImages.pdf`.

Errata

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

Piracy

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

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

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

Questions

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

1
Installing PHPUnit

To be able to write and run PHPUnit tests, PHPUnit has to be installed on the machine. With the right approach, it is not difficult; however, in the past, this was a bit of an issue for many users because they were struggling with the right settings and all of the dependencies that needed to be installed.

Until recently, the recommended installation method for PHPUnit was by using PEAR; however, from PHPUnit 3.7 onwards, there is an option available to use C**omposer**. Composer is a tool for dependency management in PHP, which makes PHP tools or libraries' installations much easier. There are a few more options to install PHPUnit, and this chapter describes them step by step.

Apart from what is required for PHPUnit installation, this chapter also describes the Xdebug installation process. This is optional, but this PHP extension really helps with processes such as remote debugging, code coverage analysis, and profiling PHP scripts. Later in the book, how to use Xdebug for debugging and generating code coverage will be explained, and it is more than an option for every PHP developer to have this extension.

This chapter describes how to install PHPUnit, and mentions what is required to be able to run PHPUnit tests; some of these tests are as follows:

- **PHP CLI**: This is a PHP command-line interface to run tests from the command line
- **PHPUnit**: This is a testing framework and tool for unit tests execution
- **Xdebug**: This is a PHP extension for remote debugging, code coverage, and much more

This chapter covers various PHPUnit installation methods, among which the easiest is the Composer or PEAR installation, but there are other installation methods available too. The following are the different PHPUnit installation methods:

- Composer installation
- PEAR installation
- Linux package installation
- Manual installation

As an extra step, it is recommended that you have Xdebug installed. Xdebug is a PHP extension that provides debugging and profiling capabilities and is also used by PHPUnit. Xdebug installation and configuration is described at the end of this chapter.

On 21st of April, 2014, it was announced end of life for the PEAR installation method. PEAR installation should be available during the entire 2014. However, recommended installation methods are the Composer installation or PHAR archive.

Requirements

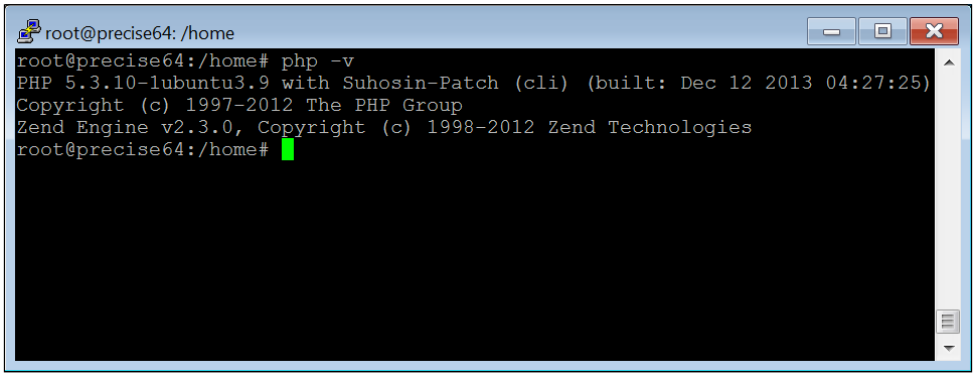
A developer who starts with PHPUnit is probably already working with PHP and has already installed the web server with PHP support such as Apache, NGINX, or IIS. It is strongly recommended that you use at least PHP 5.3.3 or higher; older PHP versions are not supported and maintained anymore.

Information about PHPUnit and downloads and documentation can be found on the official PHPUnit site at `http://phpunit.de`.

PHPUnit is a unit testing framework written in PHP, but PHPUnit tests are not usually executed on a web server but through the PHP CLI—the PHP command-line interface. Through PHP CLI, the PHP script can be run as any other shell script.

Running PHP from the command line

As mentioned earlier, PHP CLI allows to run any PHP script from the command-line interface. Whichever tool is used to run PHPUnit tests, even if it has used the IDE, which seems to have everything preinstalled, it will always call PHP CLI in the background.



You can install PHP CLI as a package (PHP5-CLI), or you can find it already installed with the web server package. Even software packages such as WAMP or XAMP have it there, but you might need to specify a path to web server binaries to environment variables to be able to run PHP from the command line anywhere on your machine.

