Guide pour mettre en place l'authentification avec Symfony

Introduction

Protéger ses applications est essentiel. Ce guide montre comment ajouter un système de connexion / authentification à vos applications Symfony pour les rendre sûres.

Ce qu'il vous faut

Avoir installé :

- Symfony et être prêt à coder.
- Security-bundle avec la commande **composer require symfony/security-bundle.**
- Maker-bundle avec la commande **composer require --dev symfony/maker-bundle.**
- Avoir créé un utilisateur avec la commande **php bin/console make:user**

Mise en place

A. Premiers réglages ce fait dans le fichier security.yaml

Ce fichier contient les règles de sécurité :

- **Où vérifier ?** On dit à Symfony où et comment protéger l'accès.
- **Comment charger les infos ?** On indique comment trouver les infos sur les utilisateurs.
- **Comment garder les mots de passe secrets ?** On choisit la manière de cacher les mots de passe.

B. Configuration du fichier security.yaml

- **Créer le fichier security.yaml** : Tout d'abord, assurez-vous que le fichier security.yaml existe dans le répertoire config/packages de votre application Symfony.

1. Encodeurs de mot de passe et stockage des utilisateurs

```yaml
# config/packages/security.yaml
security:
    # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
        App\Entity\User:
            # Use native password encoder
            # This value auto-selects the best possible hashing algorithm
            # (i.e. Sodium when available).
            algorithm: auto
    # https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers
    providers:
        app_user_provider:
            entity:
                class: App\Entity\User
                property: username
        users_in_memory: { memory: null }
```

- **Ligne 5** la valeur ‘auto’ pour l’encodeur de mot de passe sélectionne automatiquement le meilleur encodeur disponible, soit bcrypt en ce moment.
- **Ligne 12** dans ‘providers’, on déclare que nos utilisateurs sont stockés dans la base de données, avec l’entité User. Notre base contiendra alors une table ‘user’ sur le modèle de cette entité
- **Ligne 16** on déclare que nos utilisateurs sont identifiés de façon unique par leur propriété ‘username’. Cela induit alors que la connexion d’un utilisateur se fera par le couple nom d’utilisateur et mot de passe

2. Les firewalls

```yaml
        firewalls:
            dev:
                pattern: ^/(_(profiler|wdt)|css|images|js)/
                security: false
            main:
                lazy: true
                provider: app_user_provider
                form_login:
                    login_path: login
                    check_path: login_check
                logout:
                    path: logout
                    target: homepage
```

On peut voir 2 firewall déclarés dans notre fichier de configuration :

- **Le firewall "dev"** est conçu pour s'assurer que les outils de développement Symfony restent accessibles lorsque l'application est en mode développement. Il garantit notamment que la barre d'outils Symfony reste visible.
- **Le firewall "main"** gère toutes les autres URL de notre application.

Nous avons également configuré plusieurs attributs pour ces pare-feux, notamment :

- **Le provider** qu' on retrouve “app_user_provider”, qui est déclaré plus haut dans le fichier. Il s'agit de l'entité User dans la base de données.
- **form_login et logout** qui sont les chemins de connexion et de déconnexion. Les utilisateurs se connectent via un formulaire accessible à l'adresse "/login" et se déconnectent via "/logout". Symfony gère automatiquement la redirection vers la page de connexion si un visiteur tente d'accéder à une ressource nécessitant une authentification.

3. Les rôles et les contrôles d’accès

```
40          access_control:
41          # - { path: ^/admin, roles: ROLE_ADMIN }
42          # - { path: ^/profile, roles: ROLE_USER }
```

- **La configuration access_control** permet de restreindre l'accès à certaines URL en fonction des rôles des utilisateurs. Par exemple, { path: ^/admin, roles: ROLE_ADMIN } restreint l'accès à toutes les URL commençant par "/admin" aux utilisateurs ayant le rôle "ROLE_ADMIN". Cela fonctionne en vérifiant le rôle de l'utilisateur lorsqu'il accède à une URL et en lui permettant ou non l'accès en fonction de ses autorisations.

Comment ça marche ?

Symfony utilise un système pour vérifier qui vous êtes :

- **Qui peut entrer ?** D'abord, le Security Bundle vérifie l'identité de la personne qui veut entrer. En l'occurrence, cela se fait par un nom d'utilisateur et un mot de passe.
- **Où peut-on aller ?** Ensuite, une fois que quelqu'un est reconnu, le Security Bundle regarde ce qu'il a le droit de faire ou de voir sur le site en fonction des rôles.
- **Garder un œil ouvert :** Le Security Bundle surveille aussi ce qui se passe sur le site pour s'assurer que tout le monde est là où il doit être.

Qui peut voir quoi ?

On définit des règles (rôles) pour dire qui a accès à quoi. Vous pouvez dire, par exemple, qui peut voir certaines parties du site. On peut le faire depuis les controllers ou directement dans

les fichiers twig. Dans les exemples ci-dessous, si l'utilisateur n'a pas le rôle ADMIN ces options déclencherait une erreur 403 (Accès refusé).

A. Dans les controllers

Avec l'attribut #[IsGranted] sur une méthode : **#[IsGranted('ROLE_ADMIN')]**
Avec la méthode denyAccessUnlessGranted : **denyAccessUnlessGranted("ROLE_ADMIN");**

B. Dans les fichiers twig

{% if is_granted("ROLE_ADMIN") %}{% endif %}

Où garder les infos des utilisateurs ?

Les infos peuvent être gardées de différentes manières, comme dans une base de données.

Conseils pour bien faire

- **Soyez secret :** utilisez HTTPS pour que personne ne puisse espionner vos connexions.
- **Cachez les mots de passe :** Gardez-les de manière sûre.
- **Testez votre système :** Assurez-vous qu'il fonctionne bien en le testant.

En résumé

Mettre en place l'authentification est crucial mais demande de bien comprendre comment tout fonctionne. Ce guide est là pour vous aider à démarrer. N'hésitez pas à expérimenter et à chercher plus d'infos dans la documentation de Symfony.

Pour aller plus loin

La documentation officielle de Symfony sur l'authentification.