SIN6U5 : Développement web 2
(Prog. web coté serveur)
*Découverte de **Symfony***

Luigi Santocanale
LIF, Aix-Marseille Université

22 février 2018

Plan

Plan

Les frameworks

C'est quoi un framework :

- Ensemble de bibliothèques (composantes)
- . . . pour le développement web (dans notre cas)
- Exemples :
  - mailer, templates, formulaires, persistance (BD) . . .

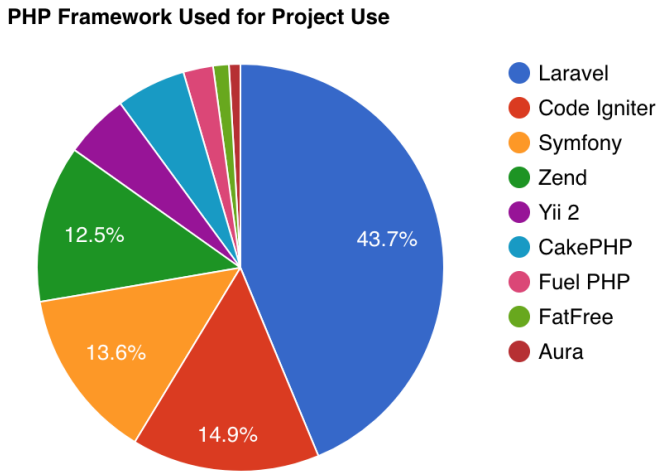
Pourquoi utiliser un framework :

- bonnes pratiques de programmation
- . . . permettant le développent en équipe
- . . . et l'intégration avec du code « third party »
- bibliothèques testées et fiables
- développement/prototypage rapide
- outils pour le test et le déploiement.

Le framework ***Symfony***

- Lien : `https://symfony.com/`,
- Auteur : Fabien Potencier@SensioLabs
- Developpé à partir de 2004
- Version actuelle : 4
- Version utilisée en cours : 3.4
- Open source, licence MIT :
  Licence permissive, permettant la réutilisation dans du code propriétaire à la condition que la licence soit redistribuée dans le code.

Des frameworks PHP



Source : `https://coderseye.com/best-php-frameworks-for-web-developers/`

Remarques :

- Laravel est construit autour de **Symfony**
- Selon le site ci-dessus : apprentissage de **Symfony** serait difficile
- SensioLabs confirme ! ! !

