Documentation technique

Sommaire :

1. User Entity

L'utilisateur correspond à l'entité **User** (AppBundle/Entity/User). Cette classe implémente **UserInterface,** indispensable pour la création d'une classe utilisateurs et l'accès aux différentes fonctions liées à la gestion d'un utilisateur.

```
# AppBundle/Entity/User.php
namespace AppBundle\Entity;

class User implements UserInterface
```

Doc Sf

2. Security components

2.1. Encoders

L'encodage utilisé pour *encoder* le mot de passe d'utilisateur est le *bcrypt*

```
# app/config/security.yml
encoders:
   AppBundle\Entity\User: bcrypt
```

Doc Sf

2.2. Providers

Le *provider* utilise la classe *User*. Ici on peut configurer :

- L'entité sélectionnée pour créer l'utilisateur
- La propriété pour l'identification de l'utilisateur (ici *username*)
- L'ORM sélectionnée pour la connexion à la BBD de celui-ci (*Doctrine*).

```
# app/config/security.yml
providers:
   doctrine:
       entity:
           class: AppBundle:User
           property: username
```

Doc Sf

2.3. Firewalls

Le pare-feu va permettre de donner l'accès ou non à certaines pages.
Par exemple, les pages d'authentification et de déconnexion.

- La config *dev* permet d'autoriser le chargement des pages des *assets* et du *template* de votre site, ainsi que le profiler de Symfony.
- La config *main* indique que les personnes anonymes peuvent accéder aux pages de login.
- La config *form_login* détermine les routes pour accéder au formulaire de connexion et de vérification des données.
- Le chemin par défaut est « / ».
- La config *logout :* ~ indique que tous les *users* peuvent se déconnecter.

```
# app/config/security.yml
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        anonymous: ~
        pattern: ^/
        form_login:
            login_path: login
            check_path: login_check
            always_use_default_target_path: true
            default_target_path: /
        logout: ~
```

Doc Sf

2.4. Access Control

Ici vous pouvez configurer l'accès de vos utilisateurs à différentes routes en fonction de leurs *rôles*.

```
# app/config/security.yml
access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/users/create, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/users, roles: ROLE_ADMIN }
    - { path: ^/users, roles: [ROLE_ADMIN, ROLE_USER] }
```

Doc Sf

3. User Voter

La classe *Security/UserVoter* vérifie d'abord que l'*user* est une instance de *UserInterface,* ensuite que seulement les utilisateurs ayant un rôle d'ADMIN ($user->isAdmin()) accèdent et gèrent le CRUD utilisateurs.
Les *users* de type ADMIN peuvent effectuer les opérations suivantes :

- ADD : ajouter un nouvel utilisateur
- GET : récupérer la liste des utilisateurs
- EDIT : modifier les données d'un utilisateur
- REMOVE : effacer un utilisateur

Doc Sf