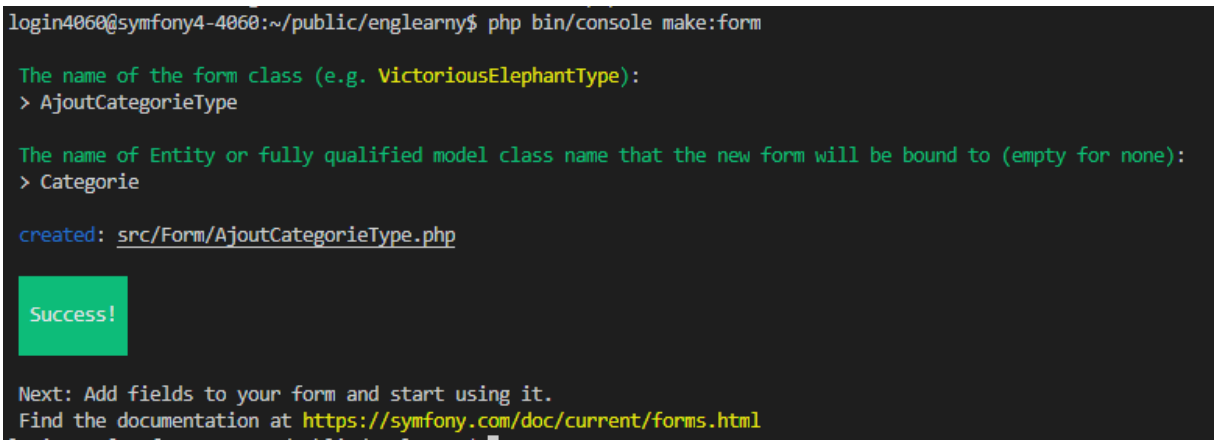
Commandes et résultats

Une fois la base de données créée, on va faire les formulaires, pour cela on utilise la commande suivante :

php bin/console make:form

Puis on insère le nom de notre formulaire et l'entité avec laquelle le formulaire est relié.

Ce qui donne :



On a le success, le formulaire a bien été créé sous forme de Template que nous allons modifier. Pour cela, dans le buildForm, nous allons construire le formulaire pour l'ajout d'une catégorie.

Cependant, il faut d'abord importer les types avec un « use »

Puis on ajoute les éléments que l'on souhaite ainsi que leur type dans le buildForm.

Ce qui donne :

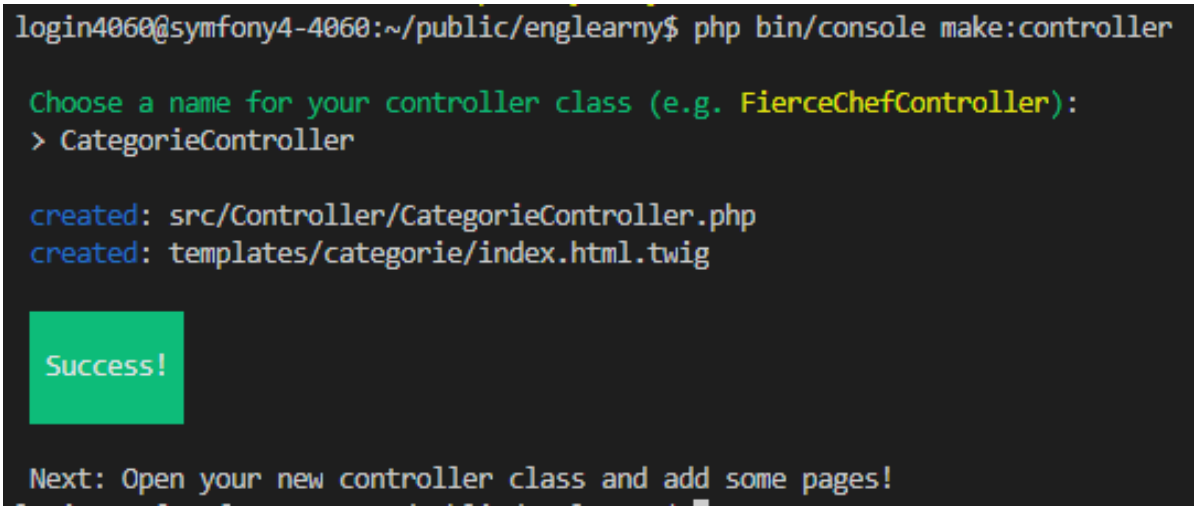
```php
<?php

namespace App\Form;

use App\Entity\Categorie;
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 AjoutCategorieType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('libelle',TextType::class)
            ->add('ajouter',SubmitType::class);
        ;
    }
```

Une fois cela réalisé, nous allons créer le contrôleur grâce à la commande suivante :

php bin/console make:controller

Puis on insère le nom de notre contrôleur.

