ToDo & Co

Documentation technique

Auteur : Luigi GANDEMER

Table des matières

Contexte, Installation du projet & Pré-requis

Contexte : Le projet est une application où nos utilisateurs créé des tâches et peut changer le statut à « *terminé* ».

On a aussi des administrateurs qui peuvent créer des utilisateurs et supprimer des tâches « *anonymes* » qui donc ne sont pas rattaché à un utilisateurs.

Le projet vient d'être mis à jour, initialement il tourné sur **Symfony 3.1 & PHP 5.6.40** on vient de le passer sur **Symfony 6.4 & PHP 8.2**.

Pour l'installation et les pré-requis : voir le **README.md** du repository Github.

2. Authentification

2.1. Fichiers à modifier et pourquoi

`/config/packages/security.yaml`

Ce fichier gère la configuration de sécurité dans Symfony.

Il faudra adapter :

- le **Firewall** pour l'authentification
- Configurer le **Provider** (indique où et comment les utilisateurs sont chargés)
- Le custom **Authenticator** (service responsable de gérer l'authentification des utilisateurs pour le firewall)
- Définir les **règles d'encodage** des mots de passe.

`/src/Entity/User.php`

Cette entité repésente un utilisateur dans notre système. Elle doit absolument implémenter l'interface **UserInterface** de Symfony.

Il faut s'asssurer que l'entité contient les champs nécessaires et les méthodes requises par l'interface **UserInterface**.

`/src/Security/UserAuthenticator.php`

Ce fichier gère le processus d'authentification pour les formulaires de connexion.

Il faudra adapter :

- L'URL de connexion
- La gestion de la réussite
- Le processus d'authentification

`/src/Controller/SecurityController.php`

Ce controller gère la logique, le traitement et la soumission du formulaire de connexion.

`/templates/security/login.html.twig`

Cette vue gère la présentation et le formulaire de la connexion d’un utilisateur.

Il faudra l’adapter en fonction des mise-à-jour.

2.2. Comment s’opère l’authentification

Le processus d’authentification est le suivant :

1. **Demande de connexion :** L’utilisateur soumet son identifiant et son mot de passe via le formulaire.

2. **Firewall :** Le firewall intercepte la demande.

3. **UserAuthenticator :** Le UserAuthenticator analyse les identifiants soumis et initie le processus de vérification des informations de l'utilisateur.

4. **User Provider :** Le provider charge les détails de l’utilisateur.

5. **Vérification des credentials :** Le système compare les informations soumises avec les données chargées par le provider.

6. **Session :** Si les identifiants sont valides alors le système créé une session pour l’utilisateur et redirige l’utilisateur sur le page d’accueil.

7. **Gestion des échecs :** En cas d’échec le système redirige l’utilisateur sur le formulaire de connexion avec un message d’erreur.

2.3. Où sont stocké les utilisateurs

- **Base de données**

Les utilisateurs sont stocké dans notre **base de données MySQL**.

La table « **user** » contient les informations liées aux utilisateurs (nom, email, mot de passe hashé, etc ...)

- **Entity « User »**

L'entité **User** dans Symfony est mappée à la table « **user** » dans la base de données grâce aux annotations ORM de doctrine.

HTTP Cache

Actuellement le temps de cache est fixé à 60 secondes pour les 3 méthodes suivantes :

```
DefaultController::indexAction()
TaskController::listAction()
UserController::listAction()
```

On définie le temps du cache avec l'annotation PHP suivante :

```
#[Cache(smaxage: "60")]
```

À partir du moment où notre nombre d'utilisateurs augmente ainsi que les requête SQL on peut baisser le temps du cache pour que nos vues et données se rafraîchissent avec les données plus récent.

On a juste à diminuer la valeur du « `smaxage` ».