

DOCUMENTATION TECHNIQUE

Projet 8 Reprenez et améliorez un projet existant



Ressources : [Documentation Framework Symfony](#)

Documentation technique

Cette documentation se destine aux prochains développeurs juniors qui rejoindront l'équipe dans quelques semaines.

1. Mise en place de l'authentification

Pour gérer le processus d'authentification, il est nécessaire de comprendre les fichiers suivants :

- `src/Controller/SecurityController.php`
 - gère la page d'authentification (qui contient le formulaire)
 - gère le processus d'authentification : chargement de l'utilisateur + vérification du mot de passe
- `config/packages/security.yaml`
 - définit, dans la partie providers, l'entité utilisée pour l'authentification et la propriété utilisée comme identifiant (ici le username)

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

→ définit quel hachage est utilisé pour l'authentification (ici automatique)

```
security:
    enable_authenticator_manager: true
    # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
```

→ restreint l'accès à certaines pages ou groupes de pages en définissant des rôles : `Role_User` ou `Role_Admin`

```
access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/users, roles: ROLE_ADMIN }
    - { path: ^/users/^, roles: ROLE_ADMIN }

    # allow unauthenticated users to access the login form
    - { path: ^/login, roles: PUBLIC_ACCESS }
    - { path: ^/, roles: ROLE_USER }
```

- `src/Entity/User.php`
 - L'entité authentifiée qui est générique à la création du système d'authentification de la librairie Security-bundle et qui implémente `UserInterface`. Elle est obtenue avec un `make user`
- `templates/security/login.html.twig`
 - Vue du formulaire de connexion

2. Fonctionnement de l'authentification

Page connexion :

Dans un premier temps, L'utilisateur saisit ses identifiants sur la page d'authentification (/login). Pour accéder à cette page, la méthode loginAction est exécutée, sur le Contrôleur : SecurityController. Cette méthode sert à générer la page, et à envoyer des données à la vue avec un render.

Soumission du formulaire :

Quand le formulaire est validé par l'utilisateur, les données sont récupérées par la classe AuthenticationUtils, qui se charge de l'authentification qui s'opère comme suit:

- Récupération des champs du formulaire grâce à la méthode loginCheck
- Récupération de l'utilisateur dans la BDD via son username, en passant par la vérification du password associé à l'utilisateur
- Une fois les informations d'identification de l'utilisateur vérifiées, l'utilisateur est chargé ainsi que ses droits pour autoriser ou refuser l'accès aux pages selon le niveau d'autorisation défini pour chaque page avec le système de Rôle.

Redirection de l'utilisateur :

Selon si l'utilisateur s'est bien authentifié ou pas, la méthode LoginAction le redirigera vers la page d'accueil. Sinon, un message d'erreur sera affiché au-dessus du formulaire de connexion.

```
form_login:
  login_path: login
  check_path: login_check
  always_use_default_target_path: true
  default_target_path: /
  logout: ~
```

Consulter [FormLogin](#)

3. Stockage des utilisateurs

Les données utilisateurs sont stockées dans la base de données MySQL dans la table "user" qui contient les champs suivants :

* Nom d'utilisateur : le champ username est unique et il est possible de rajouter un autre champ unique. Il faudra simplement, dans l'entité user : src/entity/user, rajouter « unique = true » à l'annotation @Column

```
/**
 * @ORM\Column(type="string", length=25, unique=true)
 * @Assert\NotBlank(message="Vous devez saisir un nom d'utilisateur.")
 */
private $username;
```

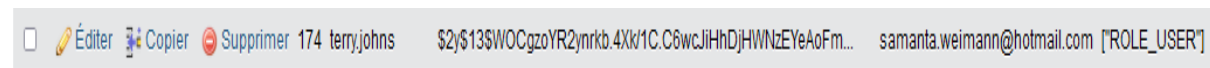
* e-mail

* mot de passe : il est encrypté dans la base de donnée, pour plus de sécurité. L'encryptage est en mode automatique. On peut changer le mode dans le fichier de configuration « security.yam » dans le bloc « security »

* rôles : relatif aux droits de l'utilisateur.

Ceci est géré dans le fichier « security.yam » au bloc du Contrôle des accès aux différentes pages du site. Il est possible de changer le rôle de l'utilisateur.

Voici une entrée dans la table user. On peut voir que le password est bien encrypté



id	username	email	password	role
174	terry.johns	\$2y\$13\$WOCgzoYR2ynrkB.4Xk/1C.C6wcJiHhDjHWNzEYeAoFm...	samanta.weimann@hotmail.com	["ROLE_USER"]

Pour plus d'information consulter [Symfony Security](#)

4. Collaboration

Pour contribuer à la documentation en respectant les bonnes pratiques de git, il faudra créer une branche par fonctionnalité à développer ou sinon une branche par développeur au minimum. Cela évitera qu'un développeur modifie le même fichier qu'un autre il y aura alors un conflit de fichiers.

A consulter le document CONTRIBUTING.md