Authentification et Sécurité

L'application Todolist est basée sur le Framework Symfony 3.1, veuillez-vous rapprocher de la documentation référence sur la sécurité https://symfony.com/doc/3.1/security.html

Le système de sécurité de Symfony s'organise essentiellement sur trois points :

- L'authentification
- Les autorisations
- L'utilisateur authentifié (session)

Fichier de configuration

Le système de sécurité est configuré dans **app/config/security.yml** et quelques explications sur les clés sont nécessaires.

- security.firewalls

Pour déclarer les différents pares-feux.

- security.firewalls.dev

Est le pare-feu qui permet à la barre de profilage de fonctionner sans contrainte de sécurité.

- security.firewalls.main

Est le pare-feu principal qui s'applique sur tous les URL (pattern : ^/)

- security.firewalls.main.form_login

Utilisation d'un formulaire d'authentification pour le pare-feu principal.
https://symfony.com/doc/3.1/security/form_login_setup.html

- security.firewalls.main.logout

Symfony désauthentifiera l'utilisateur actuelle.
https://symfony.com/doc/3.1/security.html#logging-out

- security.access_control

Permet de sécuriser plus finement des URL à l'aide de rôle
https://symfony.com/doc/3.1/security.html#securing-url-patterns-access-control

- security.role_hierarchy

Définit la hiérarchie des rôles
https://symfony.com/doc/3.1/security.html#hierarchical-roles

- security.encoders

Définit l'algorithme d'encodage du mot de passe de l'objet User (UserInterface)
https://symfony.com/doc/3.1/security.html#c-encoding-the-user-s-password

- security.providers.doctrine.entity

Déclare le fournisseur des utilisateurs sécurisés. Dans notre cas une entité Doctrine.
https://symfony.com/doc/3.1/security/entity_provider.html#configure-security-to-load-from-your-entity

- *L'application*

Il est nécessaire d'avoir un objet UserInterface **src\AppBundle\Entity\User.php**
https://symfony.com/doc/3.1/security/entity_provider.html#create-your-user-entity

- *Page d'authentification*

A l'emplacement **src\AppBundle\SecurityController::loginAction** pour l'action et une vue **app\Ressources\views\security\login.html.twig** avec deux noms de champs **_username** et **_password** définis par défaut dans la configuration.

La validation de l'authentification (login_check) et la déconnexion est géré par le composant sécurité de Symfony en s'appuyant les paramètres du fichier de configuration **app\config/security.yml**

- *Déconnexion*

Le composant de sécurité gère cette fonctionnalité. Il faut toutefois renseigner le fichier de configuration pour le pare-feu principal security.main.logout et une route « /logout » dans **src\app\config\routing.yml**
https://symfony.com/doc/3.1/security.html#logging-out

- *Contrôle d'accès d'une action contrôleur*

Pour contrôler l'accès d'une action en fonction du rôle de l'utilisateur authentifié, il faut utiliser l'annotation

@Security("has_role('ROLE_ADMIN')")

De la classe Sensio\Bundle\FrameworkExtraBundle\Configuration\Security
https://symfony.com/doc/current/bundles/SensioFrameworkExtraBundle/annotations/security.html

Cette manière de procéder ne doit pas être une généralité. Le fichier de configuration (security.access_control) déclare un premier niveau de sécurité sur toutes les URL avec ROLE_USER exigée.

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

Pour l'action TaskController::deleteAction un autre type de contrôle d'accès.

@Security("**task.isAuthor(user)**")

Vérifie que l'utilisateur courant (user) est bien l'auteur de la tâche (task). isAuthor est une méthode de l'entité src\AppBundle\Entity\User .php

- *Sécuriser l'accès d'un élément dans une vue Twig*

{% is_granted(…) %}
http://symfony.com/doc/3.1/reference/twig_reference.html#is_granted