ToDo & Co

Guide technique de l'implémentation de l'authentification

Laura Lazzaro

30 May 2024

Table des matières

Implémentation de l’authentification

Cette documentation vise à aider à comprendre la mise en œuvre de l'authentification dans cette application ToDo. À la fin de ce guide, vous devriez être capable de :

- Comprendre quels fichiers modifier et pourquoi.
- Comprendre comment fonctionne l'authentification.
- Savoir où les données utilisateur sont stockées.

Aperçu

Symfony offre un système de sécurité robuste qui gère l'authentification et l'autorisation. Dans notre application, nous utilisons la fonctionnalité de connexion par formulaire intégrée pour l'authentification des utilisateurs.

Composants Clés

- Fichier de configuration “security.yaml” *: Configure les paramètres de sécurité.*
- *Entité “User” : Représente l'utilisateur dans la base de données.*
- *User Repository : Gère les données des utilisateurs.*
- *Security Controller : Gère les actions de connexion et de déconnexion.*
- *Formulaire de Login : Présente le formulaire de connexion à l'utilisateur.*
- *Security Voter : Gère le contrôle d'accès basé sur les rôles des utilisateurs.*

Fichiers à Modifier

1. config/packages/security.yaml

Ce fichier contient la configuration de sécurité de votre application Symfony.

- **Objectif** : Définir les règles de sécurité, les mécanismes d'authentification et le contrôle d'accès.
- **Sections Clés** :
    - *firewalls* : Configurer la manière dont les utilisateurs s'authentifient (connexion par formulaire dans notre cas).
    - *access_control* : Définir les règles d'accès pour différentes parties de l'application.
    - *providers* : Spécifier où charger les données des utilisateurs (par exemple, base de données).

2. ***src/Entity/User.php***

Ce fichier définit l'entité User, qui représente les utilisateurs dans la base de données.

- **Objectif** : Définir les propriétés de l'utilisateur, comme le nom d'utilisateur et le mot de passe, et implémenter l'interface UserInterface.

3. ***src/Repository/UserRepository.php***

Ce fichier gère la récupération des données des utilisateurs depuis la base de données.

- **Objectif** : Fournir des méthodes de requête personnalisées pour interagir avec la base de données des utilisateurs.

4. ***src/Controller/SecurityController.php***

Ce fichier gère les actions de connexion et de déconnexion.

- **Objectif** : Gérer l'authentification des utilisateurs et la logique de redirection.

5. ***templates/security/login.html.twig***

Ce fichier contient le HTML pour le formulaire de connexion.

- **Objectif** : Fournir une interface utilisateur pour la connexion.

Fonctionnement de l'Authentification

- **Soumission du formulaire de connexion :** L'utilisateur accède à la route `/login` et soumet son email et son mot de passe via le formulaire de connexion.
- **Gestion de la connexion par formulaire :** Symfony écoute les tentatives de connexion. Il utilise les identifiants pour charger l'utilisateur depuis la base de données via le UserRepository.
- **Vérification de l'utilisateur :** L'encodeur de mot de passe vérifie le mot de passe fourni par rapport au hachage stocké dans la base de données.
- **Connexion réussie :** En cas de succès, l'utilisateur est authentifié et obtient un accès basé sur ses rôles. Il est redirigé vers la page avec la liste de taches à faire.
- **Connexion échouée :** En cas d'échec, un message d'erreur est affiché et l'utilisateur est invité à réessayer.

Comment Les Utilisateurs Sont Stockés

Les utilisateurs sont stockés dans une base de données SQLite définie par l'entité User et configurée dans le fichier .env.
La classe UserRepository fournit des méthodes pour interroger et interagir avec les données des utilisateurs.

- **Configuration de la Base de Données** : Assurez-vous que le fichier .env contient les paramètres de connexion à la base de données corrects.
- **Table Utilisateur** : L'entité User est associée à une table dans la base de données où les données des utilisateurs sont stockés.