L'authentification sur symfony

La partie authentification est gérée dans symfony par le security bundle. Le security bundle est bien sur déjà installé sur ce projet, vous pouvez le voir dans le fichier composer.json dans la partie require. Si vous travaillez sur un projet ou le security bundle n'est pas installé vous pouvez l'installer via composer avec la commande suivante.

> composer require symfony/security-bundle

L'entité User

Pour gérer vos utilisateurs vous avez besoin d'une entité User. Cette classe représente vos utilisateurs en base de donnée. La seule règle pour qu'elle soit valide est qu'elle doit implémenter la UserInterface. Dans ce projet cette classe existe déjà vous pouvez la trouver ici : src/Entity/User.php

Si vous travaillez dans un projet ou il n'y à pas d'entité User, vous pouvez la créer facilement avec la commande suivante

> php bin/console make:user

Attention vous avez besoin du symfony/maker-bundle pour réaliser cette opération. N'oubliez pas ensuite d'exécuter vos migrations pour la nouvelle entité.

> php bin/console make:migration php bin/console doctrine:migrations:migrate

L'encodage

Le security bundle vous donne accès à un fichier de configuration que nous allons voir en détail. Ce fichier de configuration est accessible ici : config/packages/security.yaml

La partie encoders va tout simplement indiquer qu'elle algorithme d'encodage va être utilisé pour chiffrer les mots de passe des utilisateurs

```
# config/packages/security.yaml
security:
    encoders:
        App\Entity\User: bcrypt
```

On voit ici l'utilisation de l'algorithme bcrypt pour notre encodage.

Les User providers

Les User providers ou « fournisseur d'utilisateurs » en français sont des classes PHP qui vont permettre de rattacher un utilisateur souhaitant s'authentifier à un utilisateur en base de donnée.

```
providers:
        in_database:
            entity:
                class: App\Entity\User
                property: username
```

On voit ici notre classe User qui est indiquée et on voit également la propriété username qui va être utilisé comme identifiant dans le formulaire de connexion.

A noté si vous travaillez dans un autre projet que la commande make:user vu précédemment vous créera directement votre configuration

Les Firewalls

Avec le firewall ou « pare-feu » vous allez définir comment vos utilisateurs pourront s'authentifier.

```
firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false

        main:
            logout_on_user_change: true
            anonymous: ~
            pattern: ^/
            form_login:
                login_path: login
                check_path: login_check
                always_use_default_target_path:  true
                default_target_path:  /
            logout: ~
```

Vous n'avez pas à vous soucier de la partie dev, elle sert pendant la phase de développement à ce que vous ne bloquiez pas accidentellement les outils de développement de symfony. C'est le firewall main qui va gérer la totalité de votre site. La ligne pattern: ^/ vous indique justement que tout les url de votre site à partir de la racine sont gérés par ce firewall main.

Un firewall peut avoir plusieurs modes d'authentification. Le plus souvent on utilise comme ici le mode anonymous. Ce mode permet à n'importe qu'elle utilisateur arrivant sur le site d'être authentifié comme anonymous. Le pare-feu vérifie de cette façon qu'il ne connaît pas l'identité de la personne. Cela lui donne tout de même accès à certaine ressource, notamment au formulaire de connexion pour pouvoir s'authentifier comme utilisateur du site.

Le formulaire de connexion est définit dans la partie form_login.

Pour pouvoir ensuite avoir plus de contrôle sur l'authentification vous avez accès au security controller. Vous avez accès également à votre templete de formulaire de connexion login.html.twig. Vous pouvez le personnaliser selon vos besoin.

A noté si vous travaillez dans un nouveau projet que vous pouvez générer facilement votre système d'authentification avec la commande suivante :

> php bin/console make:auth

Cela vous générera automatiquement votre controller et votre templete ainsi que votre fichier security.yaml avec les valeurs par défaut.

Nous allons maintenant voir la dernière partie de notre fichier security.yaml

Les Acces Control

Un acces-control va définir une limite d'accès à un url. Seul les personnes ayant le rôle spécifié pourront accéder à cette partie du site.

```
access_control:
        - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/users, roles: ROLE_ADMIN }
        - { path: ^/, roles: ROLE_USER }
```

On voit par exemple ici que tout les utilisateurs anonymes peuvent accéder à la route /login. Par contre seule les admins peuvent accéder à l'url /users. Et enfin on peut voir que les utilisateurs authentifié grace au formulaire peuvent accéder à l'ensemble du site (hormis la route /users évidemment).

Noté que toute personne connecté via le formulaire posséde au moins le rôle ROLE_USER. Vous pouvez voir cela dans l'entité User au niveau de la fonction getRoles().

```
public function getRoles(): array
    {
        $roles = $this->roles;
        // guarantee every user at least has ROLE_USER
        $roles[] = 'ROLE_USER';

        return array_unique($roles);
    }
```

Sur ce projet vous avez donc 3 type d'utilisateur.

1. L'utilisateur anonyme
2. L'utilisateur connecté qui à le rôle ROLE_USER
3. L'administrateur qui à les 2 rôles ROLE_USER et ROLE_ADMIN