Créer un formulaire

Les types des formulaires

Avec les formulaires `Symfony` tout est a un type qui est une classe `PHP`.

Par exemple, un champ texte a le type `TextType`, un champ mot de passe a le type `PasswordType`, un bouton radio a le type `RadioType` etc.

Un groupe de champs peut avoir également un type, comme par exemple `RepeatedType` pour un champ mot de passe et un champ de validation de mot de passe.

Les boutons ont des types : par exemple `ResetType` pour un bouton de réinitialisation, `SubmitType` pour le bouton d'envoi etc.

Nous verrons tous les types au fur et à mesure de nos besoins, vers la liste de tous les types n'a pas d'intérêt à ce stade. Retenez simplement, que **tout a un type dans un formulaire `Symfony`**.

Ces types permettent d'accéder à des fonctionnalités grâce à l'utilisation de classes spécifiques pour l'ensemble des éléments d'un formulaire.

Créer et afficher un formulaire

Nous avons vu que la première étape était de créer le formulaire, avant de pouvoir l'afficher avec `Twig`.

Il faut commencer par créer un objet à partir de la classe `FormBuilderInterface`.

La méthode `createFormBuilder()`

Pour ce faire, `Symfony` met à disposition la méthode **`createFormBuilder()`**.

Après avoir installé le composant `SymfonyForm`, cette méthode est disponible dans tous les contrôleurs qui étendent la classe `AbstractController`.

Modifions le fichier `src/Controller/DefaultController.php` pour créer l'objet `FormBuilderInterface` :

```
<?php

namespace App\Controller;
```

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController extends AbstractController
{
  function __construct()
  {
  }

  #[Route('/', name: 'index')]
  public function index()
  {
    $form = $this->createFormBuilder();
  }
}
```

Pour l'instant l'objet retourné est un `FormBuilderInterface` mais il n'a aucun champ, ni aucun bouton.

Symfony propose un ensemble de méthode pour créer le formulaire que vous avez besoin à partir d'un objet `FormBuilderInterface`.

La méthode `add()`

La méthode `add()` permet d'ajouter un élément au formulaire comme par exemple un champ ou un bouton.

Le premier argument est le **nom unique de l'élément**. `Symfony` l'utilisera pour l'attribut `id`.

Le deuxième argument est **le type de l'élément à créer**. Il faut lui passer une classe. Cela permet de préconfigurer de nombreuses choses pour l'élément. Nous l'expliquerons plus en détail dans la leçon suivante.

Le troisième argument est **un tableau associatif d'options de configuration pour l'élément**. Par exemple, pour rendre un champ optionnel etc.

Par exemple :

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
```

```
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController extends AbstractController
{
  function __construct()
  {
  }

  #[Route('/', name: 'index')]
  public function index()
  {

    $form = $this->createFormBuilder()
      ->add('content', TextType::class)
      ->add('submit', SubmitType::class)
  }
}
```

Ici nous créons un champ `content` qui a le type `TextType`.

Nous ajoutons un élément de type `SubmitType` pour le bouton d'envoi.

La méthode `getForm()`

Il faut ensuite créer l'objet formulaire avec la méthode `getForm()` qui va retourner un objet `FormInterface` :

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController extends AbstractController
{
  function __construct()
```

```
  {
  }

  #[Route('/', name: 'index')]
  public function index()
  {

    $form = $this->createFormBuilder()
      ->add('prenom', TextType::class)
      ->add('submit', SubmitType::class)
      ->getForm();
  }
}
```

Nous avons maintenant notre objet formulaire qui est prêt à être affiché.

La méthode `createView()`

Il faut ensuite créer la vue pour le formulaire à partir de l'objet `FormInterface`. Pour cela, il faut utiliser la méthode **`createView()`** :

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController extends AbstractController
{
  function __construct()
  {
  }

  #[Route('/', name: 'index')]
  public function index()
  {

    $form = $this->createFormBuilder()
```

```
        ->add('content', TextType::class)
        ->add('submit', SubmitType::class)
        ->getForm();

      return $this->render('page1.html.twig', [
        'myform' => $form->createView()
      ]);
    }
  }
```

Une fois que nous avons la vue de notre formulaire : un objet `FormView` retourné par la méthode `createView()`, nous la passons au `template Twig` pour pouvoir l'afficher.

Pour le `template`, dans le fichier `templates/page1.html.twig`, nous mettons simplement :

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

{% block body %}
  {{ form(myform) }}
{% endblock %}
```

La fonction `Twig form()` permet simplement d'afficher le formulaire `HTML` en utilisant la vue `FormView`.

Note sur le mot clé `class`

`::class` permet de résoudre le nom d'une classe.

Ce n'est pas du tout propre à `Symfony`, c'est une fonctionnalité du langage `PHP`.

En faisant, `NomDeClasse::class` nous récupérons le nom complet de la classe avec le `namespace` utilisé.