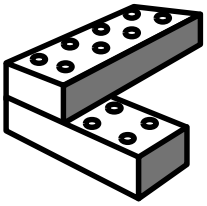

Travaux Pratiques : Enoncés
TP_ReviewPhoto_Partie_03

Objectif : **Consolidation sur le framework Symfony 6**
Création de l'authentification et lien entre entités

Un internaute non inscrit peut visualiser les photos et voir les commentaires.
Ce que l'on veut maintenant, c'est que les personnes inscrites s'authentifient afin qu'elles puissent commenter les photos et en poster de nouvelles.

Nous allons créer en base une table « user ». Cette table servira pour l'authentification des utilisateurs sur le site.

Réalisations :

1 : création de l'entité « User »

Pour gérer l'authentification, Symfony met à notre disposition un maker qui permet de créer une classe utilisateur où tout sera mis en œuvre pour que le système d'authentification fonctionne :

- **Entrer la commande :**

  symfony console make:user

  - **nom de la classe : User (valeur par défaut)**
  - **persistance en base : yes (valeur par défaut)**
  - **propriété unique : email (par défaut c'est email)**
  - **encrypte password : yes (valeur par défaut)**

La différence avec une entité standard, c'est que la classe créée implémente les interfaces « UserInterface » et « PasswordAuthenticatedUserInterface ».
Elle possède en plus une propriété « role » de type ArrayCollection (un tableau)

La commande aura aussi modifié le fichier « security.yaml » du dossier « config/packages »
Enfin elle aura créé le repository qui implémente « PasswordUpgraderInterface »

2 : ajout d'une propriété « pseudo » à l'entité « User »

Faites la commande de création d'une « entity » avec le nom « User »
Symfony va se rendre compte que cette entité existe déjà et va vous proposer d'ajouter des propriété.
Ajoutez unes propriété « pseudo » de type « string », longueur 150 et non null.

3 : Création de la table « user » et des jeux d'essai en base

- **Entrez les commandes :**

symfony console make:migration
symfony console doctrine:migrations:migrate

- **Création des fixtures**

  symfony console make:fixtures UserFixtures

- **On va utiliser la classe « AppFixture » :**

  - **Créer une propriété « $passwordHasher » valorisée par le constructeur ci-dessous :**

```
private $passwordHasher;

public function __construct(UserPasswordEncoderInterface $passwordHasher) {
        $this->passwordHasher = $passwordHasher;
}
```

N'oubliez pas d'ajouter le « *use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;* » en tête du fichier.

  - **Copier la méthode « load() » de la classe « PhotoFixtures » que nous avions créée. Renommez la « loadPhotos() ».**
  - **Créez une méthode « loadUsers() » qui va nous permettre de créer un « User »**

```
public function loadUsers(ObjectManager $manager): void
{
}
```

  - **Dans la méthode « load() » appelez les méthodes « loadPhotos() et « loadUsers() » en leur passant le manger :**

```
public function load(ObjectManager $manager): void
{
   $this->loadPhotos($manager);
   $this->loadUsers($manager);
}
```

  - **Dans la méthode « loadUsers() :**
    - **Créez un User : $user = new User() ;**
    - **Settez l'email : $user->setEmail('user1@dwwm.fr');**
    - **Settez le pseudo : $user->setPseudo('user_1');**
    - **Settez le mot de passe :**

Utiliser la méthode « hashPassword »

$user->setPassword($this->passwordHasher->hashPassword($user, 'user1'));

- **Persister : $manager->persist($user)**
- **Enregistrer en base $manager->flush()**

- **créer les jeux d'essai :**

  symfony console doctrine:fixtures:load

  accepter la confirmation qui va recréer les enregistrements de la table des photos et ceux de la table des utilisateurs

3 : Vérification du fichier de configuration « security.yaml »

La clé « security » doit ressembler à cela :

```
security:
    # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
    # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
    providers:
        # used to reload user from session & other features (e.g. switch_user)
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            lazy: true
            provider: app_user_provider
```

4 : Création du contrôleur d'authentification et du « UserController »

- **Utiliser la commande « make:auth » :**

  symfony console make :auth (ou php bin/console make:auth)

  Choisissez l'option 1
  Entrez le nom de la classe d'authentification : UserAuthenticator
  Laissez par défaut le nom de la classe de sécurité : SecurityController
  Répondez « yes » à la création de la route « logout »

Cette commande à créé plusieurs fichiers et dossiers :

Création du dossier « src/Security » avec la classe « UserAuthenticator ».
Création du contrôleur « SecurityController » qui possède les deux méthodes « login » et « logout ».
Création du dossier templates/security » avec la vue « login.html.twig » qui contient le formulaire de connexion.

5 : Création du bouton de connexion

On va modifier le « jumbotron » qui affiche « Commenter des photos ». Il est présent dans la vue « base.html.twig ».
L'idée est qu'il ait deux parties et qu'il affiche sur sa droite :

- **la date du jour**
- **le bouton de connexion si l'utilisateur n'est pas connecté**
- **le pseudo de celui qui est connecté suivi du bouton de déconnexion**

Recopier le code suivant :

base.html.twig

```
<body>
  <div class="jumbotron text-center">
    <div class="row">
      <div class="col-8">
        <h1>Commenter des photos</h1>
      </div>
      <div class="col-4">
        <h5>Date : {{ "now"|date("m/d/Y") }}</h5>
    {% block connexion %}
        {% if not app.user %}
          <a href="{{ path("app_login") }}" class="btn btn-primary">Se connecter</a>
        {% else %}
          <h4>Bienvenue : {{ app.user.pseudo }} </h4>
          <a href="{{ path('app_logout') }}" class="btn btn-secondary">Se
déconnecter</a>
        {% endif %}
    {% endblock %}
      </div>
    </div>
  </div>
```

Il a été ajouté un nouveau bloc twig appelé « connexion » ce qui va permettre de le redéfinir dans la vue « login.html.twig » comme bloc vide afin que ces boutons de connexion et déconnexion n'apparaissent pas dans cette vue :

login.html.twig

```
{% block title 'Se connecter' %}

  {% block connexion %}{% endblock %}
  {% block body %}
    …
```

7 : tester