Ce qui donne :



Ensuite, le contrôleur est construit sous forme de Template que nous allons aussi modifier.

Nous allons donc importer ce qui sera nécessaire pour le contrôleur avec « use »

On change le nom de la route et on ajoute un paramètre de type Request et on ajoute le code permettant l'ajout de la catégorie.

Ce qui donne :

```php
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;
use App\Entity\Categorie;
use App\Form\AjoutCategorieType;


class CategorieController extends AbstractController
{
    /**
     * @Route("/ajoutCategorie", name="ajoutCategorie")
     */
    public function ajoutCategorie(Request $request): Response
    {
        $categorie = new Categorie();
        $form = $this->createForm(ajoutCategorieType::class,$categorie);

        if($request->isMethod('POST')){
            $form->handleRequest($request);
            if($form->isSubmitted() && $form->isValid()){
                $em = $this->getDoctrine()->getManager();
                $em->persist($theme);
                $em->flush();

                $this->addFlash('notice','Catégorie Ajoutée');
            }
            return $this->redirectToRoute('ajout_categorie');
        }

        return $this->render('categorie/ajoutCategorie.html.twig', [
            'form'=>$form->createView()
        ]);
    }
}
```

Une fois cela fait, on va créer une nouvelle vue avec TWIG dans laquelle nous allons mettre ceci :

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

{% block title %}{{parent()}}Ajout catégorie{% endblock %}

{% block contenu %}
{{parent()}}
<div class="container-fluid">
    <div class="row justify-content-center">
        <h1 class="text-center text-secondary p-4">Ajout d'une catégorie</h1>
    </div>
    <div class="row justify-content-center">
        <div class="col-8 bg-secondary p-4 m-0 text-white">
            {{form(form)}}
        </div>
    </div>
</div>
{% endblock %}
```

On modifie aussi le fichier base.html.twig :

On ajoute l'affichage de l'alerte :

```
<div class="container">
    <div class="row">
        <div class="col-12">
            {% for message in app.flashes('notice') %}
            <br/>
            <div class="flash-notice alert alert-secondary text-center">
                {{ message }}
            </div>
            {% endfor %}
        </div>
    </div>
</div>
```

Et on ajoute le chemin :

```
<a class="dropdown-item" href="{{path('ajoutCategorie')}}">Ajout Catégorie</a>
```

Maintenant que cela est fait, on peut ajouter des catégories , maintenant affichons les.

Dans le contrôleur que nous venons de créer, nous allons ajouter ce qui permet d'obtenir la liste

```php
/**
 * @Route("/listeCategorie", name="listeCategorie")
 */
public function listeCategorie(Request $request): Response
{
    $em = $this->getDoctrine();
    $repoCategorie = $em->getRepository(Categorie::class);

    $categories = $repoCategorie->findBy(array(),array('libelle'=>'ASC'));
    return $this->render('categorie/listeCategorie.html.twig', [
        'categories'=>$categories
    ]);

}
```

Puis nous créons une vue :

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

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

{% block contenu %}
{{parent()}}
<div class="container-fluid">
    <div class="row justify-content-center">
        <h1 class="text-center text-secondary p-4">Liste des catégories</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>
                        </tr>
                    </thead>
                    <tbody>
                        {% for categorie in categories %}
                            <tr class="{{ cycle(['table-primary', 'table-secondary'], loop.index0) }}">
                                <td>{{categorie.libelle |capitalize }}</td>
                            </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

Sans oublier la route qui permet d'accéder à la page :

```twig
<a class="dropdown-item" href="{{path('listeCategorie')}}">Liste Catégorie</a>
```

Maintenant, nous allons modifier, pour cela nous recréons un formulaire de la même manière que celui pour l'ajout :

```
login4060@symfony4-4060:~/public/englearny$ php bin/console make:form

 The name of the form class (e.g. GentlePuppyType):
 > ModifCategorieType

 The name of Entity or fully qualified model class name that the new form will be bound to (empty for none):
 > Categorie

 created: src/Form/ModifCategorieType.php

  Success!

