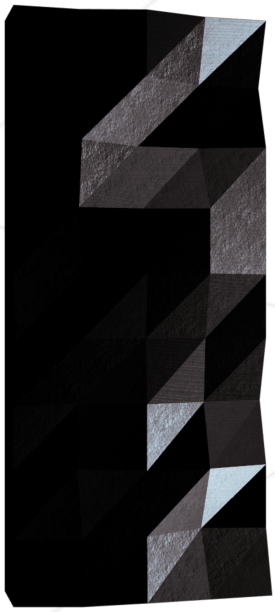

Formation PHP Symfony - 0

OOP

Summary: Following 42 formation course, you will learn about the basics of OOP and advanced functionality of PHP programming language.

Version: 1.1

Contents

Chapter I

Foreword

Today we are going to learn about Object Oriented Programming in PHP and how to write modular, scalable and reliable programs in PHP.

Chapter II

General rules

- Your project must be realized in a virtual machine.
- Your virtual machine must have all the necessary software to complete your project. These softwares must be configured and installed.
- You can choose the operating system to use for your virtual machine.
- You must be able to use your virtual machine from a cluster computer.
- You must use a shared folder between your virtual machine and your host machine.
- During your evaluations you will use this folder to share with your repository.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a `0` during the evaluation.
- We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Chapter III

Day-specific rules

If no other explicit information is displayed, you must assume the following versions of languages :

- `PHP - Symfony LTS`
- `HTML 5`
- `CSS 3`
- You should follow the requirements of every exercise exactly and respect the file naming conventions.

Chapter IV

Exercise 00

	Exercise 00
Exercise 00: HTML parameters	
Turn-in directory : $ex00/$	
Files to turn in : `TemplateEngine.php,index.php`	
Allowed functions :	

Given the template file **book_description.html**, you have to create a file **TemplateEngine.php** with a class **TemplateEngine** which contains a method **createFile($fileName, $templateName, $parameters)**. This method should generate a new HTML file with the name **$fileName** by replacing the parameters inside **{}** from the template file with the value of the parameters that come from the array **$parameters**.

Also create a file **index.php** which should instantiate the class and call the method with a set of arguments you specify.

Chapter V

Exercise 01

	Exercise 01
Exercise 01: Render object	
Turn-in directory : $ex01/$	
Files to turn in : `TemplateEngine.php,index.php,Text.php`	
Allowed functions :	

You must create a **Text** class who must have :

- a constructor with an array of strings as parameter.
- a method **append** to insert new strings into it
- a method **readData** to return all strings in HTML, each embedded in a **<p>** tag.

You then need to modify the **createFile($fileName, $templateName, $parameters)** method of the **TemplateEngine** class. It must take a **Text** object as the last parameter: **createFile($fileName, $text)**.

The new method freshly modified createFile must create a static HTML file file similar to the one generated in Exercise 00. The body must include the content rendered by the Text class.

Chapter VI

Exercise 02

	Exercise 02
Exercise 02: Render object 2	
Turn-in directory : $ex02/$	
Files to turn in : `TemplateEngine.php,index.php,HotBeverage.php,Coffee.php,Tea.php`	
Allowed functions :	

Create a base class **HotBeverage** with the attributes name, price and resistence and create getters for all of them.

Create two classes which extend the **HotBeverage** class, **Coffee** and **Tea** which both attribute a value to the name, price and resistence fields and also have the private attributes description and comment which should also have a value and a getter.

Modify the **TemplateEngine** class's **createFile(HotBeverage $text)** method to take a **HotBeverage** object and using the template file **template.html**, create a static HTML file with the name of the **HotBeverage**'s object class by replacing the parameters in the template with the value of the attributes which come from the object. The PHP **ReflectionClass** should be used to retrieve all the attributes of the class and then get the values using calls to the respective getters.

The **index.php** file should create a **Coffee** and **Tea** object and call the **createFile** function for both of them.

Chapter VII

Exercise 03

	Exercise 03
Exercise 03: Generate HTML	
Turn-in directory : $ex03/$	
Files to turn in : `TemplateEngine.php,index.php,Elem.php`	
Allowed functions :	

Create a class **Elem** which represents an HTML element tag.The class should have a constructor with the parameters:

- element - the HTML tag to use for the element
- content - the content of the tag (optional)

The class should support the following HTML tags: meta, img, hr, br, html, head, body, title, h1, h2, h3, h4, h5, h6, p, span, div.

The class should also have the method **pushElement** to add a new element to the content of the tag. A function **getHTML** should render the element as HTML code. If the content of the element contains other elements, those should be rendered as well.

Example:

```
$elem = new Elem('html');
$body = new Elem('body');
$body->pushElement(new Elem('p', 'Lorem ipsum'));
$elem->pushElement($body);

echo $elem->getHTML();
```

Should generate:

```
<html>
    <body>
        <p>Lorem ipsum</p>
    </body>
</html>
```

Modify the **TemplateEngine** class and add a constructor which takes an **Elem** object as a parameter. Then the **createFile($fileName)** should render the HTML output of the element to a file.

The **index.php** file should create some **Elem** objects and then save the rendered HTML to a file.

Chapter VIII

Exercise 04

	Exercise 04
Exercise 04: Generate HTML 2	
Turn-in directory : $ex04/$	
Files to turn in : `TemplateEngine.php,index.php,Elem.php,MyException.php`	
Allowed functions :	

Modify the **Elem** class from the previous exercise to throw an exception class **MyException** if an unauthorized tag is used in the constructor.

Add support for the following HTML tags also: table, tr, th, td, ul, ol, li.

Also modify the constructor to take a new optional parameter, **attributes**, which should contain as an array the attributes to render on the tag.

Example:

```
$elem = new Elem('html');
$body = new Elem('body');
$body->pushElement(new Elem('p', 'Lorem ipsum', ['class' => 'text-muted']));
$elem->pushElement($body);

echo $elem->getHTML();

$elem = new Elem('undefined');  // Throws MyException
```

Should generate:

```
<html>
    <body>
        <p class="text-muted">Lorem ipsum</p>
    </body>
</html>
```

Chapter IX

Exercise 05

	Exercise 05
Exercise 05: Validate HTML	
Turn-in directory : $ex05/$	
Files to turn in : `TemplateEngine.php,index.php,Elem.php,MyException.php`	
Allowed functions :	

Add a new function to your **Elem** class, **validPage()**, which should return either *true* or *false* if the HTML of the element is a valid webpage. You will have to traverse the elements and check if all of the following conditions are true:

- the parent node is **html** and contains exactly one node **head** followed by a node **body**
- the **head** should contain a single **title** tag and a single **meta charset** tag
- the **p** tag should not contain any other tags inside, only text
- the **table** tag should only contain **tr** tags and the **tr** tag should only contain **th** or **td** tags
- the **ul** and **ol** tags should only contain **li** tags

Add tests to validate that all of the above conditions are respected.

Chapter X

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.

`The evaluation process will happen on the computer of the evaluated group.`