



# Implémentation de l'authentification

*Documentation technique*

Filâtre Killian  
21/06/2024

## Sommaire :

- Création de l'entité User
- Migration de base de données
- Dépot de l'utilisateur
- SecurityBundle
- Création du formulaire de connexion
- Création du système d'authentification
- Conclusion

---

## Introduction

L'authentification dans Symfony 6.4 permet de sécuriser l'accès aux parties de votre application. Cette documentation vous guide à travers les étapes nécessaires pour mettre en place l'authentification en utilisant Symfony 6.4.

### Pré-requis

- Symfony 6.4 installé
- Composer installé
- Une base de données configurée

## Création de l'entité User

En premier lieu, avant de pouvoir authentifier un utilisateur, nous avons besoin de créer une entité User qui comprendra les informations de l'utilisateur. Les utilisateurs sont représentés par une entité Doctrine. Une entité est une classe PHP annotée avec des métadonnées ORM pour définir la structure de la table et les relations avec d'autres entités.

```
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    1 usage
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    2 usages
    #[ORM\Column(length: 255)]
    private ?string $email = null;

    2 usages
    #[ORM\Column(length: 255)]
    private ?string $username = null;

    2 usages
    #[ORM\Column(length: 255)]
    private ?string $password = null;

    2 usages
    #[ORM\Column(length: 255)]
    private array $roles = [];

    /** @var Collection<int, Task> ...*/
    5 usages
    #[ORM\OneToMany(targetEntity: Task::class, mappedBy: 'id_user', orphanRemov
    private Collection $tasks;
```

---

Cette entité User a quelques méthodes que Symfony doit utiliser pour la gestion de la sécurité, telles que `getRoles()`, `getSalt()`, et `eraseCredentials()`.

## Migration de Base de Données

Une fois l'entité définie, il faut générer et exécuter une migration pour créer la table correspondante dans la base de données. Utilisez les commandes suivantes :

### Génération de la migration :

=> `php bin/console make:migration`

### Exécution de la migration :

=> `php bin/console doctrine:migrations:migrate`

## Dépôt de l'Utilisateur

Le dépôt (repository) est utilisé pour interagir avec la base de données. Symfony génère un dépôt de base lors de la création de l'entité utilisateur. Voici un exemple de dépôt utilisateur :

```
1  <?php
2
3  namespace App\Repository;
4
5  use ...
6
7
8
9  /**
10   * @extends ServiceEntityRepository<User>
11   */
12   class UserRepository extends ServiceEntityRepository
13   {
14       public function __construct(ManagerRegistry $registry)
15       {
16           parent::__construct($registry, User::class);
17       }
18   }
```

## SecurityBundle

SecurityBundle est le composant principal responsable de l'authentification et de l'autorisation dans Symfony. SecurityBundle utilise une combinaison d'encodeurs de mots de passe, de providers de l'utilisateur, de firewalls, et de règles d'accès pour gérer l'authentification et l'autorisation.

### Configuration du fichier de sécurité

La config du SecurityBundle se fait dans le fichier `config/packages/security.yaml` :

Voici un exemple :

```
1 security:
2     # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
3     password_hashers:
4         Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
5     # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
6     providers:
7         app_user_provider:
8             entity:
9                 class: App\Entity\User
10                property: username
11    firewalls:
12        dev:
13            pattern: ^/(_(profiler|wdt)|css|images|js)/
14            security: false
15        main:
16            lazy: true
17            provider: app_user_provider
18            form_login:
19                login_path: app_login
20                check_path: app_login
21                always_use_default_target_path: true
22                default_target_path: /
23                enable_csrf: true
24            logout:
25                path: app_logout
26                target: app_login
```

Le provider de l'utilisateur est défini pour charger des utilisateurs à partir de l'entité User, en utilisant le username comme identifiant. Le firewall est configuré pour utiliser le provider `app_user_provider`, avec une protection par formulaire pour l'authentification.

## Création du formulaire de connexion

Nous devons ensuite créer un formulaire de connexion avec un route /login via la commande : `php bin/console make:auth`

```
11 class SecurityController extends AbstractController
12 {
13     /**
14      * Handles the login functionality.
15      *
16      * @param AuthenticationUtils $authenticationUtils The authentication utility.
17      * @return Response The response object.
18      */
19     #[Route(path: '/login', name: 'app_login')]
20     public function login(AuthenticationUtils $authenticationUtils): Response
21     {
22         // Get the login error if there is one
23         $error = $authenticationUtils->getLastAuthenticationError();
24
25         // Last username entered by the user
26         $lastUsername = $authenticationUtils->getLastUsername();
27
28         return $this->render(view: 'security/login.html.twig', [
29             'last_username' => $lastUsername,
30             'error' => $error,
31         ]);
32     }
```



---

## Création du système d'authentification

L'authentification est gérée par des authenticators (Guard Authenticators). Un authenticator est une simple classe PHP qui fait trois choses :

- Lire les informations d'authentification du message de requête (c'est-à-dire du formulaire de connexion).
- Valide ces informations
- Gère une réponse réussie ou échouée à la suite de l'authentification (succès ou échec).

## Conclusion

Ce document fournit un aperçu de base de l'implémentation de l'authentification dans Symfony 6.4. Pour plus d'informations, veuillez consulter la documentation officielle de la sécurité Symfony - <https://symfony.com/doc/6.4/security.html>