 Next: Add fields to your form and start using it.
 Find the documentation at https://symfony.com/doc/current/forms.html
```

De même nous importons ce dont nous avons besoin et modifions le buildForm :

```php
<?php

namespace App\Form;

use App\Entity\Categorie;
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 ModifCategorieType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('libelle',TextType::class)
            ->add('modifier',SubmitType::class);
    }

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

Puis la vue :

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

{% block title %}{{parent()}}Modif catégorie{% endblock %}

{% block contenu %}
{{parent()}}
<div class="container-fluid">
    <div class="row justify-content-center">
        <h1 class="text-center text-secondary p-4">Modifier une catégorie</h1>
    </div>
    <div class="row justify-content-center">
        <div class="col-8 bg-secondary p-4 m-0 text-white">
            {{form(form)}}
        </div>
    </div>
</div>
{% endblock %}
```

Il faut aussi ajouter une route pour y accéder, donc dans listeCategorie.html.twig :

```twig
<thead>
    <tr>
        <th scope="col">Libellé</th>
        <th></th>
    </tr>
</thead>
<tbody>
    {% for categorie in categories %}
        <tr class="{{ cycle(['table-primary', 'table-secondary'], loop.index0) }}">
            <td>{{categorie.libelle |capitalize }}</td>
            <td><a href="{{path('modifCategorie',{'id': categorie.id})}}" class="text-white"><span class="material-icons" title="Modifier une categorie">create</span></a></td>
        </tr>
    {% endfor %}
</tbody>
```

Enfin la suppression :

Pour cela, nous modifions le contrôleur liste :

```php
/**
 * @Route("/listeCategorie", name="listeCategorie")
 */
public function listeCategorie(Request $request): Response
{
    $em = $this->getDoctrine();
    $repoCategorie = $em->getRepository(Categorie::class);

    if ($request->get('supp')!=null){
        $categorie = $repoCategorie->find($request->get('supp'));
        if($categorie!=null){
        $em->getManager()->remove($categorie);
        $em->getManager()->flush();
        }
        $this->addFlash('notice', 'Catégorie supprimée');
        return $this->redirectToRoute('listeCategorie');
    }


    $categories = $repoCategorie->findBy(array(),array('libelle'=>'ASC'));
    return $this->render('categorie/listeCategorie.html.twig', [
        'categories'=>$categories
    ]);

}
```

et on lui donne une route afin de l'activer :

```
<thead>
    <tr>
        <th scope="col">Libellé</th>
        <th></th>
        <th></th>
    </tr>
</thead>
<tbody>
    {% for categorie in categories %}
        <tr class="{{ cycle(['table-primary', 'table-secondary'], loop.index0) }}">
            <td>{{categorie.libelle |capitalize }}</td>
            <td><a href="{{path('modifCategorie',{'id': categorie.id})}}" class="text-white"><span class="material-icons" title="Modifier une categorie">create</span></a></td>
            <td><a href="{{path('listeCategorie',{'supp':categorie.id})}}" class="text-white"><span class="material-icons" title="Supprimer une categorie"> delete </span></a></
        </tr>
    {% endfor %}
</tbody>
```

S'il existe un lien avec une autre table, comme par exemple Test, alors il faut faire comme ceci :

```php
<?php

namespace App\Form;

use App\Entity\Test;
use App\Entity\Theme;
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;
use Symfony\Component\Form\Extension\Core\Type\IntegerType;
use Symfony\Bridge\Doctrine\Form\Type\EntityType;

class AjoutTestType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('libelle',TextType::class)
            ->add('niveau',IntegerType::class)
            ->add('theme',EntityType::class, array("class"=>"App\Entity\Theme","choice_label"=>"libelle"))
            ->add('ajouter',SubmitType::class);
        ;
    }

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

lors de la création des deux formulaires (ajoutTest et modifTest) il faudra importer l'entité qui dépend de celle que nous avons créé ainsi que l'EntityType.

Puis il faudra donner l'entité que l'on souhaite récupérer ainsi que l'attribut que nous souhaitons dans le buildForm.

Enfin, pour l'afficher :

```
<td>{{test.theme.libelle |capitalize }}</td>
```

On passe par thème à partir de test pour récupérer son nom.