Mastering Test-Driven Development with PHP 8

Building secure and reliable PHP 8 applications through TDD, design patterns, and functional data handling

Dr. Flávio Gomes da Silva Lisboa

www.bpbonline.com

First Edition 2025

ISBN: 978-93-65892-000

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true and correct to the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but the publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

My beloved wife, ***Maria****, and my dear daughter,* ***Koriander***

About the Author

Dr. Flávio Gomes da Silva Lisboa holds a PhD and a master's degree in social studies of science and technology from the Federal Technological University of Paraná. He holds a degree in computer science with a specialization in object-oriented programming and Java technology. Flávio is a certified software architect and engineer from Zend Technologies. He works as a technology analyst for the Brazilian Federal Data Processing Service (serpro), the largest public information technology company in Latin America, building and maintaining applications for cross-platform services.

At Serpro, Flávio coordinated a corporate program to promote free and open-source software, worked on the review of systems architecture for the Presidency of the Republic and the Ministry of Justice of Brazil, was head of the solution adaptation and mobility sector and architect of the ExpressoV3 project, and worked with support for the PHP programming language, behavior-oriented development, NoSQL databases, log analysis, and detection of bot attacks. He has experience in software development with emphasis for free and open-source software. FGSL is interested mainly in software architecture, area where he has experience in working with reuse, patterns, and frameworks.

He has been a speaker in PHP Conference Brazil for more than a decade and was honored by the event organization with a permanent room, alongside great members of the PHP Brazil community, as Elton Minetto, Guilherme Blanco, Ricardo Coelho, Rafael Dohms and Er Galvão. Flávio is also a professor at Centro Paula Souza, a network of colleges and technical schools, where he teaches systems analysis and design, web programming, and databases. Flávio is the author of the book Laminas Web Development, a complete guide about a PHP framework of the Linux Foundation. He also writes science fiction and in this segment, he has published the novels The One: the Solitude and the Harmony; Pandino, the Emperor: the Revenge of the One; and Freedom or Equality.

About the Reviewers

- **Mohamed A. Youssef** is a PHP developer with over 8 years of experience in back-end development, specializing in Symfony, Laravel, Magento, and OpenCart. He has worked on various web projects, focusing on server-side development, payment gateways, and XML integrations. He is also experienced in PHPUnit and **test-driven development** (**TDD**) to ensure code reliability and maintainability. Comfortable with HTML, CSS, JavaScript, and Linux environments, he enjoys building and optimizing web applications.

  Aside from coding, Mohamed has a keen interest in reading and reviewing books, especially in history, technology, and digital trends. With a bachelor's degree in history from Cairo University, he appreciates both technical and historical perspectives in his reviews.

- **Sushil Kumar Gupta** is a seasoned senior software engineer with over 9 years of experience in developing scalable web applications using PHP, CodeIgniter, Laravel, Magento, and Shopify. He specializes in software redevelopment, API integrations, payment gateways, and CRM development, ensuring optimized performance and seamless user experiences.

  Currently, he is a senior PHP developer at a fintech company, leading the development of a business loan system from scratch, focusing on security, scalability, and third-party financial integrations. Previously, at Addison IT Solutions, he played a key role in redeveloping healthcare software and led the homecare project, improving efficiency and system performance.

  Sushil is also experienced in mentoring teams, conducting code reviews, and establishing best coding practices. His technical expertise extends to server management (Linux), MySQL, PostgreSQL, and API integrations. He holds an MBA in IT and a BCA, reflecting his strong academic foundation.

  Passionate about technology and innovation, he enjoys tackling complex software challenges and continuously improving system performance. His contributions to the software development industry make him a valuable asset in any project requiring technical excellence and leadership.

Acknowledgement

I would like to express my sincere gratitude to all those who contributed to the completion of this book.

First and foremost, I extend my heartfelt appreciation to my fellow teachers and members of the PHP community in Brazil. Their recognition they have of my work have been a constant source of motivation.

I would like to extend my special thanks to the following individuals for their valuable input and contributions to this project: Anderson Fernandes Burnes, Daiana Alves, Eduardo Herbert Ribeiro Bona, Er Galvão Abbott, Felipe Augusto Rezende, Fernando Silva, Flávio Augusto da Silveira, Matheus Gontijo, Vítor Mattos and Wellton Costa de Oliveira. Your articles, classes and lectures contributed to many ideas I had and our meetings in events gave me valuable experiences to think about regarding a solid PHP education.

I am immensely grateful to BPB Publications for their guidance and expertise in bringing this book to fruition. Their support and assistance were invaluable in navigating the complexities of the publishing process.

