

## Workshop : Validation d'un formulaire côté serveur

### Objectifs

- Le but de ce workshop est de valider un formulaire côté serveur.

**NB** : Dans ce workshop, nous allons travailler avec une seule entité « **Student** » comme le montre la figure suivante :

<b>Student</b>
<u>Nsc</u>
Email

Avec **Nsc** : le numéro d'inscription

### Partie1 : Personnalisation de l'affichage du formulaire

- Pour créer un formulaire, nous pouvons le faire par deux méthodes :
  - ☐ Soit directement dans le contrôleur
  - ☐ Soit dans une classe externe (FormType)
- Pour ce faire, lancez la commande suivante dans le terminal :

```
php bin/console make:form
```

- Cette commande vous demandera si votre formulaire est associé à une entité ou non.
- Dans notre exemple, nous allons créer un formulaire pour l'entité « **Student** ». Le suffixe de la classe est le mot **Type** :

```
C:\>php bin/console make:form

The name of the form class (e.g. GentleElephantType):
> Student

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

created: src/Form/StudentType.php

Success!

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

Comme l'indique le message dans le terminal, le formulaire a été créé dans notre projet sous le dossier **src/form**.

Le contenu de cette classe est similaire à celui montré dans la figure suivante :

```
<?php

namespace App\Form;
use ...

class StudentType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add( child: 'nsc')
            ->add( child: 'email')
        ;
    }

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

Figure 1: Le contenu du fichier "StudentType.php"

- Pour afficher le formulaire dans un Template Twig, nous devons procéder par les deux étapes suivantes :

- Pour visualiser le formulaire dans le Template Twig, il existe deux méthodes :
- ❑ Afficher directement la totalité du formulaire grâce à la fonction Twig « **form()** » :

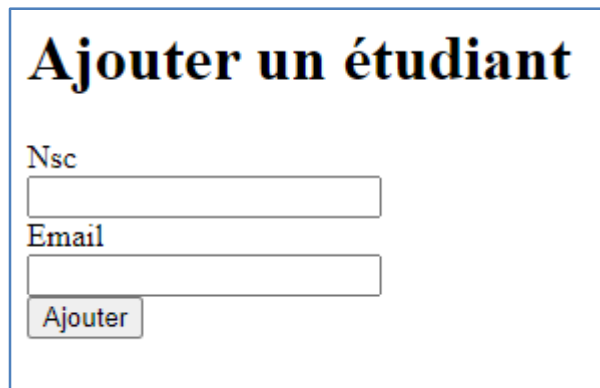
```
{{ form(nomDuFormulaire) }}
```

Comme le montre l'exemple de la figure 2 suivante:

```
<h1>Ajouter un étudiant</h1>
{{ form(form) }}
```

**Figure 2: Le contenu du fichier « add.html.twig »**

**NB :** Cette fonction exploite un thème par défaut pour afficher le formulaire (Voir figure 3).

The image shows a web form with a title "Ajouter un étudiant" in a large, bold, serif font. Below the title, there are two input fields. The first is labeled "Nsc" and the second is labeled "Email". Both labels are in a smaller, bold, serif font. Below the "Email" field, there is a button labeled "Ajouter" in a bold, serif font. The entire form is enclosed in a blue border.

**Figure 3: Le rendu de la page d'ajout d'un formulaire**

- ❑ Personnaliser l'affichage du formulaire (changement de l'ordre des champs, changement de la valeur du label des certains champs, etc) en utilisant les fonctions prédéfinies du moteur de template Twig:

★ **form\_start () :**

⇒ C'est l'équivalent de la balise <form> en HTML

⇒ Accepte deux paramètres :

1. Le premier est la variable formulaire (passée depuis le contrôleur).
2. Le deuxième, facultatif, contient l'index « **attr** » des paramètres

(par exemple nom de la classe bootstrap).

**Exemple:**

```
{{ form_start(form, {'attr':{'class':'form-horizontal'}}) }}
```

★ **form\_errors() :**

⇒ Elle affiche les erreurs relatives au champ en question (celui passé en argument)

★ **form\_label() :**

⇒ affiche le label HTML d'un élément du formulaire.

