Alfonso De La Guardia

Composer

Overview

Composer is a tool that manages the dependencies of our PHP project.

Thanks to this tool, we can work with the installation of those libraries that our project needs in a comfortable way, avoiding all the problems involved in installing the dependencies manually and without any control or supervision.

The dependencies are installed in a directory (default called / vendor). What if my bookstores depend on others? It is also able to solve that and download everything necessary to make it work and thus avoid doing it manually.

Table of Contents

Project Requirements

- You must perform all the steps using only the command line.
- You must configure your repository to ignore the following files and directories
- Directory where composer dependencies are installed
- You should be able to run the Guzzle library and make a small example using the methodology provided by composer.
- You must be clear about the use of composer and its tool by command line.
- You must be clear about the difference between a development unit and a production unit
- Create a clear and orderly directory structure
- Both the code and the comments must be written in English
- Use the camelCase code style to define variables and functions
- In the case of using HTML, never use online styles
- In the case of using different programming languages always define the implementation in separate terms
- Remember that it is important to divide the tasks into several sub-tasks so that in this way you can associate each particular step of the construction with a specific commit
- You should try as much as possible that the commits and the planned tasks are the same
- Delete files that are not used or are not necessary to evaluate the project

Risk Management Plan

Every project has risks. These risks must be taken into account to improve the workflow of the project. I've listed the risks for the project, along with the impact they might have, and the priority of them.

iD	Risk	Consequence	Prob. (1-5)	imp. (1-5)	Pri. (1-25)	Mitigation approach
1	Breaking my computer	Can't do anything	1	5	5	Keep my repo work to date, look for other computers

2	Getting sick	Wouldn't be as productive	2	3	6	Eat and sleep well
3	Not concentrating enough	Won't com	2	5	10	Focus on the Minimum Viable Product.
4	Unrealistic deadlines	Deadlines wouldn't be met + development shortcuts would have been taken affecting the robustness of the code	2	3	6	Be more organized with the tasks, set new deadlines.
5	Being unfocused	Loss of control over the development flow of the project	4	5	20	Focus on finishing the most important tasks only.

Project Tasks

Defining this part is crucial to the development of the project. It is important to make a good analysis of the situation to organize the project in a good way:

#	Task	Priority (1-5)	Description	Difficulty (1-5)	Estimation
1	Reading the description	4	Reading the description of the project to understand its scope	1	30 min
2	Create Repo	3	Creating git repo for the project	1	2 min
3	Install Composer	5	Installing composer to be used for the development of the project	3	2 hr

4	Install the necessary dependencies	5	Installing all the dependencies necessary for the project	2	1 hr
5	Run your script importing dependencies	4	Running the script for importing all the necessary dependencies	4	2 hr
6	Researching other functionalities	3	Researching for the functionalities mentioned in the project description	3	1 hr
7	Review	2	Review Project	2	30 min

Chronogram

Task #	Monday	Tuesday	Wednesday
1	X		
2	X		
3	X		
4		X	
5		X	
6		X	
7			X

Other functionalities

- Find out how you can update a dependency on your project:

  To update, you can use the following command: composer update.

- Research how you can eliminate a dependency from your project:

  To delete a dependency from your project you can use: php composer remove vendor/package

- Investigate what the composer.json file is and what it is for:

  The composer.json file is the configuration of your project. This is usually located at the top level of your project. Ideally this is a directory outside the scope of your web server.

Git Workflow

For this project, all commits we'll be pushed directly to the Master branch. All commits will use a descriptive message, so that myself or other users can easily go to the Git version that they need to. This is very important for working in teams as it increases communication and efficiency between all members.

Technologies used

For this project, we will use the following technologies:

- Composer
- Visual Studio Code
- Guzzle

Lessons

- Composer is an application-level package manager for the PHP programming language that provides a standard format for managing dependencies of PHP software and required libraries.
- Composer runs from the command line and installs dependencies (e.g. libraries) for an application.
- It also allows users to install PHP applications that are available on "Packagist" which is its main repository containing available packages.
- It also provides autoload capabilities for libraries that specify autoload information to ease usage of third-party code.

Incidents

- At first, difficulty understanding how to use Guzzle in the project.
- Also some difficulties finding a good API to test Guzzle, because some APIs had a limited amount of logins possible.