

# Symfony 5 : gestion d'utilisateurs

**Achref El Mouelhi**

Docteur de l'université d'Aix-Marseille  
Chercheur en Programmation par contrainte (IA)  
Ingénieur en Génie logiciel

`elmouelhi.achref@gmail.com`



# Plan

- 1 Introduction
- 2 Création d'utilisateur
- 3 Préparation de l'authentification
- 4 Déconnexion
- 5 Contrôle d'accès
  - Dans `security.yaml`
  - Dans le contrôleur
  - Dans la vue
- 6 Utilisateur authentifié
  - Dans le contrôleur
  - Dans la vue
- 7 Rôles hiérarchiques

# Symfony

## But de la sécurité

Interdire, à un utilisateur, l'accès à une ressource à laquelle il n'a pas droit

© Achref EL ME

# Symfony

## But de la sécurité

Interdire, à un utilisateur, l'accès à une ressource à laquelle il n'a pas droit

## Deux étapes

- Qui veut accéder à la ressource ?
- A t-il le droit d'y accéder ?

## Configuration de la sécurité

- En utilisant des données statiques (en mémoire)
- En utilisant des données dynamiques (stockées dans une base de données)

© Achre

# Symfony

## Configuration de la sécurité

- En utilisant des données statiques (en mémoire)
- En utilisant des données dynamiques (stockées dans une base de données)

## Pour cela

On va utiliser un bundle **Symfony** à savoir `security-bundle`

## Configuration de la sécurité

- En utilisant les annotations
- Et en définissant quelques règles dans `config/packages/security.yml`
- Mais on peut aussi utiliser :
  - le format XML
  - les tableaux imbriqués de PHP

# Symfony

## Contenu de security.yaml

```
security:
    # https://symfony.com/doc/current/security.html#where-do-users-come
    # -from-user-providers
    providers:
        users_in_memory: { memory: null }
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            anonymous: lazy
            provider: users_in_memory
    access_control:
        # - { path: ^/admin, roles: ROLE_ADMIN }
        # - { path: ^/profile, roles: ROLE_USER }
```



# Symfony

## Plusieurs étapes

- Préparation de la partie utilisateur (qui va se connecter)
- Préparation de la partie authentification (formulaire d'authentification, déconnexion...)
- Gestion de rôles

# Symfony

Si on n'a pas choisi la version complète à la création du projet, exécutez

- `composer require symfony/security-bundle`

# Symfony

## Pour créer la classe `User`

- exécutez la commande `php bin/console make:user`
- répondez à The name of the security user class **par** `User`
- répondez à Do you want to store user data in the database (via Doctrine)? **par** `yes`
- répondez à Enter a property name that will be the unique "display" name for the user **par** `email`
- répondez à Does this app need to hash/check user passwords? **par** `yes`



# Symfony

## Pour créer la classe `User`

- exécutez la commande `php bin/console make:user`
- répondez à The name of the security user class **par** `User`
- répondez à Do you want to store user data in the database (via Doctrine)? **par** `yes`
- répondez à Enter a property name that will be the unique "display" name for the user **par** `email`
- répondez à Does this app need to hash/check user passwords? **par** `yes`

## Le résultat est

```
created: src/Entity/User.php
created: src/Repository/UserRepository.php
updated: src/Entity/User.php
updated: config/packages/security.yaml
```

## Nouveau contenu de security.yaml

```
security:
  encoders:
    App\Entity\User:
      algorithm: auto

  # https://symfony.com/doc/current/security.html#where-do-users-come
  # -from-user-providers
  providers:
    # used to reload user from session & other features (e.g.
    # switch_user)
    app_user_provider:
      entity:
        class: App\Entity\User
        property: email
  firewalls:
    dev:
      pattern: ^/(_(profiler|wdt)|css|images|js)/
      security: false
    main:
      anonymous: lazy
      provider: app_user_provider

  access_control:
```

# Symfony

## Pour créer la table `User`

- exécutez la commande `php bin/console make:migration`
- et ensuite la commande `php bin/console doctrine:migrations:migrate`

© Achref EL MOU

# Symfony

## Pour créer la table `User`

- exécutez la commande `php bin/console make:migration`
- et ensuite la commande `php bin/console doctrine:migrations:migrate`

## Pour remplir la table `User` avec des données aléatoires

- installez le bundle de fixture `composer require --dev doctrine/doctrine-fixtures-bundle`
- demandez à ce bundle de remplir la table `php bin/console make:fixtures`
- répondez à The class name of the fixtures to create **par** `UserFixtures`

