Doc personnelle PPE B2

Noémie LECHERF

Notre mission était de créer le MCD pour le projet Innov'Anglais ainsi que 2 formulaires avec ajout suppression et modification.

Avec mon groupe nous avons donc répartie les tâches.

Bastien s'est donc chargé du MCD, de la base de données avec l'entière création des entités des liaisons, migrations etc….

Quant à moi, je me suis chargée (avec son aide) des formulaires.

Voici donc les étapes de la création d'un formulaire :

Tout d'abord, il faut créer un contrôleur, pour cela il suffit de taper la commande :

php bin/console make:controller

Suivi du nom du contrôleur.

Ensuite, il faut créer un formulaire, pour cela on tape la commande suivante :

php bin/console make:form

Suivi du nom qu'on veut donner au formulaire (exemple : AjoutRoleType pour le formulaire concernant l'ajout d'un rôle)

Puis on indique le nom de l'entité à laquelle correspond ce formulaire.

On se rend ensuite à l'interieur de ce fichier form (RoleType) et on y fait quelques modifications, on rajoute le type des valeurs à rentrer(ex : TextType) ce qui donne :

```php
<?php

namespace App\Form;

use App\Entity\Role;
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\SubmitType;

class AjoutRoleType extends AbstractType
{
```

```
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('libelle', TextType::class)
            ->add('ajouter', SubmitType::class)
        ;
    }

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

Ensuite, on modifie le code contenu dans le controleur pour pouvoir appeler le formulaire.

Ce qui donne :

```
/**
    * @Route("/ajout_role", name="ajout_role")
    */
    public function ajoutRole(Request $request)
    {
        $role = new Role(); // Instanciation d'un objet Role
        $form = $this->createForm(AjoutRoleType::class,$role);
        // Création du formulaire pour ajouter un role, en lui donnant l'insta
nce.
        if ($request->isMethod('POST')) {
            $form->handleRequest($request);
            if ($form->isSubmitted() && $form->isValid()) {
                $em = $this->getDoctrine()->getManager();
                // On récupère le gestionnaire des entités
                $em->persist($role); // Nous enregistrons notre nouveau role
                $em->flush(); // Nous validons notre ajout
                 $this->addFlash('notice', 'Role inséré');
                // Nous préparons le message à afficher à l'utilisateur sur la
 page où il se rendra
                }
            return $this->redirectToRoute('ajout_role');
            // Nous redirigeons l'utilisateur sur l'ajout d'un role après l'in
sertion.
            }
            return $this->render('role/ajout_role.html.twig', [
            'form'=>$form->createView() // Nous passons le formulaire à la vue
        ]);
    }
```

Ensuite on crée la vue (ex : ajout_role.html.twig) du formulaire et on importe le formulaire à l'intérieur

Voici le code contenu dans ce ajout_role.html.twig :

```
{% extends 'base.html.twig' %}

{% block title %}{{ parent()}} - Ajout Role{% endblock %}
{% block content %}

 <div class="container-fluid">
 <div class="row justify-content-center">
 <h1 class="text-center text-primary p-4"> Ajouter un role</h1>
 </div>
 <div class="row justify-content-center">
 <div class="col-12 bg-white p-4 m-0 text-primary">
 {{ form_start(form) }}
<div class="form-group">
 {{ form_label(form.libelle, 'Nom du role (*) ', {'label_attr': {'class': 'fon
t-weightbold'}}) }}
 {{ form_widget(form.libelle, {'attr': {'class': 'form-control'}}) }}
 </div>
 <div class="form-group mx-auto text-center">
 {{ form_widget(form.ajouter, { 'label': "Ajouter", 'attr': {'class': 'btn fon
t-weightbold bg-primary text-white mx-auto text-center'}}) }}
 </div>
 {{ form_end(form) }}
 </div>
 </div>
</div>
{% endblock %}
```

On peut ici voir qu'on appelle bien le formulaire.

Cela donne :

C'est donc tout pour le formulaire, on passe ensuite à la liste des valeurs contenues dans la table rôle :

Pour cela, on crée une fonction listeRole dans le Controller Rôle :

