Envoi d'un formulaire

Action et méthode

Comme vous le savez si vous avez fait la formation `HTML` ou `JavaScript`, un formulaire `HTML` a une méthode et optionnellement une action.

La méthode permet de définir la méthode HTTP qui sera utilisée pour envoyer les données au serveur. Par défaut, `Symfony` définira la méthode comme étant `POST`.

L'action permet de définir l'`URL` sur laquelle sera envoyée le formulaire. Par défaut, si aucune action n'est précisée, le formulaire sera envoyé sur l'`URL` de la page où il se trouve.

`Symfony` met à disposition les méthodes `setAction()` et `setMethod()` pour définir l'action et la méthode d'un formulaire.

En reprenant notre exemple, nous pouvons donc faire :

```
<?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)
      ->setAction('/test')
      ->setMethod('get')
```

```
      ->getForm();

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

Bien sûr nous n'avons rien défini pour la route `/test` et il y aura donc une erreur lors de l'envoi.

A noter que `Symfony` recommande officiellement d'utiliser la même route et le même contrôleur pour afficher le formulaire et l'envoyer.

Soumettre un formulaire

Nous savons donc maintenant que par défaut, en appuyant sur le bouton d'envoi du formulaire, il sera envoyé avec la méthode `POST` sur la route où se trouve le formulaire.

Dans notre exemple, c'est la route `/` .

Pour gérer l'envoi du formulaire il faut le code suivant :

```
<?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\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController extends AbstractController
{
  function __construct()
  {
  }

  #[Route('/', name: 'index')]
  public function index(Request $request)
  {
```

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

      $form->handleRequest($request);

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

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

`handleRequest($request)` permet de vérifier si le formulaire est envoyé ou affiché. Si le formulaire est envoyé, la méthode va récupérer les données du formulaire automatiquement, le valider, enregistrer les éventuelles erreurs pour l'affichage etc. C'est une méthode qui fait beaucoup de chose pour vous, mais uniquement dans le cas où le formulaire est `submit`.

Cette méthode doit être appelée avant `$form->isSubmitted()` et avant `$form->createView()`.

`$form->isSubmitted()` retourne `true` uniquement si le formulaire est `submit`, c'est-à-dire envoyé.

`$form->isValid()` retourne `true` uniquement si le formulaire est valide, c'est-à-dire si toutes les validations effectuées ont réussies. Nous verrons en détail la validation plus tard.

`$form->getData()` permet de récupérer les données du formulaire dans un tableau associatif. Ici nous l'utilisons simplement pour afficher les données.

Il faut donc distinguer 3 cas :

1 - Le formulaire est affiché et `handleRequest()` ne fait rien, `isSubmitted()` retourne `false` et nous passons simplement au rendu avec `$this->render()`.

2 - Le formulaire est envoyé mais il n'est pas valide. `$form->isValid()` retourne `false` et dans ce cas `handleRequest()` a placé les erreurs sur un objet qui sera utilisé pour afficher

les erreurs dans la vue. C'est pour cette raison que `createView()` doit absolument être utilisée après `handleRequest()`.

3 - Le formulaire est envoyé et il est valide. Dans ce cas nous allons dans le `if` et nous gérons les données. Le plus souvent pour les enregistrer dans une base de données puis nous redirigeons l'utilisateur sur une autre page.