Documentation technique Authentification Symfony

1. Introduction

Cette documentation explique comment l'authentification est implémentée dans le projet ToDo List. Elle couvre :

- les fichiers à modifier et leur rôle,
- le fonctionnement de l'authentification,
- le stockage des utilisateurs,
- les formulaires de login et de gestion (édition) des utilisateurs,
- les bonnes pratiques pour collaborer sur le projet.

2. Fichiers clés et leur rôle

- /config/packages/security.yaml

C'est le fichier central de la configuration de sécurité Symfony. Il définit :

- Le hachage des mots de passe, configuré sur "auto" afin de permettre à Symfony de choisir automatiquement l'algorithme le plus sûr disponible sur le serveur.
- Le provider (récupère les users) : ici, l'entité User et le champ username sont utilisés pour identifier les users.
- Le firewall : gère l'authentification et les routes associées : connexion, déconnexion et protection des pages.
- Le contrôle des autorisations d'accès :
    - tout le monde peut accéder à la page de login,

    - seul un admin peut accéder aux pages concernant les users
    - n'importe quel user connecté peut accéder au reste des pages

Attention, pour les autorisations, Symfony applique la première règle qui correspond au chemin, il faut donc aller du cas le plus précis au cas le plus général.

- /SRC/ENTITY/USER.PHP

Ce fichier définit l'entité User/ la structure de la table User dans la bdd.

Contrainte d'unicité sur les champs email et username.

Méthodes importantes :

- getUsername() : retourne l'email pour identifier un user. (A noter que le formulaire de login utilise le username, comme vu dans security.yaml)
- getRoles() : retourne les rôles du user, garantit toujours qu'un user enregistré a au moins le `ROLE_USER`.
- getPassword()/setPassword() : gère l'accès au mot de passe hashé

LE STOCKAGE DES UTILISATEURS :

→ Les users sont stockés en base de données via Doctrine.

→ La table user contient les colonnes : id, email, username, password, roles.

→ Les mots de passe sont hashés avant insertion en base pour sécuriser les comptes.

- /SRC/CONTROLLER/SECURITYCONTROLLER.PHP

Méthode login() :

- Affiche le formulaire LoginType (détaillé ci-dessous), éventuellement pré-rempli avec le nom du dernier user connecté.
- Récupère les erreurs de connexion via AuthenticationUtils.
- Redirige les utilisateurs déjà connectés vers la page d'accueil.

L'AUTHENTIFICATION :

Le User va sur la page /login et soumet le formulaire de connexion

→ Symfony récupère le User via le provider

→ Symfony vérifie le username et le password (hash)

→ si OK : création d'une session active, attribution du rôle du user et redirection vers / (page d'accueil)

→ si KO : erreur affichée sur le formulaire

Méthode logout() :

- Symfony intercepte la route /logout pour détruire la session.
- La méthode peut rester vide.

/SRC/CONTROLLER/USERCONTROLLER.PHP

Ce fichier gère la création, modification et affichage des utilisateurs. Comme mentionné ci-dessus, ces pages sont accessibles uniquement à un user ayant le rôle admin.

1. list() : liste tous les utilisateurs.
2. create() : création d'un nouvel utilisateur :
    - récupère les données du formulaire UserType (détaillé ci-dessous) et les valide,
    - hashe le mot de passe avec UserPasswordHasherInterface,
    - enregistre le user en base de données,
    - message Flash de succès/renvoie sur le formulaire en cas d'erreur.
3. edit() : modification d'un utilisateur :
    - récupère les données du formulaire UserType et les valide,
    - si le mot de passe est modifié, il est hashé avant sauvegarde,

- message Flash de succès/renvoie sur le formulaire en cas d'erreur.

- /SRC/FORM/LOGINTYPE.PHP

C'est le formulaire de connexion, avec les champs _username (pseudo) et _password.

On utilise un token CSRF pour sécuriser le formulaire.

- /SRC/FORM/USERTYPE.PHP

C'est le formulaire de création/modification d'un utilisateur, avec les champs username, email, password (répété pour vérification) et rôles.
ATTENTION : dans la mission, il est demandé qu'un utilisateur ne puisse choisir qu'un seul rôle. Dans Symfony, un utilisateur a au moins un rôle et peut en avoir plusieurs (array). Pour respecter ces contraintes, les rôles apparaissent en tant que string dans le formulaire et sont transformés en array par la suite.

On utilise un token CSRF pour sécuriser le formulaire.

3. Pour aller plus loin

Pour ajouter un rôle :

- mettre à jour UserType et access_control dans security.yaml.
- ajouter le rôle aux utilisateurs existants si nécessaire.