Symfony 6 : 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

Symfony

But de la sécurité

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

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 ?

Symfony

Configuration de la sécurité

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

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`

Symfony

Configuration de la sécurité

- En utilisant les annotations/attributs
- Et en définissant quelques règles dans `config/packages/security.yaml`
- 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 ne choisit pas la version complète à la création du projet

```
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
```

Symfony

Pour créer la table `User`, exécutez la commande

```
php bin/console make:migration
```

Ensuite

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

Symfony

Pour créer la table `User`, exécutez la commande

```
php bin/console make:migration
```

Ensuite

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

Ou

```
php bin/console d:m:m
```

Symfony

Pour remplir la table `User` avec des données aléatoires, installez le bundle de fixture

```
composer require --dev doctrine/doctrine-fixtures-bundle
```

Symfony

Pour remplir la table `User` avec des données aléatoires, installez le bundle de fixture

```
composer require --dev doctrine/doctrine-fixtures-bundle
```

Pour générer un fichier de données aléatoires

```
php bin/console make:fixtures

The class name of the fixtures to create
UserFixtures
```

Symfony

Contenu généré pour `UserFixtures`

```php
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(UserPasswordHasherInterface $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->hashPassword(
            $user,
            'wick'
        ));
        $manager->persist($user);
        $user2 = new User();
        $user2->setEmail('john@john.us');
        $user2->setPassword($this->passwordEncoder->hashPassword(
            $user2,
            'john'
        ));
        $manager->persist($user2);
        $manager->flush();
    }
}
```

Symfony

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

```
php bin/console doctrine:fixtures:load
```

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

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`

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 connexion

Symfony

Pour résoudre ce problème, il faut modifier la méthode `onAuthenticationSuccess` **définie dans** `src/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.fr.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="4">
                <source>Invalid credentials.</source>
                <target>Identifiants invalides.</target>
            </trans-unit>
        </body>
    </file>
</xliff>
```

Symfony

Pour modifier les messages d'erreurs de la page d'accueil, créez un fichier `security.fr.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="4">
                <source>Invalid credentials.</source>
                <target>Identifiants invalides.</target>
            </trans-unit>
        </body>
    </file>
</xliff>
```


Pour plus de détails

https://github.com/symfony/security-core/blob/6.1/Resources/translations/security.fr.xlf

Symfony

Pour se déconnecter

essayez la route `/logout`

Symfony

Pour se déconnecter

essayez la route `/logout`

Question

Comment rediriger vers la page d'authentification ?

Symfony

Allez à la section `logout` **de** `config/packages/security.yaml`

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

Symfony

Allez à la section `logout` de `config/packages/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
```

Symfony

Pour interdire l'accès à une page : trois 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: PUBLIC_ACCESS }
    - { path: '^/*', roles: [IS_AUTHENTICATED_FULLY] }
```

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 autoriser les utilisateurs qui ont le rôle admin (ROLE_ADMIN)

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

Symfony

Pour autoriser les utilisateurs qui ont le rôle admin (ROLE_ADMIN)

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


Remarques

- Connectez vous avec (wick@wick.us, wick) et vérifiez qu'il a accès à toutes les routes (y compris /personne/add) car il a le rôle admin.
- Connectez vous avec (john@john.us, john) et vérifiez qu'il a accès à toutes les routes sauf /personne/* car il n'a pas le rôle admin.

Symfony

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

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

Symfony

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

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

Remarques

- Connectez vous avec `(wick@wick.us, wick)` et vérifiez qu'il a accès à toutes les routes (y compris `/personne/add`) car il a le rôle admin.
- Connectez vous avec `(john@john.us, john)` et vérifiez qu'il a accès à toutes les routes (y compris `/personne/add`) car il a le rôle user.

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(...)
    {
        $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
{
    #[Route("/personne/add", name: "personne_add")]
    #[IsGranted('ROLE_ADMIN')]
    public function addForm(...)
    {
        // le 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 vérifier les droits d'accès dans un service ou n'importe quel autre fichier de l'application, on injecte `Security` **et on l'utilise**

```php
namespace App\Service;

use Symfony\Component\Security\Core\Exception\AccessDeniedException;
use Symfony\Component\Security\Core\Security;

class SalesReportManager
{

     public function __construct(private Security $security)
     {
     }

      public function doSomething()
      {

         if ($this->security->isGranted('ROLE_ADMIN')) {
             // ...
         }

          // ...
      }

      // ...
  }
```

Symfony

Pour récupérer l'utilisateur authentifié dans une méthode de contrôleur

```
class PersonneController extends AbstractController
{
    #[Route("/personne/add", name: "personne_add")]
    public function addForm(...)
    {
        $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 rôle `ROLE_ADMIN`, `ROLE_USER` et `ROLE_ALLOWED_TO_SWITCH`