|  |  |
| --- | --- |
|  | Année universitaire : 2024-2025  **Workshops Framework Symfony6**  **UP-Web** |
| **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 :

|  |
| --- |
| **Student** |
| Nsc  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 :

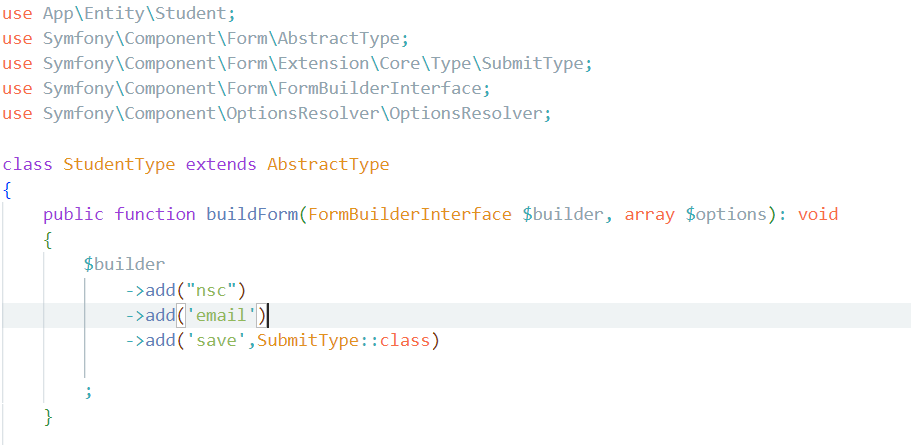
php bin/console make:form

* 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 :**



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 :



**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() »** :

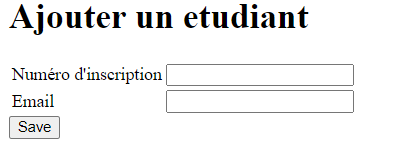
|  |
| --- |
| {{ form(nomDuFormulaire) }} |

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



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

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



**Figure 3: Le rendu du 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

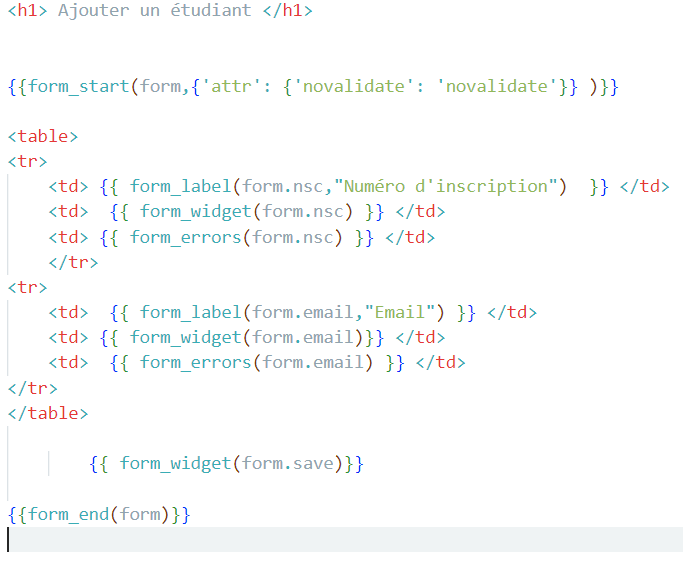
* ***form\_rest() :***

⇨ Affiche le reste du formulaire (les champs que nous n’avons pas précisé)

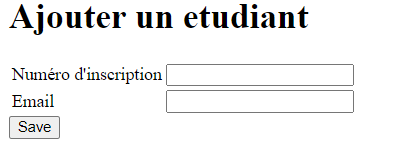
* ***form\_end():***

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 :



