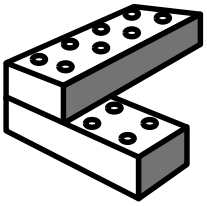

Travaux Pratiques : Enoncés
TP_ReviewPhoto_Partie_06

Objectif : **Consolidation sur le framework Symfony 6**
Ajout des boutons de connexion et de déconnexion

Afin de permettre à l'utilisateur de se connecter, nous allons ajouter dans le « jumbotron » du fichier « base.html.twig » les boutons de connexion et de déconnexion.

Nous afficherons aussi son pseudo une fois connecté.

Réalisations :

1 : pour être cohérent avec notre convention d'appellation de nos routes :

name: 'nom_entity.action' ex #[Route('/', name: 'photo.list')] Entity Photo, action list

Nous allons renommer les routes « app_login » et « app_logout » par « user.login » et « user.logout ».

Ces noms de route sont à modifier dans plusieurs fichiers :

- **Dans le contrôleur « securityController.php »**
- **Dans le fichier « security.yaml »**
- **Dans le fichier « UserAuthenticator.php »**

2 : nous aurons besoin plus tard de réaliser des actions sur l'entité « User » (create, update, delete).

Il existe déjà les 2 actions que nous venons de renommer : login et logout ; mais ces actions se trouvent dans un contrôleur qui se nomme « securityController ».

Nous allons donc le renommer « UserController » :

1. **Renommer le fichier « SecurityController.php » en « UserController.php »**
2. **Renommer la classe à l'intérieur de ce fichier « UserController »**

3 : modification du fichier de template « base.html.twig »

- **Remplacer le code du « jumbotron » par :**

```
<div class="col-6">
   <h1>Commenter des photos</h1>
</div>
<div class="col-2">
   <h5>Date : {{ "now"|date("m/d/Y") }}</h5>
</div>
<div class="col-4">
   {% block connexion %}
      {% if not app.user %}
         <a href="{{ path("user.login") }}" class="btn btn-primary">Se connecter</a>
```

```
        {% else %}
            <span class="h3">Bienvenue : {{ app.user.pseudo }} </span>
            <a href="{{ path('user.logout') }}" class="btn btn-secondary">Se déconnecter</a>
        {% endif %}
    {% endblock %}
</div>
```

4 : De la même façon, la vue « login » fait partie de l'entité « User » (les vues vont avec les actions correspondantes liées à une certaine entité).

Pour le moment, elle se trouve dans le dossier « templates/security/ » alors qu'elle devrai être dans le dossier « templates/user/ » :

1. **Renommer le dossier « security » du dossier « templates » en « user »**
2. **Dans le contrôleur « UserController », modifier l'url qui se trouve dans la méthode « render() » en « user/login.html.twig »**

5 : une fois connecté, nous voulons rediriger l'utilisateur vers la vue qui liste les photos :

Dans le fichier « UserAuthenticator », dans la méthode « onAuthenticationSuccess » décommenter la ligne « return » et mettre la bonne route « photo.list ».

6 : tester

7 : Création du compte utilisateur

a) Création du formulaire

Dans un terminal, entrer la commande :

```
symfony console make:registration-form
ou
php bi/console make :registration-form
```

Répondez aux questions :

```
Creating a registration form for App\Entity\User

Do you want to add a @UniqueEntity validation annotation on your User class to make
sure duplicate accounts aren't created? (yes/no) [yes]:
>

Do you want to send an email to verify the user's email address after registration? (yes/no)
[yes]:
> no

Do you want to automatically authenticate the user after registration? (yes/no) [yes]:
>
```

updated: src/Entity/User.php
created: src/Form/RegistrationFormType.php
created: src/Controller/RegistrationController.php
created: templates/registration/register.html.twig

b) Nous allons renommer et déplacer 2 des 3 fichiers créés pour corresponde à l'entité « User » :

src/Form/RegistrationFormType.php **devient** *src/Form/User/CreateFormType.php*
templates/registration/register.html.twig **devient** *templates/user/create.html.twig*

c) Copier la méthode « register » de la classe « RegistrationController » dans la classe « UserController ».
Ne pas oublier les « use » nécessaires.
Renommer cette méthode « create » puis :

Modifier la ligne de création du formulaire :

```
$form = $this->createForm(CreateFormType::class, $user);
```

Attention au « use » qu'il faut modifier pour la classe « App\Form\User\CreateFormType »

Modifier l'annotation en :

```
#[Route('/create', name : 'user.create')]
```

Modifier l'appel à la méthode « render() » en :

```
…render('user/create.html.twig', ['createForm' => $form->createView(),]) ;
```

Supprimer le fichier « RegistrationController.php »

d) Dans le fichier « CreateFormType.php » :
Renommer la classe « CreateFormType » comme le nom du fichier
Modifier le namespace en « App\Form\User »
Ajouter la création du champ « pseudo » de type « TextType::class » avant l'email.

e) Supprimer le dossier « templates/registration ».

f) Dans la vue « user/create.html.twig », renommer la variable twig « registrationForm » par « createForm » qui a été passée lors de l'appel à la méthode « render » du contrôleur.

g) Afin de rendre les formulaires au « look bootstrap », vous pouvez modifier le fichier de configuration de twig :

Ouvrez le fichier « config/package/twig.yaml » et ajouter la clé « form-themes » comme ceci :

```
twig:
    default_path: '%kernel.project_dir%/templates'
    form_themes: ['bootstrap_5_layout.html.twig']
```

8 : ajouter un bouton de création de compte dans le « jumbotron »

9 : tester en créant un compte utilisateur