Plan

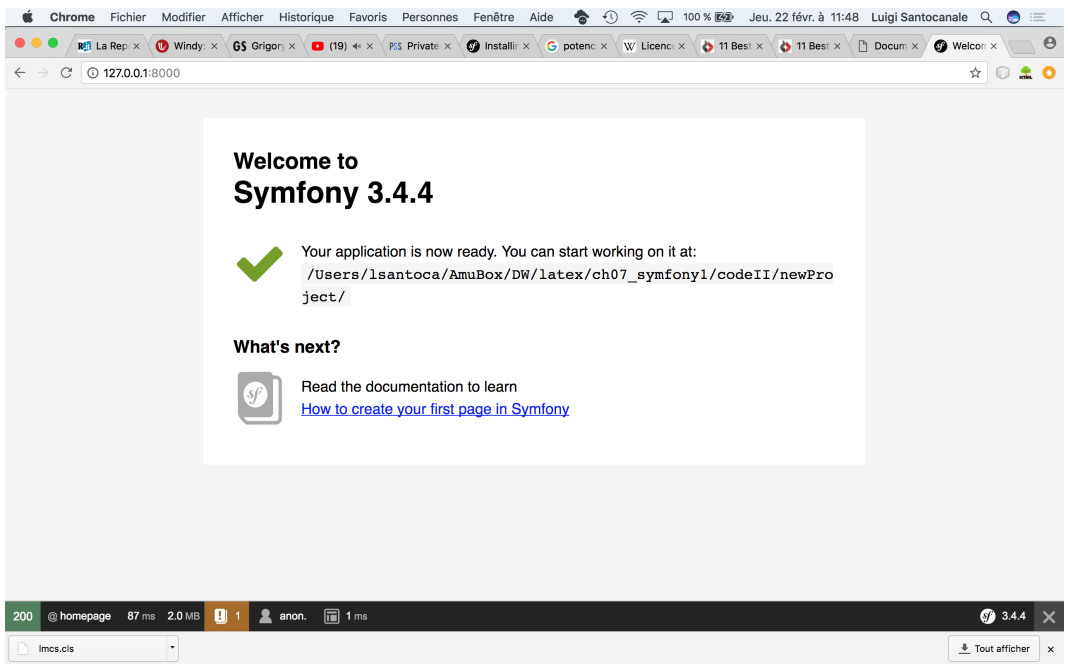
Objectifs

- Reorganiser le projet `Courses` (`premierMVC`),
  mais en utilisant Symfony.
- Avoir une première idée de comment Symfony fonctionne.
- Exo TD/TP :
  réorganiser le projet `Utilisateurs` en un projet Symfony.

Création du projet

Création du projet :

```
#!/bin/bash
monProjet=newProject
composer create-project symfony/framework-standard-edition ${monProjet}
  ↪   "3.4.*"
cd ${monProjet}
# Symfony  > 4
#composer require server --dev
php bin/console server:run
```



Structure d'une application Symfony

```
newProject/
 LICENSE
 README.md
 app
 bin
 composer.json
 composer.lock
 phpunit.xml.dist
 src
 tests
 var
 vendor
 web

7 directories, 5 files
```

Plan

Paramétrage des moteurs de templates

Symfony utilise des fichiers au format `yaml` pour la configuration.

On souhaite utiliser des templates `html+php`

`app/config/config.yml`

```
framework:

    templating:
        engines: ['twig', 'php']
```

Remarque : l'autre moteur de templates est `twig` (compilé).

Paramétrage de la base de données

On souhaite utiliser une base de données sqlite

app/config/config.yml

```
# Doctrine Configuration
doctrine:
    dbal:
        #driver: pdo_mysql

        driver: pdo_sqlite
        # if using pdo_sqlite as your database driver:
        #   1. add the path in parameters.yml
        #     e.g. database_path:
 ↪   '%kernel.project_dir%/var/data/data.sqlite'
        #   2. Uncomment database_path in parameters.yml.dist
        #   3. Uncomment next line:
        path: '%database_path%'
```

app/config/parameters.yml

```
# This file is auto-generated during the composer install
parameters:

    database_path: '%kernel.project_dir%/var/db/database.sqlite'
```

Paramétrage : les annotations

On souhaite utliser des annotations pour les routes :

`app/config/routing.yml`

```
app:
    resource: '@AppBundle/Controller/'
    type: annotation
```

et le modèle :

`app/config/config.yml`

```
    orm:

        mappings:
            AppBundle:
                type: annotation
```

Plan

Engendrer les classes du modèle

Modification de la base de données originaire,
création de la colonne `id` sur la table `choices` :

```
CREATE TABLE choices (
       id INTEGER PRIMARY KEY AUTOINCREMENT,
       student_id INTEGER,
       course_id INTEGER,
       UNIQUE(student_id, course_id),
       FOREIGN KEY(student_id)
                       REFERENCES students(id) ON DELETE CASCADE,
       FOREIGN KEY(course_id)
                       REFERENCES courses(id) ON DELETE CASCADE
);
```

Création de la base de données et des classes à partir de la BD :

```
#!/bin/bash
monProjet=newProject
cd ${monProjet}
sqlite3 var/db/database.sqlite < var/db/create.sql
php bin/console doctrine:mapping:import --force AppBundle yml
php bin/console doctrine:mapping:convert annotation ./src
```

Les entités

```php
<?php

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Students
 *
 * @ORM\Table(name="students")
 *
 ↪   @ORM\Entity(repositoryClass="AppBundle\Repository\StudentsRepository")
 ↪
 */
class Students
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;
```

Les entités

```
    /**
     * @var string
     *
     * @ORM\Column(name="firstname", type="text", nullable=true)
     */
    private $firstname;

    /**
     * @var string
     *
     * @ORM\Column(name="lastname", type="text", nullable=true)
     */
    private $lastname;

}
```

