



Training PHP Symfony - 1

Base of Symfony

Summary: Following this project you will get to know basic concepts in Symfony framework.

Version: 1

Contents

I	Foreword	2
II	General rules	3
III	Day-specific rules	4
IV	Exercise 00	5
V	Exercise 01	7
VI	Exercise 02	8
VII	Exercise 03	10
VIII	Submission and peer-evaluation	11

Chapter I

Foreword

The void type, in several programming languages derived from C and Algol68, is the type for the result of a function that returns normally, but does not provide a result value to its caller. Usually such functions are called for their side effects, such as performing some task or writing to their output parameters. The usage of the void type in such context is comparable to procedures in Pascal and syntactic constructs which define subroutines in Visual Basic. It is also similar to the unit type used in functional programming languages and type theory.

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


- Every exercise should be resolved in a different bundle.
- Every controller has to go under Controller folder inside the bundle.
- File names containing controller classes should be suffixed with **Controller** and should contain a class with the same name.
- The content of every page should be in twig files and every twig file should have the file extension `.html.twig`
- The server used for this day is the one integrated in Symfony. It should be started and stopped using Symfony console commands.
- Only explicit request URLs should render a page without error. The not configured URLs should generate a 404 error.
- The requested URLs should work with and without trailing slash. E.g: both `/ex00` and `/ex00/` must work

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

- PHP - Symfony LTS
- HTML 5
- CSS 3

Chapter IV

Exercise 00

	Exercise 00
Exercise 00: First page	
Turn-in directory : <i>ex00/</i>	
Files to turn in : All the files of the application	
Allowed functions : All the functionalities of Symfony	

In this exercise you have to create a simple page in a symfony application.

First, create a Symfony project using composer :

```
composer create-project symfony/skeleton d04 "~versionLTS"
```



You must change the `versionLTS` with the real LTS version.

Or, using symfony:

```
symfony new d04 --version=lts
```

Create a bundle `E00Bundle` and register it in the Bundle file.



You can create the bundle using the built-in commands from Symfony.


For defining the routes you'll have to use annotations in the controller file and you should register the routes inside the `/config/routing.yml` file.

For this exercise you have to create a page that is viewable on the following URL : `/e00/firstpage`. On this page you should display only the following string : `"Hello world!"`.

In this exercise you should not use any templates and you should have only the controller and you should use **Response** object from **HttpFoundation** component.

Chapter V

Exercise 01

	Exercise 01
Exercise 01: Multiple pages	
Turn-in directory : <i>ex01/</i>	
Files to turn in : All the files of the application	
Allowed functions : All the functionalities of Symfony	

Using the project already created in the first exercise create a new bundle **E01Bundle** and register it in the Bundle file.

You need to create an application that will display a single page split into 3 parts with two articles. You'll have to integrate only **.html.twig** files inside the application.

The application should contain a main page at the address **/e01** with at least a header a footer and the base page. These articles will be accessible through a URL structured as follows: **/e01/[articles]**. For example: **/e01/gull** will have to display the content of an article on gulls as well as the header and footer of your main page.

You must have at least 3 articles available, you can choose the subjects you want.


In case a url with a wrong category is accessed the main page should be displayed instead. So accessing **/e01/wrongurl** should display the main page that contains a list of link to the article pages.

You can define the list of categories in an array inside the controller or you can use set parameters inside the **/config/services.yml** file of the Bundle.

In order to make the pages render correct HTML you have to create a **base.html.twig** file that will contain the **<html>** and **<body>** tags of the page and all category templates should extend this template and overwrite the block **content** that should be also defined in the base template.

Chapter VI

Exercise 02

	Exercise 02
Exercise 02: First form	
Turn-in directory : <i>ex02/</i>	
Files to turn in : All the files of the application	
Allowed functions : All the functionalities of Symfony	

Using the project already created in the first exercise create a new bundle `E02Bundle` and register it in the `AppKernel` file.

In this exercise you'll have on the URL `/e02` a form which will have two inputs : a text field "Message" containing a message and a dropdown "Include timestamp" with two selectable values : Yes and No. These informations will have to be written in a file on the disk using the following business logic : if "Yes" is selected from the dropdown you have to write both the message and the timestamp of the message on a line in the file; if "No" is selected then you have to write only the message in the file.

You have to add a server-side validation on the Message field to not be blank. Be careful that the validation should be added on the server side, the HTML5 validation message is not enough.

The name of the file should be configurable inside `/config/services.yml` and should be created in the project root folder, next to `composer.lock` and `composer.json` files. If the file does not exist, the application should not fail, but it should create the file in the specified location and with the specified name.

After if the form is submitted several times the content of the file "notes.txt" should be updated with the information from the form on a new line.


And also after the form is submitted the page should refresh and the form should remember the submitted information and the last line added in the file should be displayed under the form.

You should not hardcode the form in HTML, but you should use the Form component

from Symfony and you should use the built-in types from symfony.

Chapter VII

Exercise 03

	Exercise 03
Exercise 03: Fifty shades of colors	
Turn-in directory : <i>ex03/</i>	
Files to turn in : All the files of the application	
Allowed functions : All the functionalities of Symfony	

Using the project already created in the first exercise create a new bundle **E03Bundle** and register it in the Bundle file.

In this exercise you'll have to create a page that will display a number of shades for the following colours : *black, red, blue, green*. The number of the shades should be configurable from **/config/services.yml** file in the parameter : **e03.number_of_colors**.

The page should contain also a header for the table with the colors for which the shades are displayed.

Table cells should have the following characteristics :

- Height : 40 pixels.
- Width : 80 pixels.
- Background color : a shade of the color corresponding to the column.

The table should have the number of lines specified in the **/config/services.yml** file in the parameter : **e03.number_of_colors**.

You are not allowed to hardcode the values in the HTML, you have to create them dynamically and send them to the Twig template as parameter from the controller.

Chapter VIII

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.