

Table des matières

[1. Les formulaires. 3](#_Toc12627194)

[2. Enregistrer les données du formulaire. 5](#_Toc12627195)

[4. Mettre à jour un formulaire. 6](#_Toc12627196)

[5. Le formulaire en automatique 8](#_Toc12627197)

[6. La validation du formulaire.(Contraintes). 9](#_Toc12627198)

[7. Astuces 10](#_Toc12627199)

Tuto 2/4, blog :

# 1. Les formulaires.

1.1 Création d’une page.

Rappel : l’ordre des routes dans le controller ont un impact sur le fonctionnement d’un site.

-Le controller à placer avant la route /blog/{id} pour éviter les conflits.

*/\*  
\* @Route ("/blog/new", name="blog\_create")  
\*/  
public function* create (){  
 *return* $this ->render ('blog/create.html.twig');  
}

-Création de la page blog/create.html.twig.

{% *extends*'base.html.twig' %}  
  
{% *block* body %}  
<h1>Création d'un article !</h1>  
  
{% *endblock* %}

-Mettre à jour la nav barre en modifiant le path  tel que :

<li *class*="nav-item">  
 <a *class*="nav-link" *href*="{{ path ('blog\_create') }} "> Creer un article</a>  
</li>

1.2 Création du formulaire.

Dans le controller :

*/\*  
\* @Route ("/blog/new", name="blog\_create")  
\*/  
public function* create (Request $request){  
 $article = *new* Article();  
  
 *//on cree un formulaire et on passe un Article  
 // le form est lié à l'article.* $form = $this ->createFormBuilder($article);  
 *// add permet d'ajouter un champ au formulaire* -> add('title')  
 -> add('content')  
 -> add('image')  
 *//afficher le résultat final Objet Form très complxe* -> getForm ();  
  
  
 *return* $this ->render ('blog/create.html.twig',[  
 'form'=> $form -> createView()  
 *//createView est le resultat que l'on passe à Twig* ]);  
}

1.3 Affichage du formulaire.

Dans la vue create.html.twig :



Utiliser le template prévu par twig, pour cela ajouter la ligne :

form\_themes: ['bootstrap\_4\_layout.html.twig']

dans package/twig.

Ensuite l’appeler dans la vue concernée :

{% *form\_theme* form 'bootstrap\_4\_layout.html.twig' %}

FormRow permet d’afficher les champs et de choisir leurs options :

{{ form\_row(formArticle.title,{'attr':{'placeholder':"Titre de l'article"}}) }}

Code générale de la page create.html.twig

{% *extends*'base.html.twig' %}  
  
{% *form\_theme* formArticle 'bootstrap\_4\_layout.html.twig' %}  
  
  
{% *block* body %}  
  
 <h1>Création d'un article </h1>  
  
 {{ form\_start(formArticle) }}  
  
 {{ form\_row(formArticle.title,{'attr':{'placeholder':"Titre de l'article"}}) }}  
 {{ form\_row(formArticle.content,{'attr':{'placeholder':"Contenu de l'article"}}) }}  
 {{ form\_row(formArticle.image,{'attr':{'placeholder':"Image"}}) }}  
  
 <button *type*="submit" *class*="btn btn-success">Ajouter l'article</button>  
  
 {{ form\_end(formArticle) }}  
  
{% *endblock* %}

# 2. Enregistrer les données du formulaire.

On analyse la requete, et on vérifier les données.

*//analyse la requete*$form ->handleRequest($request);  
*//dump($article);  
  
//vérifier si le formulaire a été soumis ET le formulaire est valide  
if* ($form -> isSubmitted() && $form->isValid()){  
   
}

On compile l’ensemble :

*/\*\*  
 \** ***@Route****("/blog/create",name="blog\_create")  
 \*/  
public function* creation (Request $request, ObjectManager $manager){  
 $article = *new* Article();  
  
 *//formulaire lié à l'article* $form = $this -> createFormBuilder($article)  
 ->add('title')  
 ->add('content')  
 ->add('image')  
 ->getForm();  
  
 *//analyse la requete* $form ->handleRequest($request);  
 *//dump($article);  
  
 //vérifier si le formulaire a été soumis ET le formulaire est valide  
 if* ($form -> isSubmitted() && $form->isValid()){  
 $article -> setCreatedAt(*new* \DateTime());  
  
 $manager -> persist($article);  
 $manager -> flush();  
  
 *//au deuxième passage on analyse le formulaire si ok  
 //je retourne sur la page de l'article  
 return* $this->redirectToRoute('blog\_show',[  
 'id'=> $article -> getId ()]);  
  
 }  
 *//on affiche le formulaire au premier passage  
 return* $this -> render ('blog/create.html.twig',[  
 'formArticle'=> $form ->createView()]);  
}

# 3. Mettre à jour un formulaire.

Quelques notions :





Grace à la notion «EditMode » on peut choisir le texte en fonction de la situation dans laquelle on

Se trouve.