⇒ accepte deux paramètres :

1. Le premier est le champ concerné par le label.
2. Le deuxième, facultatif, est le contenu du label (par défaut, le nom de l'attribut ou la valeur « label » dans le form type).

⇒ **Exemple :**

```
{{ form_label(form.nsc, "Numéro d'inscription") }}
```

❖ **form\_widget() :**

⇒ Affiche le champ du formulaire lui-même (soit <input>, soit <select>...)

❖ **form\_row() :**

⇒ form\_label + form\_errors + form\_widget

❖ **form\_rest() :**

⇒ Affiche le reste du formulaire (les champs que nous n'avons pas précisé)

❖ **form\_end() :**

C'est l'équivalent de la balise fermante </form> en HTML

Ci-après, le code mis à jour en utilisant les méthodes mentionnées ci-dessus :

```

{{ form_start(form) }}
<h1> Ajouter un étudiant </h1>
{{ form_errors(form.nsc) }}
{{ form_label(form.nsc, "Numéro d'inscription") }}
{{ form_widget(form.nsc) }}
{{ form_errors(form.email) }}
{{ form_label(form.email, "Email") }}
{{ form_widget(form.email) }}
{{ form_end(form) }}

```

Figure 4: Le contenu du fichier « add.html.twig » mis à jour

**Ajouter un étudiant**

Numéro d'inscription

Email

Ajouter

Twig code for 'Numéro d'inscription':  
`{{ form_label(form.nsc, "Numéro d'inscription") }}`

Twig code for 'Email':  
`{{ form_widget(form.email) }}`

Figure 5: Le rendu personnalisé de la page d'ajout d'un formulaire

## Partie 2 : Validation des formulaires

### 1) Principe

- Symfony permet l'ajout des contrôles de saisie dans les formulaires.
- La validation des objets (entités ou autres) se réalise avec le composant **Validator** de Symfony.
- Ce composant est dédié pour valider les objets selon des contraintes sur le formulaire.
- Pour assurer le fait que les données saisies sont valides, il faut ajouter la fonction **isValid ()**.

```

/**
 * @Route("/add", name="addStudent")
 */
public function addStudent(Request $request)
{
    $student = new Student();
    $form = $this->createForm( type: StudentType::class, $student);
    $form->add( child: 'ajouter', type: SubmitType::class);
    $form->handleRequest($request);
    if ($form->isSubmitted() && $form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $em->persist($student);
        $em->flush();
        return $this->redirectToRoute( route: 'student');
    }
    return $this->render( view: "student/add.html.twig", ['form'=>$form->createView()]);
}

```

## 2) Syntaxe

- **Utilisation de namespace:**

Afin de pouvoir utiliser les annotations de validation il faut importer la class Constraints et l'ajouter dans l'entité « **Student** ».

```
use Symfony\Component\Validator\Constraints as Assert;
```

- **La forme simple :**

```
@Assert\Contrainte (valeur de l'option par défaut)
```

**Avec:**

- ❑ La **Contrainte**, est utilisée pour affirmer une condition appliquée sur la valeur d'une propriété. Peut-être, par exemple, **MinLength**, **NotBlank**, **IsNull**, etc.
- ❑ La **Valeur** entre parenthèses, qui est la valeur de l'option par défaut. En effet chaque contrainte a plusieurs options, dont une par défaut. Par exemple, l'option par défaut de **MinLength** est certainement la valeur de la longueur minimale que l'on veut appliquer.
- ❑ Cette annotation est ajoutée au niveau de l'attribut.

- **La forme étendue :**

Nous pouvons également utiliser la forme étendue, qui permet de personnaliser la valeur de plusieurs options en même temps, comme indiqué ci-dessous:

```
@Assert\Contrainte (option1="valeur1", ... optionN="valeurN")
```

Les différentes options varient d'une contrainte à l'autre.

