Gestion de l’authentification, projet TodoAndCo

Sommaire

Introduction - principe général

L'authentification d'un utilisateur est une fonction essentielle de très nombreuses applications web, par conséquent il existe un composant spécifique pour Symfony permettant l'authentification des utilisateurs: le composant security. Ce composant permet de rapprocher les informations transmises dans un form avec une entité de base de donnée de manière à authentifier ou non un utilisateur. Il permettra également une gestion de rôles et l'accès ou non à certaines pages en fonction du rôle de l'utilisateur. Si vous souhaitez rentrer plus dans le détail concernant le composant security, n'hésitez pas à parcourir sa documentation officielle.

La configuration du composant security

La configuration du composant est accessible dans le fichier `config/packages/security.yaml` Ce fichier possède de nombreuses variables mais celles que vous aurez probablement à manipuler sont les suivantes:

a - providers

C'est grâce à cette variable que vous pouvez déclarer l'entité sur laquelle vous souhaitez baser votre authentification et l'attribut de cette entité qui sera le login. Dans notre cas nous utilisons l'entité user et plus particulièrement l'attribut username en tant que login

```yaml
providers:
    app_user_provider:
        entity:
            class: App\Entity\User
            property: username
```

b - firewalls

Les firewalls sont le cœur du moteur d'authentification de Symfony: ils traitent les requêtes et autorisent ou non l'accès aux pages de l'application en fonction de la personne authentifiée ainsi que les routes appelées pour le login et le logout.

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

c - access_control

La variable access_control définit les pages accessibles aux utilisateurs en fonction de leur rôle. C'est ce qui permet par exemple de restreindre l'accès à la page de gestion des utilisateurs uniquement aux administrateurs du site. Elle fait donc correspondre une regex de l'ensemble des pages devant être accessible à un rôle avec le rôle en question:

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

L'entité user

L'authentification de Symfony se base sur une entité pour effectuer l'authentification. Dans notre cas, comme vu dans le paragraphe sur la variable providers du fichier de config, il s'agit de l'entité User. Pour satisfaire aux exigences du moteur d'authentification de Symfony, cette dernière doit impérativement répondre à un certain nombre de critères comme par exemple le fait d'implémenter l'objet "userInterface" propre à Symfony.

La manière plus simple pour créer une classe user répondant aux exigences du moteur d'authentification est de laisser symfony se charger de la création de la classe user en utilisant le makerBundle de symfony et plus particulièrement la commande:

`php bin/console make:user`

C'est d'ailleurs la manière dont nous avons procédé sur ce projet.

Rôles en base de données

Les rôles attribuables aux utilisateurs de ToDoAndCo sont au nombre de deux:

- user
- admin

Pour chaque ligne de la table user de notre BDD, il existe un champ "roles" contenant un tableau contenant les valeurs ROLE_ADMIN ou ROLE_USER. C'est cette donnée qu'utilisera le moteur d'authentification de Symfony pour interpréter le type d'utilisateurs et par extension les pages auxquelles il a le droit d'accéder (grâce à la variable access_control du fichier de config du bundle security)

Le hash des mots de passe

Pour des raisons de sécurité, vous devez absolument enregistrer les mots de passe en base de données après les avoir "hashé". De cette façon, il est beaucoup plus difficile pour un hacker de récupérer les mots de passe de vos utilisateurs en cas de pénétration de la base de données.

Il est parfois utile pour de simples tests de créer manuellement des utilisateurs en base de données, si vous devez créer un mot de passe, vous pouvez obtenir sa version hashée en exécutant la commande suivante:

```
php bin/console security:hash-password <yourPassword>
```

Votre terminal vous renverra alors la valeur hashée de votre mot de passe et vous n'aurez plus qu'à l'enregistrer en BDD.

La toolbar symfony

La toolbar de symfony possède un onglet dédié à l'authentification des utilisateurs, elle permet de voir en un clin d'oeil si un utilisateur est authentifié ainsi que son login si c'est bien le cas. Elle détaille également le firewall qui a traité la requête http