Plan

Le contrôleur Registration

```php
<?php

// src/AppBundle/Controller/RegistrationController.php

namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use AppBundle\Entity\Students;
use AppBundle\Entity\Choices;
use AppBundle\Entity\Courses;

//use Doctrine\ORM\EntityManagerInterface;

class RegistrationController extends Controller {
    /* -> index.php/registration/registrations */

}
```

Utilisation des annotations pour les routes

```php
class RegistrationController extends Controller {
    /* -> index.php/registration/registrations */


    /**
     * @Route("/")
     * @Route("/registration/registrations")
     */
    public function registrations() {

        $data['registrations'] = $this->findAllRegistrations();
        return $this->render('registration/registration_list.html.php',
 ↪   $data);
    }

}
```

Utilisation de l'ORM Doctine

```php
private function findAllRegistrations() {

    $entityManager = $this->getDoctrine()->getManager();
    $choicesRepository = $entityManager->getRepository(Choices::class);
    $coursesRepository = $entityManager->getRepository(Courses::class);
    $studentsRepository = $entityManager->getRepository(Students::class);

    $students = $studentsRepository->findAll();
    $callback = function($choice) use ($coursesRepository) {
        $course = $coursesRepository->findOneById($choice->getCourseId());
        return $course->getName();
    };

    $registrations = [];
    foreach ($students as $index => $student) {
        $registrations[$index]['studentId'] = $student->getId();
        $registrations[$index]['firstname'] = $student->getFirstname();
        $registrations[$index]['lastname'] = $student->getLastname();
        $choices = $choicesRepository->findByStudentId($student->getId());
        $courses = array_map($callback, $choices);
        $registrations[$index]['courses'] = implode(',', $courses);
    }
```

Contrôleur Registration

Réponse http en renvoyant une template complété :

```
/**
 * @Route("/registration/new")
 */
public function registrations_new() {
    $data['courses'] = $this->findCourses();
    return $this->render('registration/registration_form.html.php', $data);
}
```

Utilisation de l'ORM :

```
private function findCourses() {
    $entityManager = $this->getDoctrine()->getManager();
    return $entityManager->getRepository(Courses::class)
                    ->findAll();
}
```

ORM : gestion de la persistance sans `sql`, I

```
public function delete($studentId) {
    $this->deleteRegistration($studentId);
    return $this->redirectToRoute('app_registration_registrations');
}
```

```
public function deleteRegistration($studentId) {
    $entityManager = $this->getDoctrine()->getManager();
    $studentsRepository = $entityManager->getRepository(Students::class);
    $choicesRepository = $entityManager->getRepository(Choices::class);

    $student = $studentsRepository->findOneById($studentId);
    $choices = $choicesRepository->findByStudentId($studentId);

    $entityManager->remove($student);
    foreach ($choices as $choice) {
        $entityManager->remove($choice);
    }

    $entityManager->flush();
}
```

ORM : gestion de la persistance II

```php
/**
 * @Route("/registration/create")
 */
public function create() {
    $form_data = filter_input_array(INPUT_POST, self::FORM_FILTERS, true);
    $error = $this->compute_error($form_data, self::FORM_FILTERS);
    if ($error !== false) {
        $data = ['courses' => $this->model->getCourses(),
            'error' => $error];
        return $this->render('registration/registration_form.html.php', $data);
    }
    $this->addRegistration(
            $form_data['firstname'], $form_data['lastname'], $form_data['courses']);
    return $this->redirectToRoute('app_registration_registrations');
}
```

ORM : gestion de la persistance . . .

```php
private function addRegistration($firstname, $lastname, $choices) {
    $entityManager = $this->getDoctrine()->getManager();

    $student = new Students();
    $student->setFirstname($firstname);
    $student->setLastname($lastname);
    $entityManager->persist($student);
    $entityManager->flush();

    foreach ($choices as $index => $ch) {
        $choice[$index] = new Choices();
        $choice[$index]->setStudentId($student->getId());
        $choice[$index]->setCourseId($ch);
        $entityManager->persist($choice[$index]);
    }
    $entityManager->flush();
}
```

