Formation PHP - Symfony

D03 - Composer

Timea LASZLO tlaszlo@pitechplus.com
Staff 42 bocal@staff.42.fr

Summary: Following 42 formation course, you will learn what Composer is and how you can use it in your applications.

Contents

Chapter I

Foreword

While writing applications in PHP from scratch, you will probably find that it feels like you have to keep re-inventing the wheel anytime you want to do a common task such as User Authentication, Database Management, etc. Here comes the Composer which is a dependency manager for PHP that will pull in all the required libraries, dependencies and manage them all in one place. This kind of management for dependencies in a project is not a new concept, and in fact, much of Composer is actually inspired from Node.js's NPM and Ruby's Bundler. In comparison with other former known package manager like PEAR, the Composer will allow you to install packages on a project-by-project basis rather than system wide.

Chapter II

General Rules

- This subject is the one and only trustable source. Don't trust any rumor.
- This subject can be updated up to one hour before the turn-in deadline.
- The assignments in a subject must be done in the given order. Later assignments won't be rated unless all the previous ones are perfectly executed.
- Be careful about the access rights of your files and folders.
- You must follow the `turn-in process` for each assignment. The url of your `GIT` repository for this day is available on your intranet.
- Your assignments will be evaluated by your Piscine peers.
- In addition to your peers evaluation, a program called the "Moulinette" should also evaluate your assignments. Fully automated, The Moulinette is tough and unforgiving in its evaluations. As a consequence, it is impossible to bargain your grade with it. Uphold the highest level of rigor to avoid unpleasant surprises.
- All shell assignments must run using `/bin/sh`.
- You must not leave in your turn-in repository any file other than the ones explicitly requested By the assignments.
- You have a question? Ask your left neighbor. Otherwise, try your luck with your right neighbor.
- Every technical answer you might need is available in the `mans` or on the Internet.
- Remember to use the Piscine forum of your intranet and also Slack!
- You must read the examples thoroughly. They can reveal requirements that are not obvious in the assignment's description.
- By Thor, by Odin! Use your brain!!!

Chapter III

Exercise 00

	Exercise 00
Exercise 00: Install Composer globally	
Turn-in directory : *ex*00/	
Files to turn in : `Files and folder from your application`	
Allowed functions : `All methods are allowed`	
Notes : `n/a`	

This exercise is mandatory. To resolve all the others, you need to complete this one. The requirement for this exercise is to install ***Composer*** globally, from your terminal.

Chapter IV

Exercise 01

	Exercise 01
Exercise 01: Specifying versions	
Turn-in directory : *ex*01/	
Files to turn in : `Files and folder from your application`	
Allowed functions : `All methods are allowed`	
Notes : `n/a`	

This exercise will have many subtasks. For each one of them you need to create a new composer file.
For this exercise you need to install many versions of **Monolog** package, using **Composer**.
The requirements are:

1. version 1.20.0
2. version greater than 1.19.0 and less or equal with 1.21.0
3. version between 1.19 and 1.20
4. version greater or equal with 1.19 and less than 2.0
5. version greater than 1.19 and less than 2.0

Hint: *To resolve these exercises you need to use **composer install** command.*

Chapter V

Exercise 02

	Exercise 02
Exercise 02: Development requirement	
Turn-in directory : *ex*02/	
Files to turn in : `Files and folder from your application`	
Allowed functions : `All methods are allowed`	
Notes : `n/a`	

Install the version 4.8.27 of PHPUnit package as a development requirement.

Chapter VI

Exercise 03

	Exercise 03
Exercise 03: Composer install vs composer update	
Turn-in directory : *ex*03/	
Files to turn in : `composer.json, composer.lock, files and folder from your application`	
Allowed functions : `All methods are allowed`	
Notes : `n/a`	

This exercise will have two subtasks. For each of them you need to create a subdirectory with the composer files.
Having the supplied **composer.json** and **composer.lock** files in your current directory, you need to run **composer install** and **composer update** commands. Observe which files are created/updated and what is the difference between *install* and *update*.
Steps to follow:

1. Copy the ***composer.json*** and ***composer.lock*** files in your current directory

2. Run **composer install** command

3. Run **composer update** command

Chapter VII

Exercise 04

	Exercise 04
Exercise 04: Add package via Composer	
Turn-in directory : *ex*04/	
Files to turn in : `Files and folder from your application`	
Allowed functions : `All methods are allowed`	
Notes : `n/a`	

Create a custom package, that will be included in projects via Composer. The package needs to implement a function that will return the current weather from your current city. The weather condition needs to be saved in a **weather.txt** file and it needs to be displayed in degrees celsius.
To create your package, you need to use a RESTClient to call the API. You need to find an API that will return the current weather, in degrees, for your current location. Maybe, to request the server for the weather you need an API key/token that will be obtained only after registration. Please do not ignore this step.

The package needs to respect the following requirements:

- the name of the package should have the format: **yourName/weather**
- it needs to have required at least 5.5.0 **PHP** version
- it needs to have required the **RESTClient** package for PHP (https://github.com/tcdent/php-restclient)
- the stability needs to be set to **dev**