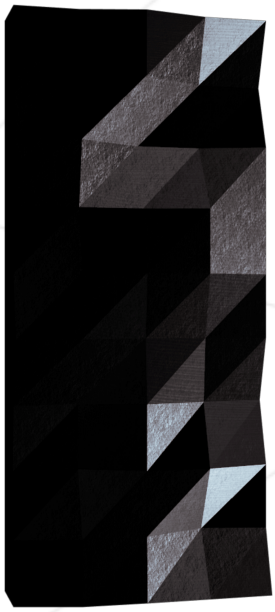

Formation PHP - Symfony

Rush00

Rares Serban rares.serban@pitechplus.com
42 pedago pedago@42.fr

Summary: You will create a Symfony application utilizing the OMDB api.

Contents

Chapter I

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 II

Day-specific rules

- You **MUST** provide an author file named `auteur` at the root folder of your repository, as always.

```
$> cat -e auteur
noich$
thor$
$>
```

- This project will be entirely graded by humans. As such, you are free to have some variation between this subject's examples and your program's actual output. However, you do have to obey the spirit of the subject and turn in a program with consistent and relevant output formatting.

- For the same reasons, you are entirely free to choose your file names and general hierarchy. But keep in mind that a consistent and relevant file organization is a good code practice and will earn you bonus points.

- Try to respect the Symfony 2.8 coding standards as much as as possible, they can earn you bonus points.

- Other third party bundles or libraries can be used.

- Remember that no bonus points are awarded for the styling of the page, only the functionality and the code is all the matters.

Chapter III

Foreword

You will create a simple game similar to Pokemon using the OMDB API.The goal of the game is to capture all movies (referred to as Moviemons) and not lose all of your health in the process.

Chapter IV

Mandatory Part

The application should use the OMDB API and retrieve a list of at least 10 movies from it. The list can be random each time or the same movies can be shown for each game. The game starts on the Options page which should have the following buttons:

- New - reset the game to zero and start a new game.The player should have the ability to enter a name for the player.
- Save - ability to save the game to a json format in a file with the same name as the player.
- Load - ability to load a game from an existing save.A list with all the saved games should be shown with the ability to pick from any of them.
- Cancel - ability to get back to the world map (discussed below).Will only be shown if a game is in progress.

After a game is started the player is pressented with the world map page. The page should have a square table at least 5x5 and the player starts in the center of it.Using links he can move up, down, left or right. The current position of the player in the table should be marked accordingly. Each time a player moves he has a chance to trigger a fight with a Moviemon. The Moviemon is chosen at random from the Moviemons which are not yet captured.

The fighting page should contain the poster for the movie. The combat is turn based. Each Moviemon has a health and a strength attribute displayed on the page. These attributes can be random. The player also has the same attributes and he gets stronger with each captured Moviemon. On this fighting page, the player can fight or return to the world map.

If he fights, depending on the player and Moviemon stats he has a random chance to hit the Moviemon and deplete its health, but the Moviemon will also hit the player back if the player did not attack that turn. If the player's health reaches 0, then a Game Over message will be shown and the game will need to be reset. If the Moviemon's health is reduced to 0 he is captured and the player returns to the world map. The player can also

leave a fight and return to the world map at any time and he will recover his health, but will not capture the Moviemon.

On the world map screen the player can view all of his captured Moviemons and all the remaining ones. If the player has captured all the Moviemons, he will win the game and will see an appropriate message.He will have to start a new game from this point.

Chapter V

Bonus Part

You can do the bonus part but it will only be rated if the mandatory part was finished 100%.

Implement a detailed MovieDex: on a separate page, show all of your captured Moviemons.

Implement a detailed view for a Moviemon which should show the film director, year, rating, plot etc.

Implement dynamic attributes for Moviemons which are lower or higher depending on the rating of the movie that comes from the API.