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



{{ form\_widget(form.email) }}

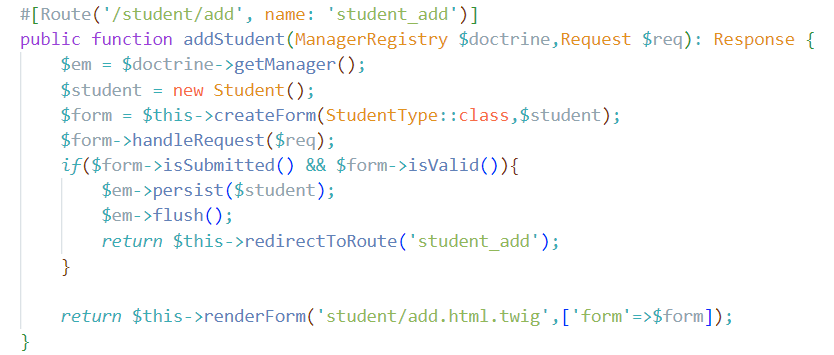
{{ form\_label(form.nsc,"Numéro d'inscription") }}

**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 ()**.



**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.
* **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 **Length** :

* **Exemple :**

|  |
| --- |
| #[Assert\Length(min:10,message:"Votre mot de passe ne contient pas {{ limit }} caractères."))] |

**3) Les contraintes**

Parmi les contraintes, on peut citer :

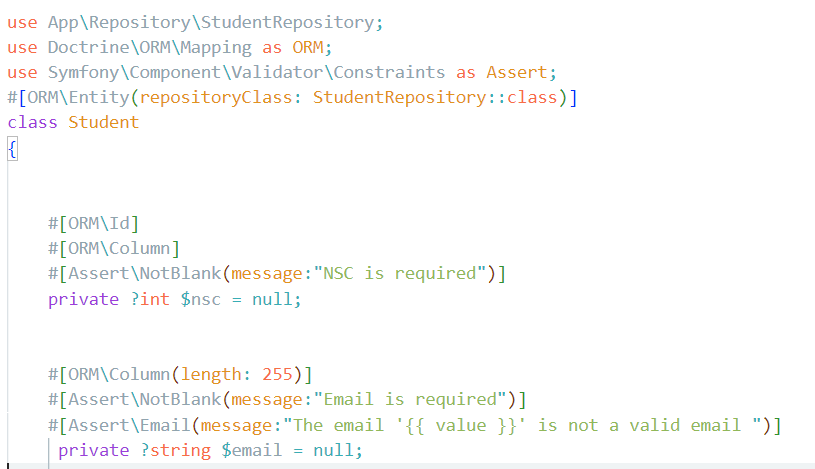
|  |  |
| --- | --- |
| **contrainte** | **Définition** |
| Blank/ NotBlank | Vérifie si la valeur d’un champ est une chaîne vide ou null (NotBlank est l’inverse) |
| IsTrue | Vérifie si la valeur d’un champ est true, 1 ou ‘1’ (pareillement pour IsFalse |
| Length | Vérifie si la longueur d’un champ se situe entre une valeur minimale et une valeur maximale.  Elle accepte plusieurs options :   * **min** * **minMessage**: le message d’erreur à afficher si la contrainte min n’est pas respectée. * **max** * **maxMessage**: le message d’erreur à afficher si la contrainte max n’est pas respectée. |
| 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. |
| Positive | Valide qu’une valeur est un nombre positif. |

* 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**).

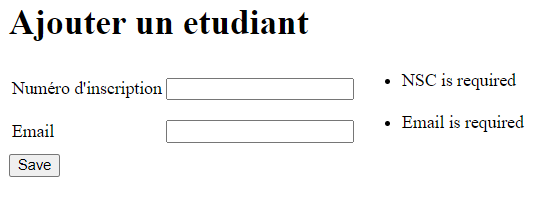
* Nous allons ajouter les annotations dans l'entité Student comme indiqué dans la figure suivante :



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

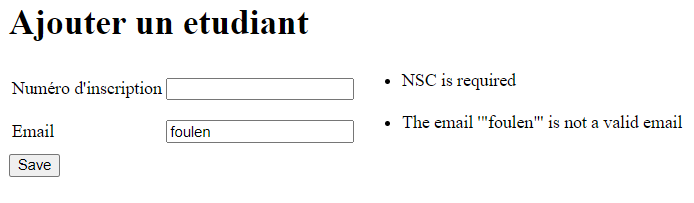
Pour afficher les messages d’erreur, il faut désactiver le contrôle de saisie HTML5. Pour ce faire, nous ajouterons cette ligne de code dans le fichier Twig:

|  |
| --- |
| {{ form\_start(form,{'attr': {'novalidate': 'novalidate'}} ) }} |



**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.



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