```
    /**
    * @Route("/liste_roles", name="liste_roles")
    */
    public function listeRoles(Request $request)
    {
        $em = $this->getDoctrine();
        $repoRole = $em->getRepository(Role::class);

        if ($request->get('supp')!=null){
            $role = $repoRole->find($request->get('supp'));
            if($role!=null){
                $em->getManager()->remove($role);
                $em->getManager()->flush();
            }
            return $this->redirectToRoute('liste_roles');
        }
```

➔ Cette fonction va permettre de faire une liste des roles

➔ On peut également voir qu'il y a une requête 'supp' qui va permettre la suppression dans la table.

Ensuite on crée une nouvelle vue liste_roles.html.twig qui contient ceci :

```
{% extends 'base.html.twig' %}

{% block title %}{{parent()}}Liste roles{% endblock %}

{% block body %}
{{parent()}}
<div class="container-fluid">
    <div class="row justify-content-center">
        <h1 class="text-center text-primary p-4">Liste des roles</h1>
    </div>
    <div class="row justify-content-center">
        <div class="col-8 p-4 m-0 text-primary">
            <div class="table-responsive">
<table class="table table-hover">
 <thead>
 <tr>
  <th scope="col">Libellé</th>
  <th></th>
 <th></th>
 </tr>
 </thead>
 <tbody>
        {% for role in roles %}
        <tr class="{{ cycle(['table-primary', 'table-
secondary'], loop.index0) }}">
        <td>{{role.libelle |capitalize }}</td>
```

```
        <td><a href="{{path('modif_role',{'id':
role.id})}}" class="text-white"><span class="material-
icons" title="Modifier un role">create</span></a></td>
 <td><a href="{{path('liste_roles',{'supp':role.id})}}" class="text-
white"><span
class="material-icons" title="Supprimer le role">
delete
 </span></a></td>


        </tr>
        {% endfor %}
 </tbody>
            </table>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

- ➔ Cela va permettre d'afficher une liste
- ➔ On peut voir qu'il y a un span pour supprimer le rôle mais également un span pour le modifier qui va nous permettre d'acceder à une page de modification mais nous le verrons par la suite.

On obtient donc ceci :

Nous passons donc à la modification :

Pour la modification j'ai choisi de créer un nouveau controller et un nouveau formulaire,

On retape donc les même commandes vues au dessus.

Pour le formulaire on change « ajouter » par modifier

Et pour le controller on met le code suivant :

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
```

```php
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;
use App\Entity\Role;
use App\Form\ModifRoleType;

class ModifRoleController extends AbstractController
{
    /**
     * @Route("/modif_role/{id}", name="modif_role", requirements={"id"="\d+"}
)
     */
    public function modifRole(int $id, Request $request)
    {
        $em = $this->getDoctrine();
        $repoRole = $em->getRepository(Role::class);
        $role = $repoRole->find($id);
        if($role==null){
            $this->addFlash('notice', "Ce role n'existe pas");
            return $this->redirectToRoute('liste_roles');
        }
        $form = $this->createForm(ModifRoleType::class,$role);
        if ($request->isMethod('POST')) {
            $form->handleRequest($request);
            if ($form->isSubmitted() && $form->isValid()) {
                $em = $this->getDoctrine()->getManager();
                $em->persist($role);
                $em->flush();
                $this->addFlash('notice', 'Role modifié');
            }
            return $this->redirectToRoute('liste_roles');
        }
        return $this->render('role/modif_role.html.twig', [
        'form'=>$form->createView()
        ]);
    }
}
```

Code contenant une requête permettant de modifier une ligne de la table rôle.

Ensuite, on crée une vue modif_role.html.twig dans laquelle on met :

```twig
{% extends 'base.html.twig' %}

{% block title %}{{parent()}}Modif Role{% endblock %}
{% block body %}
{{parent()}}
<div class="container-fluid">
 <div class="row justify-content-center">
```

```
 <h1 class="text-center text-primary p-4">Modifier un role</h1>
 </div>
 <div class="row justify-content-center">
 <div class="col-8 bg-white p-4 m-0 text-primary">
 {{form(form)}}
 </div>
 </div>
</div>
{% endblock %}
```

On obtient donc un formulaire de modification.

Ensuite on reproduit la même chose pour toutes les autres tables.