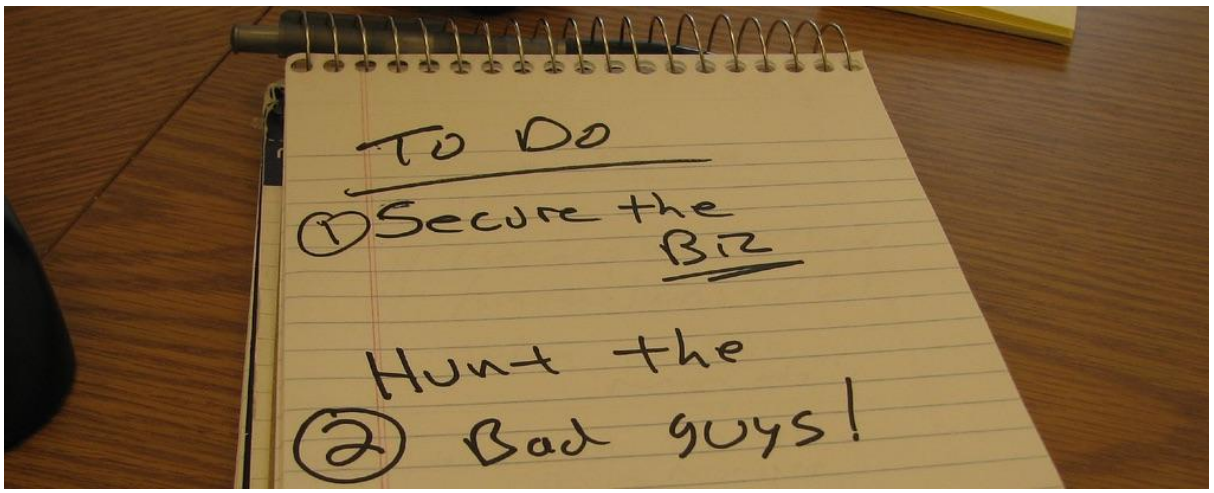

Documentation technique
ToDo & Co

Sommaire

1. Introduction

L'authentification s'effectue en plusieurs temps. Il faut tout d'abord créer une entité User sur Symfony pour définir les caractéristiques de chaque user avec les champs correspondants (email, username, password, etc…).
Ensuite, il faut gérer la sécurité dans security.yaml pour définir notre authentification.

2. Gestion de l'entité User

a. Créer l'entité

- Créer l'entité User en ligne de commande : **php bin/console make:user**
- Suivre les étapes du terminal.
- Une fois terminé, vérifier que tous les champs nécessaires au projet sont définis. S'il en manque, il faut taper en ligne de commande :
  php bin/console make:entity, puis, choisir l'entité User. Vous pourrez alors ajouter les champs manquants.

b. Annotations

Par sécurité, il est nécessaire de définir des conditions pour les utilisateurs.

- Importer la classe Assert :

use Symfony\ComponentValidator\Constraints as Assert

- Définir les Assert pour l'username comme sur la photo suivante :

```php
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=180, unique=true)
     * * @Assert\NotBlank(
     *      message = "Ce champ est requis !"
     * )
     * @Assert\Length(
     *      min = 3,
     *      max = 20,
     *      minMessage = "Votre nom d'utilisateur doit contenir au moins {{ limit }} caractères !",
     *      maxMessage = "Votre nom d'utilisateur ne peut pas contenir plus que {{ limit }} caractères !"
     * )
     */
    private $username;

    /**
     * @ORM\Column(type="json")
     */
    private $roles = [];
```

Vous pouvez définir les conditions comme vous voulez.