# Symfony

## Contenu généré pour UserFixtures

```
namespace App\DataFixtures;

use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Persistence\ObjectManager;

class UserFixtures extends Fixture
{
    public function load(ObjectManager $manager)
    {
        // $product = new Product();
        // $manager->persist($product);

        $manager->flush();
    }
}
```



## Nouveau contenu de UserFixtures

```

class UserFixtures extends Fixture
{
    private $passwordEncoder;

    public function __construct(UserPasswordEncoderInterface $passwordEncoder)
    {
        $this->passwordEncoder = $passwordEncoder;
    }

    public function load(ObjectManager $manager)
    {
        $user = new User();
        $user->setEmail('wick@wick.us');
        $user->setRoles(['ROLE_ADMIN']);
        $user->setPassword($this->passwordEncoder->encodePassword(
            $user,
            'wick'
        ));
        $manager->persist($user);
        $user2 = new User();
        $user2->setEmail('john@john.us');
        $user2->setPassword($this->passwordEncoder->encodePassword(
            $user2,
            'john'
        ));
        $manager->persist($user2);
        $manager->flush();
    }
}

```

## Nouveau contenu de UserFixtures

```

class UserFixtures extends Fixture
{
    private $passwordEncoder;

    public function __construct(UserPasswordEncoderInterface $passwordEncoder)
    {
        $this->passwordEncoder = $passwordEncoder;
    }

    public function load(ObjectManager $manager)
    {
        $user = new User();
        $user->setEmail('wick@wick.us');
        $user->setRoles(['ROLE_ADMIN']);
        $user->setPassword($this->passwordEncoder->encodePassword(
            $user,
            'wick'
        ));
        $manager->persist($user);
        $user2 = new User();
        $user2->setEmail('john@john.us');
        $user2->setPassword($this->passwordEncoder->encodePassword(
            $user2,
            'john'
        ));
        $manager->persist($user2);
        $manager->flush();
    }
}

```

## Les use nécessaires

```

use App\Entity\User;
use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;

```

# Symfony

Pour insérer l'utilisateur dans la base de données, exécutez

```
php bin/console doctrine:fixtures:load ou php  
bin/console d:f:l
```

## À partir du terminal, exécutez la commande suivante

```
php bin/console make:auth
```

```
What style of authentication do you want? [Empty authenticator]:
```

```
[0] Empty authenticator
```

```
[1] Login form authenticator
```

```
> 1
```

```
The class name of the authenticator to create (e.g.
```

```
    AppCustomAuthenticator):
```

```
> LoginFormAuthenticator
```

```
Choose a name for the controller class (e.g. SecurityController) [
```

```
    SecurityController]:
```

```
> SecurityController
```

```
Do you want to generate a '/logout' URL? (yes/no) [yes]:
```

```
> yes
```

## À partir du terminal, exécutez la commande suivante

```
php bin/console make:auth
```

```
What style of authentication do you want? [Empty authenticator]:
```

```
[0] Empty authenticator
```

```
[1] Login form authenticator
```

```
> 1
```

```
The class name of the authenticator to create (e.g.
```

```
AppCustomAuthenticator):
```

```
> LoginFormAuthenticator
```

```
Choose a name for the controller class (e.g. SecurityController) [
```

```
SecurityController]:
```

```
> SecurityController
```

```
Do you want to generate a '/logout' URL? (yes/no) [yes]:
```

```
> yes
```

## Le résultat est

```
created: src/Security/LoginFormAuthenticator.php
```

```
updated: config/packages/security.yaml
```

```
created: src/Controller/SecurityController.php
```

```
created: templates/security/login.html.twig
```

# Symfony

Pour tester, allez sur la route `/login`

- essayez de vous connecter avec un email inexistant
- ensuite essayez de vous connecter avec un email existant et un mot de passe incorrect
- enfin connectez-vous avec `wick@wick.us` et `wick`

© Active

# Symfony

Pour tester, allez sur la route `/login`

- essayez de vous connecter avec un email inexistant
- ensuite essayez de vous connecter avec un email existant et un mot de passe incorrect
- enfin connectez-vous avec `wick@wick.us` et `wick`

