Documentation technique

Sommaire :

1. Entity User

L'utilisateur correspond à l'entité **User** (src/Entity/User). Cette classe implémente le **UserInterface** indispensable pour la création d'une classe utilisateurs et l'accès au différents fonction lié à la gestion d'un utilisateur.

```
# src/Entity/User.php
namespace App\Entity;

class User implements UserInterface
```

2. Component security

2.1. Encoders

L'encodage utilisé pour encoder le mot de passe d'utilisateur est le sha512

```
# config/packages/security.yaml
encoders:
   App\Entity\User: sha512
```

2.2. Providers

Le providers fournie le classe utilisateur, on peut configurer ici:
- l'entité sélectionnée pour créer l'utilisateur
- le proprité pour l'identification de l'utilisateur,
- ainsi que orm sélectionné pour la connection a la BBD de celui-ci.

```
# config/packages/security.yaml
providers:
   doctrine:
       entity:
           class: App\Entity\User
           property: username
```

2.3. Firewalls

Le pare feu va permettre de donner l'accès ou non à certain page, exemple : page d'authentification, page de déconnection.
La config **dev** permet d'autoriser le chargement des pages d'assets et du template de votre site ainsi que le profiler de symfony.
la config main indique que les personnes annonyme peuvent accéder à la page de login, logout et celle-ci utilise guard authenticators.

```yaml
# config/packages/security.yaml
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        pattern: ^/
        anonymous: true
        guard:
            authenticators:
                - App\Security\LoginFormAuthenticator
        logout:
            path: logout
```

2.4. Access Control

Ici vous pouvez configurer l'accès de vos utilisateurs en fonction de leur rôle à différentes route.

```yaml
# config/packages/security.yaml
access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/users, roles: ROLE_ADMIN }
    - { path: ^/, roles: ROLE_USER }
```

2.5. Role Hierarchy

Role hierarchy vous permet d'organiser l'héritage entre de chaque roles, ici par exemple Admin hérite de User.

```
# config/packages/security.yaml
role_hierarchy:
   ROLE_ADMIN: [ROLE_USER]
```