KNP Labs

Symfony4 Training

Basis for the PHP developer

- **3 days** (21 hours)
- **Up to 8 participants**

Contact :
hello@knplabs.com

KNP Labs

Training goal

This training will teach the participants the basic mechanisms of the Symfony 4 PHP framework with Doctrine 2.* ORM (latest version).

Detailed contents

- The training is regularly updated to be consistent with the latest Symfony evolutions.
- The training material is a pastry related web application project, that will be developed by each participant. Hands on work takes place throughout the training, so that participants progressively learn how to correct their mistakes.
- The emphasis is on **good practices**.

Program

1. **Introduction**
   1.1 Why use a framework
   1.2 Symfony features
   1.3 The Community

2. **Installation & configuration**
   2.1 Composer
   2.2 Install Symfony with `composer create-project`
   2.3 Symfony Flex
   2.4 Symfony project architecture
   - Architecture of Symfony3
   - Architecture of Symfony4

3. **Configuration file**
   3.1 Configuration language possibility: yml / xml / annotation, and why?
   3.2 Symfony environment
   3.3 Symfony recommandations & what we are going to use during this training

4. **Create a first action**
   4.1 Create a controller that just echoes a string
   4.2 Create a routing

5. **Autoloading**
   5.1 Composer autoloader
   5.2 PSR-4

6. **Display the cake list**
   6.1 Use PDO to fetch cakes
   6.2 Display them with `print_r`

7. **Twig**
   7.1 Basic instructions
   7.2 Symfony integration
   - Show the controller `render` method

   7.3 Conventions about template locations and names
   7.4 Create a first template to list cakes
   7.5 Render this template from the controller

8. **Assets (adding css)**
   8.1 What are assets?
   8.2 Use `webpack-encore` to compile assets

9. **Debugging**
   9.1 Debug toolbar
   9.2 `dump()`

10. **Symfony configuration**
    10.1 Parameters
    - Parameters provided by Symfony (i.e: `kernel.root_dir`)

    10.2 How to get a parameter from a controller
    10.3 Use the Symfony parameters to configure PDO from the controller

11. **Routing parameters**
    11.1 How to define route parameters
    11.2 How to get these parameters from the controller
    11.3 Create a `display cake` route using a route parameter

12. **Twigs paths & urls**
    12.1 How to generate paths & urls from a twig template
    12.2 Add links to the toy project templates

13. **Refactoring the controller**
    13.1 Créate a singleton to bootstrap PDO inside the controller

14. **Twig inheritance & blocks**
    14.1 How to extend a template

KNP Labs

14.2 How to define blocks
14.3 Create a layout template
14.4 Extend this layout

15. **HTTP status codes**
    15.1 What is a status code
    15.2 How to set the response status code (throwing http exception)
    15.3 Throw an exception when a cake is not found
    15.4 Explain error messages in the profiler

16. **Create a cake (part 1)**
    16.1 Create controller, template, routing
    16.2 Meet a calculated error on the browser: "the cake with id `create` does not exist"

17. **Routing refinement**
    17.1 How to add requirements

18. **Create a cake (part 2)**
    18.1 Use `$_POST` to get results
    18.2 Manually insert data with PDO
    18.3 Add a redirection to cakes list when the cake has been created
    18.4 Some explanation about `RedirectResponse`

19. **Symfony request**
    19.1 Show some available parameter bags (mainly `request` & `querry`)
    19.2 How to get the Request in a Controller
    19.3 Remove the `$_POST` usages and replace them by the request

20. **Request Session (to add flash message)**
    20.1 Describe what is the session bag
    20.2 How to add flash messages in the session flash bag
    20.3 Use the controller shortcut `addFlash()`
    20.4 Display flash messages from a twig template

21. **Symfony forms (part 1)**
    21.1 How `FormType` are composed
    21.2 Introduction to the `FormBuilder`
    21.3 Create a create cake `FormType` with simple fields
    21.4 How to instantiate the form from a controller (`createForm()`)

22. **Render a Symfony form**
    22.1 Twig functions for forms
    - `form_start`
    - `form_widget` & `form_row`
    - `form_end`

    22.2 Render our create cake `FormType`

23. **Symfony forms (part 2)**
    23.1 How to handle the request
    23.2 How to get form data
    23.3 Replace the request usage in the controller by the form data

24. **Symfony validation**
    24.1 How it works
    24.2 How to put some constraints on both Entities or Forms fields
    24.3 How to use the `isValid()` method
    24.4 Add some validation to our form

25. **Doctrine**
    25.1 `entity manager`
    25.2 A few words about ORMs
    25.3 What is an entity
    25.4 How to map an entity (`yml`)
    - How to configure doctrine bundle to use `yml` mapping

    25.5 Configure the DB connection
    25.6 Write in the DB, `persist()` & `flush()`
    25.7 Create a cake model
    - Class, mapping file

    25.8 In the controller
    - Create a cake model from the form data
    - Persist and flush this entity with the entity manager
    - Get rid of the PDO

    25.9 A few words about DQL
    25.10 `DoctrineMigration`

26. **Symfony forms (part 3)**
    26.1 How to configure the form to get instanciated Entity instead of array
    - How to use `data_class`

27. **Security**
    27.1 user providers
    27.2 password encoders
    27.3 firewall
    27.4 access_control (URL protection)
    27.5 Cheking rights (is_granted / voter)

27.6 User roles hierarchy

28. **Internationalization**
    28.1 How to setup translations
    28.2 How to use them in twig
    28.3 How to use placeholders in translations
    28.4 Entity translations

29. **Logging**
    29.1 Monolog

30. **Swiftmailer**

31. **Dependency injection (DIC)**
    31.1 How to declare services
    31.2 Kernel events

32. **Symfony commands**
    31.1 Console component

33. **Bonus topics**
    34.1 Behat
    34.2 Phpspec
    34.3 FosJsRouting
    34.4 Event Listener / Subscriber
    34.5 Dispatching d'events
    34.6 API REST / Serialization
    34.7 Useful bundles recommended by KNP Labs

Prerequisites

To obtain the best learning experience, the participants must have some professional experience in the following:

- ➔ PHP5 object oriented development
- ➔ Using a relational database

To be able to perform all the practical exercises throughout the training, participants must have a computer with admin rights and the following LAMP environment preferably including the following:

- Unrestricted internet access.
- A good PHP code editor (sublime text, php storm ...)
- A UNIX shell
- PHP 7.1 ≤ installed (as recent a version as possible)
- Git installed (useful for composer dependencies)
- One mysql server per participant

It is recommended for the participants to use a text editor they are comfortable with, so that they can focus on the training. **Installing Symfony before the training is no necessary, participants will install it with the trainer during the training**.

If the training is held in the Client's premises, we require for the Client to provide a separate **room** for the training to take place, big enough to accommodate all participants and the trainer, for the whole duration of the training. A **video projector** or wide screen TV must be provided so that the participants can see the trainer's laptop computer screen. A paperboard or whiteboard, and Internet access would be appreciated.

Provided learning material

The training will give out training material in digital format at the end of the training.

During this 3 day training, participants will create a web application versioned step by step. The trainer will also give out his application at the end of the training. Each participant is advised to keep on their PC the training project they wrote with the trainer, since this code can later be used as technical basis for other projects.