## Remarque

Problème de redirection après la connexion

# Symfony

**Pour résoudre ce problème, il faut modifier la méthode `onAuthenticationSuccess` définie dans `security/LoginFormAuthenticator` pour rediriger vers la route `home_route`**

```
public function onAuthenticationSuccess(Request
    $request, TokenInterface $token, $providerKey)
{
    if ($targetPath = $this->getTargetPath($request
        ->getSession(), $providerKey)) {
        return new RedirectResponse($targetPath);
    }

    return new RedirectResponse($this->urlGenerator
        ->generate('home_route'));
}
```



# Symfony

Pour modifier les messages d'erreurs de la page d'accueil, créez un fichier `security.en.xlf` dans `translations` avec le contenu suivant

```
<?xml version="1.0"?>
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2">
  <file source-language="en" datatype="plaintext" original="file.ext">
    <body>
      <trans-unit id="Invalid credentials.">
        <source>Invalid credentials.</source>
        <target>Le mot de passe est invalide</target>
      </trans-unit>
      <trans-unit id="Email could not be found.">
        <source>Email could not be found.</source>
        <target>Email non-trouvé</target>
      </trans-unit>
    </body>
  </file>
</xliff>
```

Pour se déconnecter

essayez la route `/logout`

© Achref EL M...

# Symfony

Pour se déconnecter

essayez la route `/logout`

Question

Comment rediriger vers la page d'authentification ?

**Allez à la section** `logout` **de** `security.yaml`

```
logout:  
  path: app_logout  
  # where to redirect after logout  
  # target: app_any_route
```

© Achref EL ME

**Allez à la section** `logout` **de** `security.yaml`

```
logout:
  path: app_logout
  # where to redirect after logout
  # target: app_any_route
```

**Décommentez la clé** `target` **et ajoutez la route**

```
logout:
  path: app_logout
  # where to redirect after logout
  target: app_login
```

## Pour interdire l'accès à une page : deux solutions possibles

- soit en configurant la section `access_control` dans `security.yaml`
- soit dans le contrôleur
- soit en utilisant la fonction `is_granted()` dans la vue

# Symfony

## Pour interdire l'accès à tout utilisateur non-authentifié

```
access_control:
  - { path: '^/login', roles: IS_AUTHENTICATED_ANONYMOUSLY }
  - { path: '^/*', roles: [IS_AUTHENTICATED_FULLY] }
```

© Achref EL MOU

# Symfony

## Pour interdire l'accès à tout utilisateur non-authentifié

```
access_control:
  - { path: '^/login', roles: IS_AUTHENTICATED_ANONYMOUSLY }
  - { path: '^/*', roles: [IS_AUTHENTICATED_FULLY] }
```

## Pour autoriser les utilisateurs qui ont le rôle admin (ROLE\_ADMIN)

```
access_control:
  - { path: '^/login', roles: IS_AUTHENTICATED_ANONYMOUSLY }
  - { path: '^/*', roles: [ROLE_ADMIN] }
```



# Symfony

## Remarques

- La clé `path` accepte les expressions régulières
- Le nom d'un rôle doit être écrit en majuscule
- Les mots composants le nom d'un rôle doivent être séparés par un underscore.
- La clé `roles` accepte une valeur ou un tableau de valeurs

# Symfony

**Pour restreindre l'accès aux routes du contrôleur `PersonneController` aux utilisateurs ayant le rôle `ROLE_ADMIN` ou `ROLE_USER`**

```
access_control:
  - { path: '^/personne', roles: [ROLE_USER, ROLE_ADMIN] }
```

© Achref EL MOU

# Symfony

**Pour restreindre l'accès aux routes du contrôleur `PersonneController` aux utilisateurs ayant le rôle `ROLE_ADMIN` ou `ROLE_USER`**

```
access_control:
  - { path: '^/personne', roles: [ROLE_USER, ROLE_ADMIN] }
```

En testant, le message d'erreur suivant est affiché

```
Passing more than one Security attribute to "Symfony\Component
\Security\Core\Authorization\AccessDecisionManager::decide()"
is not supported.
```

# Symfony

## Explication

- Bug dans la version 5 de **Symfony**
- Pour le corriger, il faut aller dans `vendor\symfony\security-core\Authorization\TraceableAccessDecisionManager.php`
- Cherchez la méthode `decide`
- Faites les modifications indiquées dans la slide suivante ou dans <https://github.com/symfony/symfony/commit/63984b013c92f5cd2373d81c19554b4270c4b776>