- De même pour le mot de passe :

```php
/**
 * @var string The hashed password
 * @ORM\Column(type="string")
 * * @Assert\NotBlank(
 *      message = "Ce champ est requis !"
 * )
 * @Assert\Length(
 *      min = 6,
 *      max = 254,
 *      minMessage = "Votre mot de passe doit contenir au moins 6 caractères.",
 *      maxMessage = "Votre mot de passe ne peut pas contenir plus que {{ limit }} caractères !"
 * )
 * @Assert\Regex(
 *     pattern = "^(?=.*[a-z])(?=.*[A-Z])^",
 *     match = true,
 *     message = "Le mot de passe doit contenir au moins une minuscule et une majuscule !"
 * )
 */
private $password;
```

- Et pour l'email :

```php
/**
* @ORM\Column(type="string", length=50)
* * @Assert\NotBlank(
*      message = "Ce champ est requis !"
* )
* @Assert\Email(
*      message = "Veuillez entrer une adresse email valide."
* )
* @Assert\Length(
*      max = 254,
*      maxMessage = "Votre adresse email ne peut pas contenir plus de {{ limit }} caractères."
* )
*/
private $email;
```

3. Gérer la sécurité

Le fichier securiy.yaml permet de gérer la sécurité sur Symfony. Il se trouve dans **config/packages**
Il est très important de respecter l'indentation dans ce fichier.

a. Provider

Il permet d'authentifier des utilisateurs en toute sécurité.
Il faut définir l'authentification sur notre entité créée au préalable, User. Enfin, il faut que l'authentification se fasse par le username, conformément au projet.

```yaml
providers:
    # used to reload user from session & other features (e.g. switch_user)
    app_user_provider:
        entity:
            class: App\Entity\User
            property: username
```

b. Firewall

Le pare-feu permet de définir les règles d'authentification de l'utilisateur.

```yaml
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        lazy: true
        provider: app_user_provider
        form_login:
            # "login" is the name of the route created previously
            login_path: account_login
            check_path: account_login
        logout:
            path: account_logout
```

- Le dev pare-feu est vraiment un faux pare-feu : il s'assure que vous ne bloquez pas accidentellement les outils de développement de Symfony - qui vivent sous des URL comme /_profileret /_wdt.

- Le form_login permet d'indiquer que l'authentification se fait par un formulaire de connexion.
- Le login_path et logout_path indiquent les noms des routes pour se connecter et se déconnecter.

c. Access Control

Cette partie permet de restreindre l'accès à certaines pages.

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

Un utilisateur avec le rôle Administrateur pourra accéder à toutes les pages du site alors qu'un utilisateur avec le rôle user ne pourra pas accéder aux pages commençant par "/users".
Il est aussi possible de renseigner un rôle anonyme (IS_AUTHENTICATED_ANONYMOUSLY) qui permet de donner l'accès anonymement (non connecté) à la page login.

d. Role Hierarchy

Il permet de donner l'héritage des rôles. Ainsi, le rôle Admin héritera du rôle User, et lui permettra d'accéder aux mêmes pages que le User (en plus des pages admin).

```
role_hierarchy:
    ROLE_ADMIN: [ROLE_USER]
```

4. La sécurité dans SecurityController

C'est dans ce controller que se trouve les méthodes de l'authentification.
On peut voir que le nom de la route correspond au login_path dans le security.yaml.

```php
/**
 * @Route("/login", name="account_login")
 */
public function loginAction(AuthenticationUtils $authenticationUtils)
{
    //$authenticationUtils = $this->get('security.authentication_utils');

    $error = $authenticationUtils->getLastAuthenticationError();
    $lastUsername = $authenticationUtils->getLastUsername();

    return $this->render('security/login.html.twig', array(
        'last_username' => $lastUsername,
        'error'         => $error,
    ));
}
```

5. Stockage des Users

Les utilisateurs sont stockés dans une base de données et sont accessibles depuis un gestionnaire de base de données MySQL.
Pour définir la base de données :

- se rendre dans le .env
- le renommer en .env.local
- changer **DATABASE_URL="mysql://votreusername:votrepassword@127.0.0.1:3306/lenomdevotredb**