I would also like to acknowledge the reviewers, technical experts, and editors who provided valuable feedback and contributed to the refinement of this manuscript. Their insights and suggestions have significantly enhanced the quality of the book.

Last but not least, I want to express my gratitude to the readers who have shown interest in my book. Your support and encouragement have been deeply appreciated.

Thank you to everyone who has played a part in making this book a reality.

Preface

Understanding the test-driven development approach is important for building quality applications. This book *Mastering Test-Driven Development with PHP 8* covers in a practical way the fundamentals necessary for a programmer to develop testable and secure PHP web applications.

Comprising 12 insightful chapters, this book covers a wide range of topics essential for creating web applications with the PHP programming language covered with testing and security mechanisms. We start with PHP fundamentals and test-driven development, providing a solid grounding in the basics. From there, we move towards object-oriented programming, exploring the implementation of this paradigm in PHP to read and write data stored in file system, relational and non-relational databases.

The book targets professionals with some experience in programming who intend or was requested to use PHP for backend. The book assumes readers know main web standards for web frontend, HTML and CSS. The book believes that the readers may have intermediate level knowledge of algorithms and data structures, requirements analysis and UML.

Through practical examples and an evolutionary approach, with heavy use of refactoring, this book takes the reader from the beginning through test-driven development, while showing the power of the PHP programming language to create easy-to-maintain web applications.

Chapter 1: Meeting and Installing PHP - This chapter of the book will cover the definition of PHP, how it works, and how we can create an environment to work with that programming language. We will start with an overview of the creation and evolution of PHP, then we will explain the structure of its community and how it is organized. We will also explain how to install PHP on a real machine with multiple options, from the simplest to the hardest way. Afterward, we will learn how to create a virtualized development environment for PHP. Finally, we will show how to prepare the environment for working with a test-driven development approach, which we will use in the next chapters. At the end of this chapter, you will be able to understand what is PHP and how to install it for developing backend applications.

Chapter 2: PHP Foundations - This chapter will cover the foundations of the PHP programming language. We will start by understanding PHP syntax and structural elements in comparison with C language. Then, we will learn the control-flow structures of

PHP in standard and alternative forms. We will also learn about data types, type-juggling, and operator precedence. Next, we will learn how to define functions and use built-in functions in PHP, along with PHP functions to manipulate cookies, sessions, HTML forms, and HTTP responses. By the end of this chapter, you will be able to understand the foundations of the PHP programming language. You will be able to recognize PHP flow control structures in their two forms.

Chapter 3: Function Driven Registration with File System Storage - This chapter of the book will cover the project and implementation of a book registration using function-oriented programming. We will start with the project presentation, which will serve as an exercise to learn PHP incrementally by refactoring an application. In this first version, the reader will learn how to code the registration with modularized code in functions and data storage in a file system. Every backend code produced from now on will be covered with tests. By the end of this chapter, you will be able to build a function driven PHP application. You will be able to use PHP to write and read data from text files in more than one format (plain text, CSV and JSON).

Chapter 4: Function Driven Registration with Relational Database Storage - This chapter of the book will cover refactoring of the book's registration application, replacing the file system storage with a relational database storage, using MySQL/MariaDB. This will be the first exercise of wide refactoring and the first step to think about the concept of decoupling. We will add the ability to read and write to database tables, making system storage configurable to work with both paradigms. At the end of this chapter, you will be able to build a function driven PHP application storing data in a relational database. You will be able to use PHP to create, recover, update and delete records from database tables.

Chapter 5: Function Driven Registration with Document Database Storage -This chapter of the book will cover a refactoring of the book registration application, replacing the relational database storage, using MySQL/MariaDB, with a document database storage based on MongoDB. We will add the ability to read and write to collections, widening the storage options of the application. By the end of this chapter, you will be able to build a function driven PHP application storing data in a document database. You will be able to use PHP to create, recover, update and delete records from collections in a MongoDB database.

Chapter 6: PHP OOP- This chapter will cover how to program object-oriented registration in PHP. We will start by learning how OOP techniques are implemented in the PHP programming language and how we replace uncoupled variables and functions with attributes (or properties) and methods encapsulated in classes. We will learn the foundations of object creation, cloning and comparison. We also will learn how to use

the main reuse mechanism of object-oriented programming, class inheritance, and how to establish communication patterns using interfaces. Next, we will learn about the magic methods, the methods which are invoked in response to events. Finally, we will learn how to manipulate different relational databases with a standardized interface and how to handle exceptions and errors.