# Symfony

## Remplacez

```
public function decide(TokenInterface $token, array $attributes,  
    $object = null): bool
```

© Achref EL MOUELHI ©

# Symfony

## Remplacez

```
public function decide(TokenInterface $token, array $attributes,  
    $object = null): bool
```

## Par

```
public function decide(TokenInterface $token, array $attributes,  
    $object = null/*, bool $allowMultipleAttributes = false*/): bool
```

© Achref EL

# Symfony

## Remplacez

```
public function decide(TokenInterface $token, array $attributes,  
    $object = null): bool
```

## Par

```
public function decide(TokenInterface $token, array $attributes,  
    $object = null/*, bool $allowMultipleAttributes = false*/): bool
```

## Et

```
$result = $this->manager->decide($token, $attributes, $object);
```

# Symfony

## Remplacez

```
public function decide(TokenInterface $token, array $attributes,  
    $object = null): bool
```

## Par

```
public function decide(TokenInterface $token, array $attributes,  
    $object = null/*, bool $allowMultipleAttributes = false*/): bool
```

## Et

```
$result = $this->manager->decide($token, $attributes, $object);
```

## Par

```
$result = $this->manager->decide($token, $attributes, $object, 3 < \  
    func_num_args() && func_get_arg(3));
```



# Symfony

**Pour restreindre l'accès à une méthode de** `PersonneController` **aux utilisateurs authentifiés**

```
class PersonneController extends AbstractController
{
    /**
     * @Route("/personne/add", name="personne_add")
     */
    public function addForm(EntityManagerInterface
        $entityManager, Request $request)
    {
        $this->denyAccessUnlessGranted('
            IS_AUTHENTICATED_FULLY');
        // le reste du contenu
    }
}
```

# Symfony

**Pour restreindre l'accès à toutes les méthodes de**  
`PersonneController` **aux utilisateurs ayant le rôle** `ROLE_ADMIN`

```
/**
 *
 * @IsGranted("ROLE_ADMIN")
 */
class PersonneController extends AbstractController
{
    // le contenu
}
```

# Symfony

**Pour restreindre l'accès à une méthode de** `PersonneController`  
**aux utilisateurs ayant le rôle** `ROLE_ADMIN`

```
class PersonneController extends AbstractController
{
    /**
     * @IsGranted("ROLE_ADMIN")
     * @Route("/personne/add", name="personne_add")
     */
    public function addForm(EntityManagerInterface
        $entityManager, Request $request)
    {
        // le reste du contenu
    }
}
```

# Symfony

**Pour restreindre une partie de la vue aux utilisateurs ayant le rôle `ROLE_ADMIN` (contenu à ajouter dans `home/index.html.twig`)**

```
{% if is_granted('ROLE_ADMIN') %}
    <a href="{{ url('personne_add') }}">
        Ajouter une personne
    </a>
{% endif %}
```

# Symfony

## Pour récupérer l'utilisateur authentifié dans une méthode de contrôleur

```
class PersonneController extends AbstractController
{
    /**
     * @Route("/personne/add", name="personne_add")
     */
    public function addForm(EntityManagerInterface
        $entityManager, Request $request)
    {
        $this->denyAccessUnlessGranted('
            IS_AUTHENTICATED_FULLY');
        $user = $this->getUser();
        // le reste du contenu
    }
}
```

# Symfony

## Pour récupérer les rôles de l'utilisateur

```
class PersonneController extends AbstractController
{
    /**
     * @Route("/personne/add", name="personne_add")
     */
    public function addForm(EntityManagerInterface
        $entityManager, Request $request)
    {
        $this->denyAccessUnlessGranted('
            IS_AUTHENTICATED_FULLY' );
        $user = $this->getUser();
        roles = $user->getRoles();
        // le reste du contenu
    }
}
```

# Symfony

**Pour récupérer l'email de la personne authentifié (contenu à ajouter dans `personne/index.html.twig`)**

```
{% if is_granted('ROLE_ADMIN') %}
    <p>Email: {{ app.user.email }}</p>
{% endif %}
```

# Symfony

Dans `security.yaml`

```
security:
    # ...

    role_hierarchy:
        ROLE_ADMIN:          ROLE_USER
        ROLE_SUPER_ADMIN:    [ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]
```

## Remarques

- L'utilisateur ayant le rôle `ROLE_ADMIN` a aussi le rôle `ROLE_USER`
- L'utilisateur ayant le rôle `ROLE_SUPER_ADMIN` a aussi les rôles `ROLE_ADMIN`, `ROLE_USER` et `ROLE_ALLOWED_TO_SWITCH`