Practical Guide in Software Engineering

Gab Amba

Practical Guide in Software Engineering

Gab Amba

This book is for sale at http://leanpub.com/practical-guide-in-software-engineering

This version was published on 2020-04-25

Contents

About this book **1**

Assumptions **2**

Getting Started **3**
Version Control using GIT 4

About this book

This is not your typical PHP programming book.

This book will not teach you how to build websites or web-based applications but serves as introductory guide about PHP (using version 7.1) as a tool for solving problems like Project Euler and focuses on the software engineering principles and practices.

Currently, there are 500+ problems to solve in the Project Euler site and we are not solving all of those problems but weâ€™ll pick some problems as requirements in evolving the product/library called ProjectEuler.

The software engineering practices and principles that you will learn in this book is not only applicable in PHP but in other programming languages as well.

Assumptions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/practical-guide-in-software-engineering.

Getting Started

This section guides you on how to create a new PHP project and setting a version control using GIT.
Setting a new PHP project
First is to create the project folder called ProjectEuler and create the initial files.

```
$ mkdir -p ProjectEuler
$ cd ProjectEuler
$ touch composer.json
$ touch phpcs.xml
$ touch phpunit.xml
```

So, let's update our composer.json file and add the following json script:

```
{
    "name": "project-euler",
    "type": "library",
    "require": {
        "php": "^7.1.0"
    },
    "autoload": {
        "psr-4": {
            "ProjectEuler\\": "src/"
        }
    },
    "autoload-dev": {
        "psr-4": {
            "ProjectEulerTest\\": "test/"
        }
    }
}
```

Now, let's update our phpcs.xml file and add the following xml script:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset name="Zend Framework coding standard">
    <rule ref="./vendor/zendframework/zend-coding-standard/ruleset.xml"/>
    <!-- Paths to check -->
    <file>src</file>
    <file>test</file>
</ruleset>
```

Next is to update our phpunit.xml file and add the following xml script:

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap="./vendor/autoload.php" colors="true">
    <logging>
        <log type="coverage-html"  target="build/coverage" title="ProjectEuler"
        charset="UTF-8" yui="true" highlight="true"
        lowUpperBound="35" highLowerBound="true" />
        <log type="coverage-clover" target="build/logs/clover.xml" />
        <log type="junit" target="build/logs/junit.xml" logIncompleteSkipped="false"\
/>
        <log type="testdox-html" target="build/phpdox/index.html"/>
        <log type="testdox-txt" target="build/phpdox/testdox.txt"/>
    </logging>
    <testsuite name="regression">
        <directory suffix="Test.php">./test/</directory>
    </testsuite>
    <filter>
        <whitelist processUncoveredFilesFromWhitelist="true">
            <directory suffix=".php">./src/</directory>
        </whitelist>
    </filter>
</phpunit>
```

Version Control using GIT

Version control systems like GIT will help us manage and track the changes made, especially source code, in our project.

In order to do that, we need to initialize GIT in our project directory.

```
$ git init
```

With that, GIT will create a .git subfolder in our directory which holds all of the change logs in our project.

Initial commit
After initializing the version control in our project, we need to add and commit the changes that we've made earlier.
Checking file status
First is to check the status of the files by the following command:

```
$ git status -s
```

The output will be:

?? composer.json
?? phpunit.xml
?? phpcs.xml

NOTE: ?? indicates that the files are not yet added in the version control.

Committing changes
So let's add them by the following commands:

```
$ git add -A
$ git commit -m "initial project files"
```

==TO DO: Explain the commands ===

IMPORTANT NOTE: Be careful on adding sensitive information or files. See documentation about GIT.

After adding the initial files in the version control, it will automatically create a branch called master.

Master, as initial, branch will serve as the container of your projects document and will be used as a point of reference, the source of truth, for adding/updating/deleting commits within the project's lifecycle.

You'll learn more about the development workflow on the next chapters of this book.