ToDo & Co

L’authentification sur Symfony

Entité **User** et **SecurityBundle**

L'entité User

Sur Symfony, les permissions sont liées à un une classe *utilisateur* et, dans notre projet, cette classe est une entité Doctrine.

Cela signifie que nos utilisateurs sont stockés en base de données, dans une table ***user***, qui a pour modèle l'entité **User**.

Cette entité User a été créée avec la commande :

`php bin/console make:user`

Cette commande nous permet de générer rapidement la classe User en configurant seulement :

- Le nom de l'entité : *User*
- Le stockage des utilisateurs : *Base de données*
- Une propriété d'affichage unique par utilisateur : *Le nom d'utilisateur*
- L'encodage sur les mots de passes : *Activé*

Les rôles des utilisateurs

Pour gérer les accès aux ressources de notre application, les utilisateurs ont différents rôles attribués.
Deux rôles sont alors définis selon nos besoins :

- *ROLE_ADMIN* : L'administrateur gère les utilisateurs, ses tâches et les tâches anonymes
- *ROLE_USER* : L'utilisateur standard gère uniquement ses tâches

Symfony récupère les rôles de l'utilisateur à l'authentification et vérifie, à chaque requête, s'il a le droit d'accéder à la ressource demandée.

Pour nos besoins, cette configuration d'accès par rôles est configurée en premier lieu dans le fichier **security.yaml** (voir plus bas).

Nous utilisons aussi des méthodes dans nos templates Twig, par exemple pour afficher les liens vers les pages des administrateurs :

```
{% if is_granted("ROLE_ADMIN") %}
        <a href="{{ path('user_create') }}">Créer un utilisateur</a>
        <a href="{{ path('user_list') }}">Liste des utilisateurs</a>
{% endif %}
```

Cependant, il est bon de savoir qu'il est aussi possible de gérer les droits d'accès dans les contrôleurs :

- Avec l'attribut *#[IsGranted]* sur une méthode :

```
#[IsGranted('ROLE_ADMIN')]
#[Route('/users', name: 'user_list')]
public function listUsers(): Response
{ ... }
```

- Avec la méthode *denyAccessUnlessGranted* :

```
#[Route('/users', name: 'user_list')]
public function listUsers(): Response
{
    $this->denyAccessUnlessGranted("ROLE_ADMIN");
}
```

Ces deux options renverrait alors une erreur **403** (*Access Denied*) si l'utilisateur n'a pas le rôle Administrateur et essaie d'exécuter ce code.

Enfin, il est à noter qu'il est bon de passer par la méthode `$this->isGranted('ROLE_***');` pour vérifier les accès des rôles d'un utilisateur, plutôt que de comparer directement la valeur de retour du tableau de `$user->getRoles()`.

En effet, ***isGranted()*** prend en compte la hiérarchie des rôles (voir plus bas).

Configuration du SecurityBundle de Symfony

Dans la mise à niveau de notre projet vers Symfony 6, on peut voir dans le composer.json le bundle “symfony/security-bundle”.

Ce bundle est configuré par le fichier “ ***/config/packages/security.yaml***”.

1. Encodeurs de mot de passe et stockage des utilisateurs

```yaml
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:
        users_in_memory: { memory: null }
        app_user_provider:
            entity:
                class: App\Entity\User
                property: username
```

- La valeur ‘auto’ pour l’encodeur de mot de passe sélectionne automatiquement le meilleur encodeur disponible, soit **bcrypt** en ce moment.

- Dans ‘providers’, on déclare que nos utilisateurs sont stockés dans la base de données, avec l’**entité User**.
  Notre base contiendra alors une table ‘*user*’ sur le modèle de cette entité.

- Enfin, on déclare que nos utilisateurs sont identifiés de façon unique par leur propriété ‘*username*’.
  Cela induit alors que la connexion d’un utilisateur se fera par le couple **nom d’utilisateur** et **mot de passe**.

2. Les firewalls

```yaml
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        provider: app_user_provider
        security: false
    main:
        lazy: true
        provider: app_user_provider
        form_login:
            login_path: login
            check_path: login
        logout:
            path: logout
```

On peut voir 2 firewall déclarés dans notre fichier de configuration :

- Le firewall **dev** est présent pour être sûr que l'on ne bloque pas l'affichage des outils de développement symfony en mode développement (*'APP_ENV=dev'* dans le fichier ".env" à la racine du projet).

  Il gère par exemple l'affichage de la toolbar Symfony.

- Le firewall **main** gère toutes les autres URLs, et donc toutes les pages de notre application.

Plusieurs attributs sont présents dans nos firewalls, notamment :

- Le **provider** d'utilisateurs utilisé : On retrouve "*app_user_provider*", déclaré plus haut dans le fichier (l'entité User en base de données).

- Les **paths** de connexion, et déconnexion.
  La connexion d'un utilisateur se fait alors par un formulaire de connexion ayant pour **path** "login", et une déconnexion par le **path** "logout".

  Symfony gère automatiquement la redirection vers la page de login si un visiteur essaie d'accéder à une ressource demandant une authentification.

3. Les hiérarchies de rôles et les contrôles d’accès

```
role_hierarchy:
    ROLE_ADMIN: ROLE_USER
```

Avec cette configuration de hiérarchie des rôles, les utilisateurs Administrateurs héritent des droits des utilisateurs standard.

```
access_control:
    # Visitors can see tasks lists
    - { path: ^/tasks/(done|todo), roles: PUBLIC_ACCESS }
    # Only users can create, edit or delete a task
    - { path: ^/tasks/, roles: ROLE_USER }
    # Only admin can create, edit or delete a user
    - { path: ^/users, roles: ROLE_ADMIN }
```

Dans la partie **access_control**, on peut configurer les droits de certains rôles sur des parties du site.

Avec la configuration actuelle:

- On autorise les visiteurs (accès public) à voir les listes de tâches en cours et terminées.
- Les opérations sur les tâches sont limitées aux utilisateurs authentifiés standard (et sont permises aussi aux administrateurs, par héritage des droits)
- Les pages de modifications d’utilisateurs sont réservés aux administrateurs