Les vues

Combiner les views :

```
<!-- app/Resources/views/registration/registration_form.html.php -->
<?php

function set_value($name) {
    return filter_input(INPUT_POST, $name,
 ↪   FILTER_SANITIZE_SPECIAL_CHARS, ['options' => ['default' => '']]);
}

$view->extend('registration/layout.html.php');
?>
```

Les vues

Le container `$view` :

```
<!-- app/Resources/views/registration/registration_form.html.php -->

<br />
<div class="container">
    <form method="post"
          action="<?= $view['router']->path('app_registration_create')
 ↪   ?>">

        <?php if (isset($error)): ?>
            <div class="alert"><?= $error ?></div>
        <?php endif; ?>

        <input type="text" name="firstname"
               value="<?= set_value('lastname') ?>"
               placeholder="Votre nom">
        <input type="text" name="lastname"
               value="<?= set_value('firstname') ?>"
               placeholder="Votre prénom">

        <?php foreach ($courses as $course): ?>
            <input type="checkbox" name="courses[]"
                   value="<?= $course->getId() ?>">
                   <?= $course->getName() ?>
                <?php endforeach; ?>
        <br />
        <button type="submit">Valider</button>
    </form>
</div>
```

Les vues

```
<!-- app/Resources/views/registration/layout.html.php -->
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Inscription</title>
        <link href="/assets/css/bootstrap.css"
              rel="stylesheet">
        <link href="/assets/css/style.css"
              rel="stylesheet">
    </head>
    <body>
        <?php echo $view->render('registration/top_bar.html.php',[]) ?>
        <?php $view['slots']->output('_content') ?>
    </body>
</html>
```

Consistance des routes

Quelles sont les routes définies :

```
>php bin/console debug:router
```

```
 ---------------------------------- -------- -------- ------ -----------------------------------
  Name                               Method   Scheme   Host   Path
 ---------------------------------- -------- -------- ------ -----------------------------------
  _wdt                               ANY      ANY      ANY    /_wdt/{token}
  _profiler_home                     ANY      ANY      ANY    /_profiler/
  _profiler_search                   ANY      ANY      ANY    /_profiler/search
  _profiler_search_bar               ANY      ANY      ANY    /_profiler/search_bar
  _profiler_phpinfo                  ANY      ANY      ANY    /_profiler/phpinfo
  _profiler_search_results           ANY      ANY      ANY    /_profiler/{token}/search/results
  _profiler_open_file                ANY      ANY      ANY    /_profiler/open
  _profiler                          ANY      ANY      ANY    /_profiler/{token}
  _profiler_router                   ANY      ANY      ANY    /_profiler/{token}/router
  _profiler_exception                ANY      ANY      ANY    /_profiler/{token}/exception
  _profiler_exception_css            ANY      ANY      ANY    /_profiler/{token}/exception.css
  _twig_error_test                   ANY      ANY      ANY    /_error/{code}.{_format}
  homepage                           ANY      ANY      ANY    /test
  app_registration_registrations     ANY      ANY      ANY    /
  app_registration_registrations_1   ANY      ANY      ANY    /registration/registrations
  app_registration_registrations_new ANY      ANY      ANY    /registration/new
  app_registration_delete            ANY      ANY      ANY    /registration/delete/{studentId}
  app_registration_create            ANY      ANY      ANY    /registration/create
 ---------------------------------- -------- -------- ------ -----------------------------------
```

Plan

Des conclusions

- gestion des routes : traductions
  $$\begin{aligned} \text{url} &\mapsto \text{contrôleur} \\ \text{contrôleur} &\mapsto \text{url} \end{aligned}$$ automatisées
- utilisation des metadonées (les annotations)
- environnement de prog. uniforme (pas de `sql`)
- gestion de la persistance (des DBs) depuis
- templates (views) composables,
- choix du langage de templates,
- outils d'aide au développement :
  - voir la barre en bas du site . . .
  - voir le navigateur lors d'un erreur PHP
  - l'outil `bin/console`
- . . . aux tests, au déploiement . . .