To test if PHP CLI is installed on your machine, run the following command line:

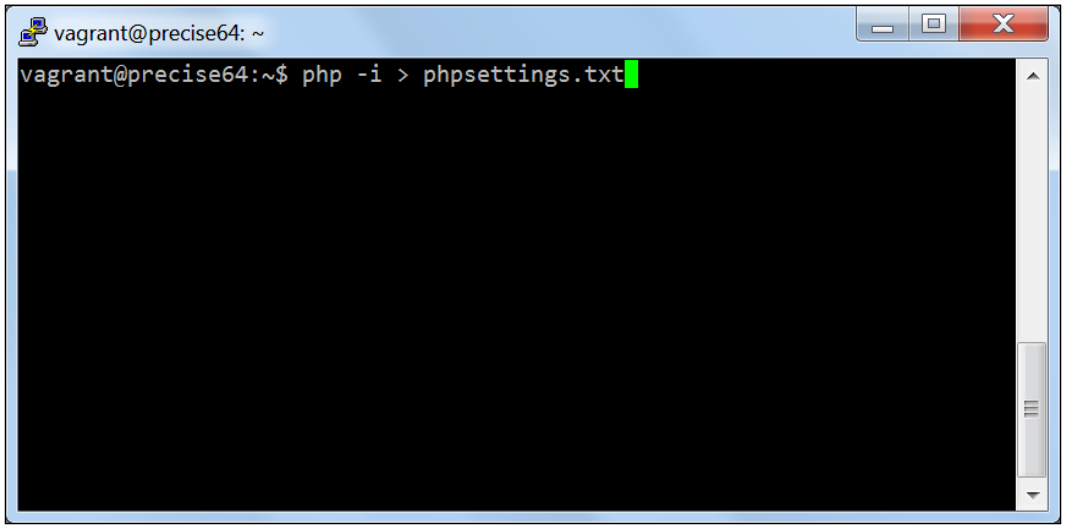
```
> php -r "echo 'Hello, World!';"
```

This is a simple `Hello, World!` code executed and outputted by the command language interpreter, but it confirms PHP CLI is present and does what is expected. If you see an error message such as command not found, it implies PHP CLI is not installed or the system path is missing—a path to the PHP binary.

CLI may use a different `php.ini` configuration file than the one your web server uses. If so, you'll need to configure your PHP extensions and other bits and bobs in there separately. To check this, you can run the following command line:

```
> php -i
```

It will output a result of the `phpinfo()` function, but in a text version, where you will be able check configuration settings. If you have problem reading so many lines, try to output it in a text file, which you can open in a text editor, and search for the desired result. The output also can be captured in the text file, which can be easily read with any text editor.



Composer – the dependency manager for PHP

Composer is a tool for dependency management in PHP. It allows you to declare the dependent libraries that the project needs, and it will install them. There is a configuration file called `composer.json`, which, as the name suggests, is a JSON format file. This is where you set the components/libraries and version your project will be using. When you run Composer, the Composer tool not only downloads all the required libraries, but also all the other required third-party libraries used. It also creates autoloader (the file to load all required classes), and all the hard work is done—everything that is required is ready to be used.

Composer is one of the best tools available for the PHP developers. It really helps and makes your life easier. It is something that was already available in other languages such as Maven for Java or Gems for Ruby and was missing in PHP. The best thing is, in this way, you can not only manage third-party libraries and packages, but also your own libraries.

Installing Composer

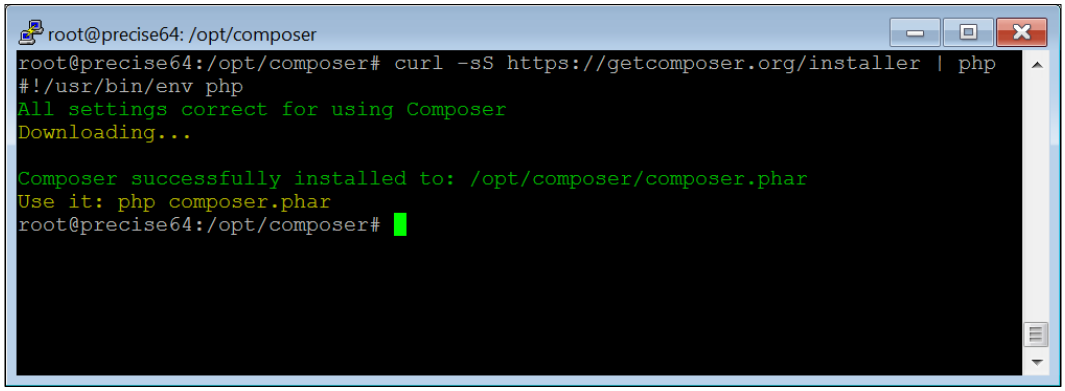
The Composer tool is just one file—`composer.phar`. The `.phar` file means a PHP archive (the PHAR file) similar to the JAR files of Java. All PHP files are packed in one archive file. You can download `composer.phar` from `http://getcomposer.org/download/` or by running the following command line:

```
curl -sS https://getcomposer.org/installer | php
```

If you haven't got curl installed, alternatively run the following command line:

```
>php -r "readfile('https://getcomposer.org/installer');" | php
```

As a result, you should see a message that `composer.phar` was downloaded, similar to the message displayed in the following screenshot:



Installation

There are two options that are of the biggest advantage to Composer, compared to other installation methods. Composer gives you the flexibility to specify where to store PHPUnit and also where to store the command-line script so that you can have on your machine as many versions as you want, and by using Composer, you can still maintain and keep them all up to date.

