Créer une classe pour la construction de formulaire

Créer une classe pour votre formulaire

`Symfony` recommande de créer une classe séparée pour construire les formulaires.

En effet, comme vous avez pu le constater cela fait beaucoup de code dans le contrôleur et cela le rend moins lisible et maintenable.

Pour générer une classe de formulaire faites :

```
symfony console make:form
```

Saisissez ensuite `Todo`.

Le `Maker` a automatiquement créé le fichier `src/Form/TodoType.php` :

```
<?php

namespace App\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;

class TodoType extends AbstractType
{
  public function buildForm(FormBuilderInterface $builder, array $opt
ions): void
  {
    $builder
      ->add('field_name')
    ;
  }

  public function configureOptions(OptionsResolver $resolver): void
  {
    $resolver->setDefaults([
      // Configure your form options here
    ]);
```

```
  }
}
```

`buildForm()` est la méthode qui va construire le formulaire. C'est ici que vous devez placer toute la logique de création du formulaire.

`configureOptions()` est la méthode pour les options de configuration du constructeur de formulaire.

Nous en reparlerons lorsque nous aurons vu `Doctrine` mais il est conseillé d'indiquer la classe contenant l'entité (appelée `data_class`) qui correspond à l'entité du formulaire dans la méthode `configureOptions()`.

Exemple de la vidéo

Voici le fichier `src/Form/TodoType.php` :

```php
<?php

namespace App\Form;

use App\Controller\Todo;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\Validator\Constraints\Length;
use Symfony\Component\Validator\Constraints\NotBlank;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type\CountryType;

class TodoType extends AbstractType
{
  public function buildForm(FormBuilderInterface $builder, array $opt
ions): void
  {
    $builder
      ->add('content', TextType::class, [
        'label' => 'Un super label',
        'attr' => [
          'placeholder' => 'Contenu de la todo'
        ],
        'help' => 'Indiquez ce que vous avez à faire',
        'constraints' => [
          new NotBlank(message: 'Le contenu ne doit pas être vide'),
```

```
            new Length([
              'min' => 10,
              'max' => 50,
              'minMessage' => 'Le contenu doit faire au moins {{ limit
}} caractères',
              'maxMessage' => 'Le contenu doit faire au plus {{ limit
}} caractères',
            ])
          ],
        ])
        ->add('country', CountryType::class);
    }

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

Nous avons simplement déplacé toute la logique depuis le contrôleur.

Voici le fichier `src/Controller/DefaultController.php` :

```
<?php

namespace App\Controller;

use App\Form\TodoType;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController extends AbstractController
{
    function __construct()
    {
    }

    #[Route('/', name: 'index')]
    public function index(Request $request)
    {
      $todo = new Todo();
```

```
        $form = $this->createForm(TodoType::class, $todo);

        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
        }

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

class Todo
{
    function  __construct(
      public string $content = '',
      public ?string $country = null,
    ) {
    }
}
```

Notez que nous utilisons la méthode `createForm()` et plus `createFormBuilder()` car le constructeur du formulaire (le `Builder`) est dans la classe spécifique que nous avons créée.