

# Documentation technique



## Sommaire :

1. Entity User
2. Component security
  - 2.1. Encoders
  - 2.2. Providers
  - 2.3. Firewalls
  - 2.4. Access Control
  - 2.5. Role Hierarchy
3. Login Form

## 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 aux différentes fonctions liées à la gestion d'un utilisateur.

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

class User implements UserInterface
```

[Doc Sf](#)

## 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
```

[Doc Sf](#)

### 2.2. Providers

Le providers fournit la classe utilisateur, on peut configurer ici:

- l'entité sélectionnée pour créer l'utilisateur
- la propriété pour l'identification de l'utilisateur,
- ainsi que l'ORM sélectionnée pour la connexion à la BDD de celui-ci.

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

[Doc Sf](#)

## 2.3. Firewalls

Le pare feu va permettre de donner l'accès ou non à certaines pages, 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 anonymes peuvent accéder à la page de login, logout et celle-ci utilise guard authenticators.

```
# 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
```

[Doc Sf](#)

## 2.4. Access Control

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

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

[Doc Sf](#)

## 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]
```

[Doc Sf](#)

## 3. Login Form

La classe **LoginFormAuthenticator** traite les informations transmises par le formulaire de connexion `/login` .

Si les informations sont valides, elle redirige l'utilisateur authentifié vers la dernière page choisie ou la page d'accueil `'homepage'` .

```
namespace App\Security;

class LoginFormAuthenticator extends
AbstractFormLoginAuthenticator
```

[Doc Sf](#)