Local installation

Here is one practical example. You can specify and download PHPUnit only for your specific project. This means that when your project was developed and tested, you were working with PHPUnit version 3.5; however, for the new project, you want to use version 3.7. For example, Zend Framework 1 uses PHPUnit version 3.5; however, applications are still developed, extended, and maintained on this system even when Zend Framework 2 is available. This is not a problem as each project will have its own version of PHPUnit with all the dependencies, and there is no need to worry about clashing versions or being stacked with dated versions.

For PHPUnit installation with Composer, you have to create a `composer.json` file. The simplest version could look the following lines of code:

```
{
    "require-dev": {
        "phpunit/phpunit": "3.7.*"
    }
}
```

System-wide installation

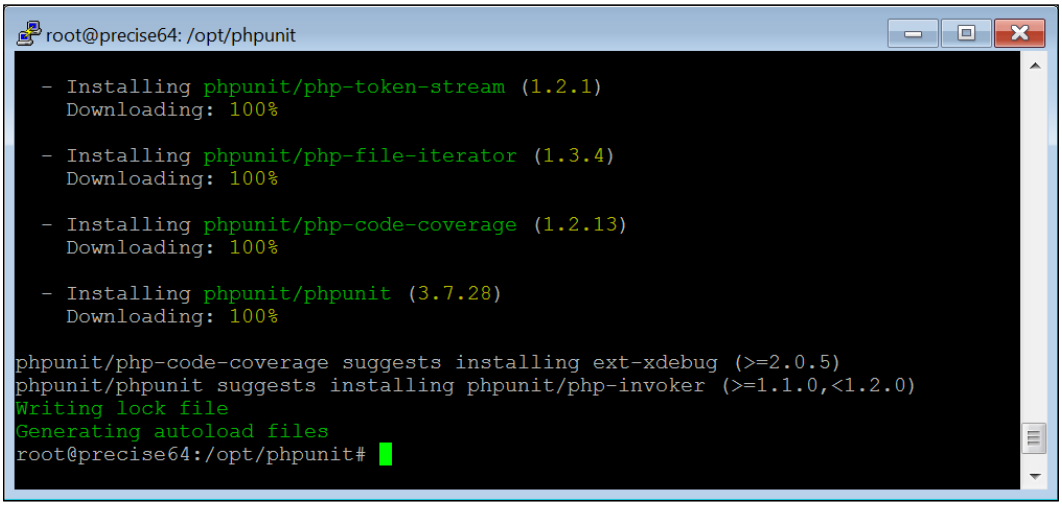
The second option is a system-wide installation. Well, basically it is the same thing, except it is created by a shell script / batch file to call PHPUnit from the command line. If a file is stored in a directory set in the system path, then it is available anywhere in your system. The following lines of code represent `composer.json`:

```
{
    "require": {
        "phpunit/phpunit": "3.7.*"
    },
    "config": {
        "bin-dir": "/usr/local/bin/"
    }
}
```

System-wide installation might be easier in the beginning, and allows you to run tests from the command line. It's easier to start with moving `composer.phar` to the project document root—the same place where the `composer.json` file is stored. Then, to install PHPUnit, just run the following command line:

```
> php composer.phar install
```

Composer explores the `composer.json` file and determines all the extra packages that need to be downloaded and installed. This should display a similar result as shown in the following screenshot:



This means PHPUnit was successfully installed, including the required dependencies (YAML) and basic plugins. The line that suggests installing `ext-xdebug` means that you should install the PHP Xdebug extension, because it is a useful thing for development and also PHPUnit uses it to generate code coverage.

Installing PEAR

Another way to install PHPUnit is to use PEAR—the framework and distribution system for reusable PHP components. PEAR has been available since the early PHP 4 days, and more than a framework, it is just a set of useful packages and classes. Also, it is a distribution system for installing and keeping these packages up to date. The first step of installation is to install PEAR. Download the PHP archive file `go-pear.phar` from `http://pear.php.net/go-pear.phar`.

Similar to `composer.phar`, it is a PHP archive file that contains all that you need. Installation is simple, just run the following command line:

```
> php go-pear.phar
```

Follow the PEAR install instructions, hitting *Enter* to accept the default configuration as you go. After installation, you can test it by running PEAR from the command line by just running the following command line:

```
> pear
```