Chapter 7: Object-oriented Registration with File System Storage -This chapter of the book will cover how to refactor a function driven registration with file system storage for an object-oriented implementation. We will learn how to create classes and their methods from functions initially by converting the author recording in a plain text file to object-oriented mode. After this experience, we will refactor the recording in other file system formats. We will continue converting read, update and remove operations from functions to class methods. By the end of this chapter, you will be able to convert functions into methods and group them into classes. You will learn how to think about call semantics by having an actor performing an action rather than an action occurring without someone responsible for it.

Chapter 8: Object-oriented Registration with Relational Database Storage- This chapter will cover how to refactor a function driven registration with relational database storage for an object-oriented implementation. In this chapter, we will replace the use of the native MySQL driver with the **PHP Data Objects** (**PDO**) driver. We will learn what **object-relational mapping** (**ORM**) is, the architectural patterns of this approach and how to implement the two of them and using them to refactor the Librarian application. By the end of this chapter, you will be able to understand how to replace the use of native MySQL driver with the PDO driver. You will be able to understand ORM and ORM architectural patterns and how to use them for refactoring an application using a relational database.

Chapter 9: Object-oriented Registration with Document Database Storage- This chapter of the book will cover how to refactor a function-oriented database implementation from a document-oriented database to an object-oriented implementation using the Row Data Gateway design pattern. We will identify duplications and eliminate them, generalizing with abstract classes and enforcing patterns with interfaces. We will review the models we created, the user interface, and the request handling. By the end of this chapter, readers will be able to understand how to use Row Data Gateway pattern for document-oriented databases. You will be able to perform refactoring to eliminate duplication using abstract classes and enforce communication patterns using interfaces.

Chapter 10: Abstracting the Application Storage- This chapter of the book will cover the refactoring of the previous version of the book registration application and the creation of a new layer for decoupling the model layer from the storage paradigm. This change will

be justified as a strategy to avoid a vendor lock-in and reducing the cost of a migration. This chapter will start covering what are design patterns and how to use them. Next, we will learn how to create a REST API for our application. After having implemented an API, we will learn how to generate a code coverage report. Then, we will follow with changes in the handling of the web pages, introducing an architecture pattern to separate the layout from the content. Finally, we will refactor the application with the Model-View-Controller pattern.

Chapter 11: Refactoring the Application with Secure Development- This chapter of the book will cover how to refactor PHP application by applying techniques to let it secure to reduce an attacker's chance of success. We will start talking about the relationship between secure development and secure code. We will learn about information resources about web application vulnerabilities and PHP vulnerabilities. After understanding these fundamentals, we will review the application's inputs, introducing filters and data validators. Next, we will review the data outputs, analyzing whether the application does not expose data that it should not. Finally, we will talk about the automated code analysis as a tool to help discover vulnerabilities.

Chapter 12: Authentication and Authorization- This chapter of the book will cover two topics of secure development. We will learn how to authenticate users using session-based and token-based authentication, as specified by the **JSON Web Token** (**JWT**) standard. We will also learn how to authorize users to perform specific operations within an application, using two different approaches, **access control list** (**ACL**) and **role-based authorization control** (**RBAC**). In the latest version of the Librarian application, users will only be able to manipulate authors and books if they are authenticated and have permission to perform the operations. By the end of this chapter, you will be able to implement authentication and authorization in PHP applications. You will be able to authenticate users and persist the authentication data in the session or use a token in the JWT standard and use the application control layer to ensure that all non-public pages are only accessed by authenticated users.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/to9bb42

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Mastering-Test-Driven-Development-with-PHP-8.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Meeting and Installing PHP

Introduction

This chapter of the book will cover the definition of **Personal Home Page** (**PHP**), how it works, and how we can create an environment to work with that programming language. We will start with an overview of the creation and evolution of PHP, then we will explain the structure of its community and how it is organized. We will also explain how to install PHP on a real machine with multiple options, from the simplest to the hardest way. Afterwards, we will learn how to create a virtualized development environment for PHP. Finally, we will show how to prepare the environment for working with a test-driven development approach.

Structure

The chapter covers the following topics:

- PHP community and ecosystem
- Preparing PHP development environment
- Ways to install PHP
- Virtualized environment for PHP
- Test-driven environment for PHP

Objectives

By the end of this chapter, readers will be able to understand what is PHP and how to install it for developing backend applications. You will be able to quickly install a minimal development environment for PHP using the XAMPP package, but you will also be able to install it in other ways, both on real and virtual machines. Additionally, you will be able to prepare your environment to work with test-driven development.