Dans la vue :

<h1>  
 {% *if* editMode %}  
 Modification de l'article  
 {% *else* %}  
 Création d'un article  
 {% *endif* %}  
</h1>

Dans le controleur :

*return* $this -> render ('blog/create.html.twig',[  
 'formArticle'=> $form ->createView(),  
 *//ici on modifie le bouton en fonction de l'edit ou l'ajout  
 // editt mode ; boolean* 'editMode' => $article -> getId()!==*null*]);

le contrôleur pour edit et new

*/\*\*  
 \** ***@Route****("/blog/create", name="blog\_create")  
 \** ***@Route****("/blog/{id}/edit", name="blog\_edit")  
 \*  
 \*/  
public function* form (Article $article=*null*,Request $request, ObjectManager $manager){  
  
 *if*(!$article){  
 $article = *new* Article();  
 }  
 *//reprise de la meme fonction pour modifier  
  
 //formulaire lié à l'article* $form = $this -> createFormBuilder($article)  
 ->add('title')  
 ->add('content')  
 ->add('image')  
 ->getForm();  
  
 *//analyse la requete* $form ->handleRequest($request);  
 *//dump($article);  
  
 //vérifier si le formulaire a été soumis ET le formulaire est valide  
 if* ($form -> isSubmitted() && $form->isValid()){  
 *//ici on précise que si l'article n'a pas d'identifiant  
 //alors on lui ajoute une date  
 if*(!$article ->getId()){  
 $article -> setCreatedAt(*new* \DateTime());  
 }  
  
 $manager -> persist($article);  
 $manager -> flush();  
  
 *//au deuxième passage on analyse le formulaire si ok  
 //je retourne sur la page de l'article  
 return* $this->redirectToRoute('blog\_show',[  
 'id'=> $article -> getId ()]);  
  
 }  
 *//on affiche le formulaire au premier passage  
 return* $this -> render ('blog/create.html.twig',[  
 'formArticle'=> $form ->createView(),  
 *//ici on modifie le bouton en fonction de l'edit ou l'ajout  
 // editt mode ; boolean* 'editMode' => $article -> getId()!==*null* ]);  
  
}

# 4. Le formulaire en automatique



*/\*\*  
 \** ***@Route****("/blog/create", name="blog\_create")  
 \** ***@Route****("/blog/{id}/edit", name="blog\_edit")  
 \*  
 \*/  
public function* form (Article $article=*null*,Request $request, ObjectManager $manager){  
  
 *if*(!$article){  
 $article = *new* Article();  
 }  
 *//grace au formtype par besoin de définir ici le constructeur  
 //reprise de la meme fonction pour modifier  
 //formulaire lié à l'article  
 //$form = $this -> createFormBuilder($article)  
 // ->add('title')  
 // ->add('content')  
 // ->add('image')  
 // ->getForm();  
  
 //utilisation du formType* $form = $this -> createForm(ArticleType::*class*, $article);  
  
  
 *//analyse la requete* $form ->handleRequest($request);  
 *//dump($article);  
  
 //vérifier si le formulaire a été soumis ET le formulaire est valide  
 if* ($form -> isSubmitted() && $form->isValid()){  
 *//ici on précise que si l'article n'a pas d'identifiant  
 //alors on lui ajoute une date  
 if*(!$article ->getId()){  
 $article -> setCreatedAt(*new* \DateTime());  
 }  
 $manager -> persist($article);  
 $manager -> flush();  
  
 *//au deuxième passage on analyse le formulaire si ok  
 //je retourne sur la page de l'article  
 return* $this->redirectToRoute('blog\_show',[  
 'id'=> $article -> getId ()]);  
  
 }  
 *//on affiche le formulaire au premier passage  
 return* $this -> render ('blog/create.html.twig',[  
 'formArticle'=> $form ->createView(),  
 *//ici on modifie le bouton en fonction de l'edit ou l'ajout  
 // editt mode ; boolean* 'editMode' => $article -> getId()!==*null* ]);  
  
}

# 5. La validation du formulaire.(Contraintes).

Dans le dossier Entity :

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

Ajout d’une contrainte :

*/\*\*  
 \** ***@ORM\Column****(type="string", length=255)  
 \** ***@Assert\Length****(min=10,max="255")  
 \*/  
private* $title;

Vérification pour l’image :



Les assert fonctionnent car le formulaire Symfony est performant et lié aux assert.

A connaitre, le formulaire erreur :

{{ form\_error(formArticle.title) }}

*/\*\*  
 \** ***@ORM\Column****(type="string", length=255)  
 \** ***@Assert\Length****(min=10,max=255,minMessage="Le titre est bien trop court l'ami !") -------------ici-------  
 \*/  
private* $title;  
  
*/\*\*  
 \** ***@ORM\Column****(type="text")  
 \** ***@Assert\Length****(min=10) ------------------------ ici ---------------------------------  
 \*/  
private* $content;  
  
*/\*\*  
 \** ***@ORM\Column****(type="string", length=255)  
 \** ***@Assert\Url****() ------------------------- et la ------------------------------------  
 \*/  
private* $image;

# 6. Astuces

-Pour l’utilisation des annotations en route : composer require annotations

-Pour ajouter un fichier CSS dans la base :

<link rel="stylesheet" href="{{ asset('bootstrap.min.css') }}" >