



Formation PHP 7 Symfony 4.3

Hajer ALAYA

Hajer.adahmeni@gmail.com

Du 15/07/2019 Au 19/07/2019



Plan

- Notions avancées
 - Les formulaires et leur validation

Les formulaires et leur validation



- Il existe plusieurs manières de traiter les formulaires en Symfony 4 :
- Tester un formulaire simple dans le projet JobRapid
- Dans la page index.html. Twig ajouter :

```
<form action="{{path('Detail_Offre',{'jobId': job.id})}}" method="POST" >  
    <input class="btn btn-primary" type="submit" value="See more"/>  
</form>
```

- Ajouter dans le fichier routes.yaml

```
Detail_Offre:  
    path: /DetailOffre/{jobId}  
    defaults: { _controller: 'App\Controller\JobController::DetailOffre' }
```



Les formulaires et leur validation



- Ajouter au niveau du JobController la méthode Detailoffre

```
class JobController extends AbstractController
{
    public function index(EntityManagerInterface $em): Response
    {
        $jobs = $em->getRepository(Job::class)->findActive(new DateTime('-30 day'));
        return $this->render('job/index.html.twig', ['jobs' => $jobs,    ]);
    }
    public function DetailOffre($jobId): Response
    {
        $em = $this->getDoctrine()->getManager();
        $job = $em->getRepository(Job::class)->findJobbyId($jobId);

        return $this->render('job/detailOffre.html.twig', ['jobs' => $job]);
    } }
```

Les formulaires et leur validation



- Ajouter au niveau du JobRepository la méthode Detailloffre

```
public function findJobbyId($Id)
{
    return $this->createQueryBuilder('j')
        ->andWhere('j.id = :Id')
        ->setParameter('Id', $Id)
        ->getQuery()
        ->getResult();
}
```

Les formulaires et leur validation



- Ajouter le fichier twig detailOffre.html.twig

```
{% extends "base.html.twig" %}
{% block body %}
    <h1 class="my-4">Détail de l'offre</h1>
    {% for job in jobs %}
        <div class="row">
            <div class="col-md-7">
                <a href="#">
                    
                </a>
            </div>
            <div class="col-md-5">
                <h3>{{ job.position }}</h3>
                <p>{{ job.description }}</p>
                <p>Posted on {{ job.createdAt|date("m/d/Y") }}</p>
            </div>
        </div> <hr>
    {% endfor %}
{% endblock %}
```

Les formulaires et leur validation



- Il existe plusieurs manières de traiter les formulaires en Symfony 4
- Lancer les commandes :

```
composer require symfony/form
```

```
composer require symfony/validator
```

- Création de formulaire pour la saisie des villes

1. Création de l'entity Ville
2. Création de formulaire

Un type de formulaire est une classe qui permet de construire un formulaire en définissant les différents types de champs.

3. Affichage dans Twing
4. Validation du formulaire
5. Mise en forme du formulaire

Création de l'entity Ville et du fichier Ville.orm.yaml

```
<?php
```

```
namespace App\Entity;
```

```
use Doctrine\Common\Collections\ArrayCollection;
```

```
class Ville
```

```
{ /** @var int */
```

```
    private $id;
```

```
    /** @var string */
```

```
    private $titre;
```

```
    /** @var Job[] */
```

```
    private $content;
```

```
    public function getId() { ...3 lines }
```

```
    public function getTitre() { ...3 lines }
```

```
    public function getContent(): array { ...3 lines }
```

```
    public function setId($id) { ...3 lines }
```

```
    public function setTitre($titre) { ...3 lines }
```

```
    public function setContent(array $content) { ...3 lines }
```

```
App\Entity\Ville:
```

```
    type: entity
```

```
    id:
```

```
        id:
```

```
            type: integer
```

```
            generator:
```

```
                strategy: AUTO
```

```
    fields:
```

```
        title:
```

```
            type: string
```

```
            length: 63
```

```
        content:
```

```
            type: string
```

```
            length: 63
```

Création du formulaire

```
<?php
// src/Form/VilleType.php
namespace App\Form;
use App\Entity\Ville;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type\TextareaType;
class VilleType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('title')
            ->add('content', TextareaType::class)
    }

    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults([
            'data_class' => Ville::class,
        ]);
    }
}
```



• I Création le controller « FormController »

```
<?php
// src/Controller/FormController.php
namespace App\Controller;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;
use App\Form\VilleType;
use App\Entity\Ville;
class FormController extends AbstractController
{
    public function newVille(Request $request)
    {
        $ville = new Ville();
        $ville->setTitle('Hello World');
        $ville->setContent('Un très court article. ');
        $form = $this->createForm(VilleType::class, $ville);
        $form->handleRequest($request);
        if ($form->isSubmitted() && $form->isValid()) {
            // dump($article);
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->persist($ville);
            $entityManager->flush();
        }
        return $this->render('job/new.html.twig', array(
            'form' => $form->createView(),
        ));
    }
}
```

•Ajouter le fichier twig

```
{# templates/job/new.html.twig #}
{{ form(form) }}
```

•Ajouter dans le fichier routes.yaml

```
new_Ville:
    path: /new_Ville
    defaults: { _controller: 'App\Controller\
                    FormController::newVille' }
```

. Tester votre projet

- Validation du formulaire
 - Créer un dossier « validation » sous le dossier config et ajouter le fichier validation.yaml

App\Entity\Ville:

properties:

title:

- NotBlank: ~

content:

- NotBlank: ~

Modifier la page new.html.twig comme suite :

```
<html>
<head></head>
<body>
    {{ form_start(form) }}
    {{ form_row(form.title) }}
    {{ form_row(form.content) }}
    <button type="submit" class="btn btn-primary">Créer</button>
    {{ form_end(form) }}
</body>
</html>
```

Lancer le serveur et pointer vers http://localhost:8000/new_Ville

- Mis en form du formulaire avec Bootstrap
 - Ouvrir le fichier twig.yaml et ajouter la dernière instruction

```
# config/packages/twig.yaml
twig:
  debug: '%kernel.debug%'
  strict_variables: '%kernel.debug%'
  form_themes: ['bootstrap_4_layout.html.twig'] # accepte plusieurs thèmes
```

Modifier la page new.html.twig comme suite :

```
<html>
  <head>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
  </head>
  <body>
    <div class="container">
      <div class="row">
        {{ form_start(form) }}
        {{ form_row(form.title) }}
        {{ form_row(form.content) }}
      <button type="submit" class="btn btn-primary">Créer</button>
      {{ form_end(form) }}
    </div>
  </body>
</html>
```

Lancer le serveur et pointer vers
http://localhost:8000/new_Ville