PHP community and ecosystem

In this section, we will answer three fundamental questions about PHP, what it is, how it works, and how it is maintained. It will provide you with the most relevant knowledge to help you understand this concept.

Creation and ascension of PHP

In 1995, a Danish programmer named *Rasmus Lerdorf* published the source code of an interpreter he had created under the **General Public License** (**GPL**), a free software license. In doing so, he allowed anyone to not only run the interpreter, but study, modify, and redistribute it, including modifications. The interpreter that *Rasmus* created was called **Personal Home Page tools**. This name reveals what *Rasmus* purpose was when he created the interpreter. He wanted a tool that would help him to create his web page.

Rasmus often says in his talks that the philosophy behind PHP is a programmatic approach to the web problem. This information is important because it shows that PHP knows what to focus on. It was not a language designed to solve any type of problem. Although today, it is possible to create PHP desktop applications or scripts for infrastructure automation, *Rasmus* designed it specifically to take advantage of the HTTP protocol. In 1995, to manipulate HTML form with two fields, *Rasmus* needed to write 63 lines in C language. The most concise alternative for him was the Perl language, with which he could write the same thing using 17 lines and the code was more readable for humans but *Rasmus* wanted an approach centered around HTML, which was a language designed for HTTP, a hypertext language for a hypertext protocol. So, he created a programming language that allowed him to write instructions to generate dynamic content within HTML tags.

By opening the source code of his interpreter, *Rasmus* got help making it more powerful. In fact, what he had produced in order to solve a personal problem attracted the interest of many people who saw enormous potential in the project for building dynamic websites. The first version of PHP had 14 source code files, collected in a single folder. Two years later, in 1997, the second version of PHP already had 68 files and its first subfolder. This second version was called **PHP/Forms Interpreter** (**PHP/FI**). In 1997, the PHP documentation group was created, which grew in the following decades and built the documentation available at **https://www.php.net/docs.php**

In 1998, with the help of *Zeev Suraski* and *Andi Gutmans*, *Rasmus* released a third version of PHP with the support of object-oriented programming. Since then, the use of PHP increased rapidly, so that by 2004 more than 18 million domains were using it as their server-side programming language. Starting with the third version, PHP began to be distributed under the PHP license, but it remains free and open-source.

When *Zeev* and *Andi* began collaborating with *Rasmus*, they created a company called *Zend Technologies*, where Zend is the combination of the first two letters of their names. In PHP version four, the duo released the Zend engine, a compiler and runtime environment for PHP. With this engine, PHP gained greater performance because the scripts began to be compiled into blocks of machine language instructions called **opcodes**. PHP was not a Personal Home Page anymore, but a recursive acronym of PHP Hypertext Preprocessor.

Zend engine's architecture added compilation power while maintaining the ease of use that *Rasmus* wanted for PHP. Let us compare PHP architecture with Java architecture to understand it better. For printing the sentence *Hello, World!* using Java, you need to create a file with five lines and execute two commands, **`javac`** for compiling a **`.java`** file, generating a **`.class`** file with the bytecode, and **`java`** for running the **`.class`** file. For doing the same thing with PHP, you can create a **`.php`** file with one single line and you need to execute one only command, **`php`**. Zend engine compiles **`.php`** file generating opcode in memory for running that opcode.

The comparison between the two processes is shown in *Figure 1.1*:

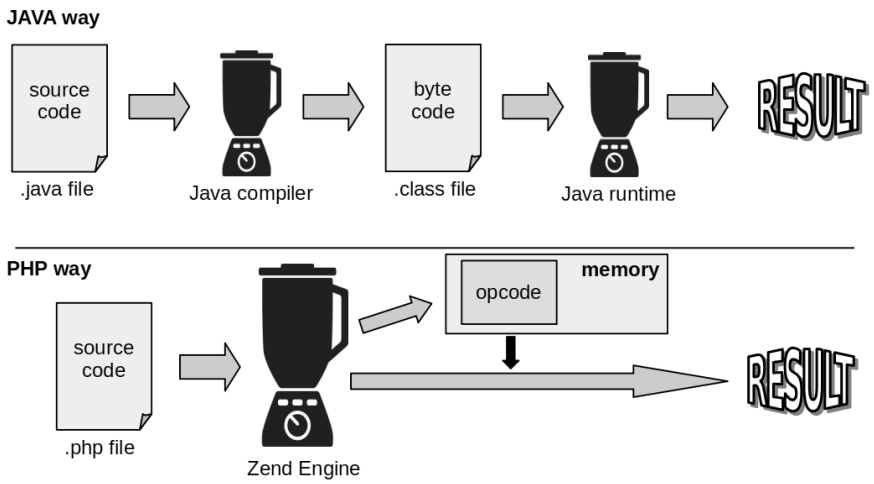

***Figure 1.1**: Comparison between Java and PHP environments*

Over two decades, PHP has fully implemented support for object-oriented programming and complementary concepts like traits, always keeping the simplicity. In addition, it expanded its library of functions and integrations with other systems. PHP version eight is a much more complex system than *Rasmus Lerdorf's* small personal project and it also requires many people to maintain it.