Ci-après, un exemple avec la contrainte **MinLength** :

- **Exemple :**

```
@Assert\MinLength(limit=10,message= "Votre mot de passe ne contient pas {{ limit }} caractères."))
```

### 3) Les contraintes

Parmi les contraintes, on peut citer :

contrainte	Définition
Blank/ NotBlank	Vérifie si la valeur d'un champ est une chaîne vide ou null (NotBlank est l'inverse)
IsTrue	Vérifie si la valeur d'un champ est true, 1 ou '1' (pareillement pour IsFalse)
Length	Vérifie si la longueur d'un champ se situe entre une valeur minimale et une valeur maximale. Elle accepte plusieurs options : <ul style="list-style-type: none"> <li>▪ <b>min</b></li> <li>▪ <b>minMessage</b>: le message d'erreur à afficher si la contrainte min n'est pas respectée.</li> <li>▪ <b>max</b></li> <li>▪ <b>maxMessage</b>: le message d'erreur à afficher si la contrainte max n'est pas respectée.</li> </ul>
url	Vérifie si la valeur est une adresse URL valide

UserPassword	Vérifie que la valeur saisie dans l'input égale à celle de l'utilisateur connecté.
Date / DateTime	Vérifie que la valeur est un objet de type Datetime, ou une chaîne de caractères du type YYYY-MM-DD / YYYY-MM-DD HH:MM:SS.

- Nous allons faire le contrôle de saisie de notre formulaire (Voir figure 5) qui doit respecter les contraintes suivantes :
  - Le numéro d'inscription doit être un champ obligatoire. Nous allons utiliser la contrainte (**\NotBlank**),
  - L'Email soit un champ obligatoire et de type Email alors nous allons utiliser deux contraintes (**\NotBlank**, **\Email**).
- Nous allons ajouter les annotations dans l'entité Student comme indiqué dans la figure suivante :

```

use Symfony\Component\Validator\Constraints as Assert;

/**
 * @ORM\Entity(repositoryClass=StudentRepository::class)
 */
class Student
{
    /**
     * @ORM\Id
     * @ORM\Column(type="string", length=255)
     * @Assert\NotBlank(message="NSC is required")
     */
    private $nsc;

    /**
     * @ORM\Column(type="string", length=255)
     * @Assert\NotBlank(message="Email is required")
     * @Assert>Email(message = "The email '{{ value }}' is not a valid
email.")
     */
    private $email;

```

**Figure 6: Le contenu de l'entité « Student.php »**



Pour afficher les messages d'erreur, il faut désactiver le contrôle de saisie HTML5.

Pour ce faire, nous ajouterons cette ligne de code dans le fichier Twig:

```
{{ form start(form, {'attr': {'novalidate': 'novalidate'}} ) }}
```



**Ajouter un étudiant**

Numéro d'inscription

- NSC is required

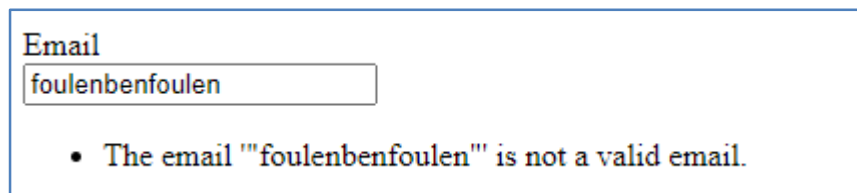
Email

- Email is required

Ajouter

Figure 7: Le rendu de la page d'ajout d'un formulaire

En cliquant sur le bouton « **Ajouter** », le message d'erreur suivant s'affiche si l'adresse mail saisie par l'utilisateur est non valide.



Email

- The email "foulenbenfoulen" is not a valid email.

Figure 8: Email saisi par l'utilisateur est non valide

#### Référence:

- <https://class.nerdy-bear.com/symfony/form-type>
- <https://symfony.com/doc/4.4/validation.html#constraints>
- [https://symfony.com/doc/4.4/form/form\\_customization.html](https://symfony.com/doc/4.4/form/form_customization.html)
- <https://symfony.com/doc/4.4/reference/constraints.html>