Formation PHP Symfony - 3

Final

Summary: Following 42 formation course, you will learn how you can create a one page app using Symfony, how to make Ajax calls from a browser and how to use Websockets for client-server communication.

Version: 2

Contents

Chapter I

Foreword

Today we are going to learn how we can make a simple one page application using the Symfony framework.

If you do not know what a one page application is, it is basically a website that does not require a page to be reloaded by the browser, everything on the page is dynamically updated using Ajax calls from javascript.

We are also going to take a look at Websockets, which allows two way client and server communication over a TCP connection.

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

- For this day, your repository must contain just one working Symfony application.
- Best practices of the Symfony framework should be respected.
- Other third party bundles or libraries CAN be used.
- Other actions CAN be created in the provided controllers, but no other controllers should be created.
- It is recommended to use services to organize your code.

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

- `PHP - Symfony LTS`
- `HTML 5`
- `CSS 3`

Chapter IV

Exercice 00

	Exercise
Exercice 00: Base Bundle & Basic entity	
Turn-in directory : *ex/*	
Files to turn in : `Files and folders from your application`	
Allowed functions : `All methods`	

For this exercise you have to set up a new Symfony application and work within the default application structure, which will be extended by the following exercises. Focus on integrating your work directly into the application's default namespace. Ensure the application structure does not contain any unnecessary folders.

You will have to create a new entity called **Post**. This entity should have at least the following fields, with their types being self-explanatory:

- id
- title
- content
- created

The **created** field should take the value of the date the object was created at using Doctrine Events.

Hint: *The command* ***bin/console make:entity*** *can be used to generate entitiesin the latest versions of Symfony, streamlining the process of creating entities and handling database operations.*

Chapter V

Exercice 01

	Exercise
Exercice 01: Post form & Security	
Turn-in directory : *ex/*	
Files to turn in : `Files and folders from your application`	
Allowed functions : `All methods`	

Only users logged in to the application should be able to create new posts.For this you need to set up security and create some simple users.Create a controller **UserController** with a **loginAction** for the route **/login**. This action should return a template **login.html.twig** which should contain a login form.The form will need to be submitted via Ajax.

Also create a controller **PostController** with an action **defaultAction** for the default path which should return a template called **index.html.twig**. The template should either display the login form if the user is not logged in or a post form. The post form should contain the **title** and **content** fields and should be created using a post type class.

The login form needs to be submitted via Ajax and if the credentials are correct, instead of the login form the post form will need to be displayed. If the credentials are not correct, a browser alert should be shown.

The post form should also be submitted via Ajax and display a confirmation message if the post was successfully added.

Hint: *Symfony provides a base JsonResponse class, authentication handlers and ability to make subrequests from templates.Also make sure to secure your controller actions!*

Chapter VI

Exercice 02

	Exercise
Exercice 02: Post List	
Turn-in directory : *ex/*	
Files to turn in : `Files and folders from your application`	
Allowed functions : `All methods`	

Create a basic list below the form on the main page. The list should display all the posts created and their title and creation date.The list can be viewed even if a user is not authenticated.

Now modify the submitting of the form to also update this list and add newly created posts to the end of it after the Ajax request is finished.

Also implement validation for the form to make sure that the title of each post is unique. If you try to add a post with a title that already exists, you should get a browser alert with an error message.

Hint: *It is highly recommended to use translations for the form field labels and error messages. And remember, the browser page should never reload!*

Chapter VII

Exercice 03

	Exercise
Exercice 03: Post Details & Deletion	
Turn-in directory : *ex/*	
Files to turn in : `Files and folders from your application`	
Allowed functions : `All methods`	

Create an action in **PostController** called **viewAction** with the route **/view/{id}** which should be called using Ajax when clicking on the title of a post and return all of the post details.The details should then be displayed above the form.

Create another action in the same controller called **deleteAction** with the route **/delete/{id}** which when called from Ajax should delete the post. Only logged in users should be able to delete a post. The post details should contain a delete button which should show a confirmation dialog and then delete the post.

It should also update the posts list and remove the deleted post from it.

Chapter VIII

Exercice 04

	Exercise
Exercice 04: Websockets!	
Turn-in directory : *ex/*	
Files to turn in : `Files and folders from your application`	
Allowed functions : `All methods`	

It is now time to learn about Websockets.You probably have not heard much about them, but now is your chance to learn more!

You will first set up the Websocket server using Symfony on port 8080. The server should be started using a custom Symfony command:

bin/console websocket:server

On the client side, you will have to create a new Websocket using Javascript. Now, each time a new post is added in any browser window, you will have to update the posts list of all the clients connected to the Websocket server to display the new post.

Make sure to also modify the delete form to remove the post from all the browser windows.

Hint: *This exercise can be solved in multiple ways. A variety of libraries can be found online that will help you with implementing mostly all of this functionality.*

Chapter IX

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.

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