Let us look at the community organization and the resources available for developers.

PHP ecosystem

PHP is written entirely in C language and it is currently maintained in a repository on GitHub for almost one thousand developers. If you are curious to know how the old versions of PHP were implemented, you can find them at the PHP Museum at **https://museum.php.net**. In addition to the development team, there is a team of documenters and translators. Guidelines for contributing to the documentation or translating it are available at **http://doc.php.net/tutorial**

PHP developers are currently working on evolving the language using **Requests for Comments** (**RFCs**). Each RFC contains a change that is discussed and then voted on by the group. If approved, it is implemented in the language. You can find out about how to create an RFC and which ones are being voted on, accepted, and implemented on the page **https://wiki.php.net/rfc**. The complete guidelines for contributing with PHP, including the explanation about the source code directory structure, are available at **https://github.com/php/php-src/blob/master/CONTRIBUTING.md**

Although it is not our goal here to learn how to modify PHP but rather how to use it to create applications using it, you may be interested in learning how to make changes to PHP. The first step to this is to find an issue at **https://github.com/php/php-src/issues** and resolve it. Remember you will have to program in C to do that.

PHP emerged and evolved through the collective contributions of numerous dedicated individuals. PHP developers keep the project active because they like it, not because they are paid to do so. They generally contribute to PHP outside of their working hours. To help maintain PHP, PHP Foundation was created, it brings together people and organizations that help support PHP. PHP Foundation receives monetary donations to maintain the PHP development infrastructure and the programming language itself. PHP Foundation is completely transparent and you can learn about its work at **https://thephp.foundation**

In addition to the community of PHP language developers, there is a much larger community of developers who use the PHP language to create web applications, desktop applications, and infrastructure automation scripts. This community has developed many widely used applications, such as database clients, content management systems, course management systems, and e-commerce platforms. In the next section, we will learn about several tools created to help PHP developers.

Once we have learned what PHP is, how it works, and how it is maintained, let us learn how to prepare a development environment for PHP.

Preparing PHP development environment

In this section, we will learn how to install a PHP development environment with the minimum requirements to create web applications that are capable of storing data in a relational database.

In 2004, almost half of Apache servers used the PHP integration module. The free and open web server Apache has become one of PHP's great partners over the years. Another great PHP partner is the MySQL database. The partnership with MySQL even generated a PHP client for the database, phpMyAdmin. This trio has become very popular under GNU/Linux operating system distributions. Thus, it was possible to develop and produce web applications using a stack of free and open-source code technologies. This stack made up of **Linux, Apache, MySQL, and PHP** (**LAMP**).

Note: GNU is a recursive acronym for GNU is not Unix. Linux is the kernel of the operating system GNU.

Although the LAMP stack worked, it was not so trivial to install each of the software and integrate them. In 2002, recognizing this difficulty, two young Germans, *Kai Oswald Seidler* and *Kay Vogelgesang*, founded the Apache Friends project. This aimed to facilitate the installation of this technology stack. In fact, Apache Friends is not restricted to the Linux operating system. The project has Apache, MySQL, and PHP installation packages for Windows and MacOS as well. This set of packages is called **XAMPP**, where *X* can be any of the three supported operating systems.

Let us learn how to install XAMPP.

Installing XAMPP

You can get XAMPP from **https://www.apachefriends.org/download.html**. This page shows XAMPP installers grouped by operating system. There are generally three installers, for the three most recent stable versions of PHP. This is important information, XAMPP provides installers with stable versions. In the next section, we will see how to install PHP in the latest and unstable version, but for now, we will learn how to create a stable development environment. On the same XAMPP download page, there are links with instructions for installation on each operating system. The process is similar for all three operating systems, you download the installer, run it, and then start the Apache server and MySQL database.

Let us now look into the steps for installing XAMPP on a GNU/Linux distribution to demonstrate the installation process, as follows:

Note: Even if you do not use Linux, know that these procedures will be replicated in the section on creating the virtual development environment.

1. Download the XAMPP installer for PHP 8.2 from the Linux section of the page at **https://www.apachefriends.org/download.html,** as shown in *Figure 1.2*: