Authentification

Le système de sécurité Symfony comprend deux mécanismes: l'authentification et l'autorisation.

L'authentification décidera qui est l'utilisateur. Soit il n'est pas identifié, et il est anonyme. Soit il est identifié et il est dans ce cas un membre authentifié. Ce processus est géré par le pare-feu (firewall).

L'autorisation décidera si un visiteur a le droit d'avoir l'accès à certaines ressources. Cette étape intervient après le pare-feu (firewall) et est gérée par Symfony via le contrôle d'accès.

```
firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
```

Définir nos utilisateurs.

Les utilisateurs de notre application sont représentés par la classe AppBundle: User, qui implémente l'interface utilisateur. Ce sont des entités Doctrine, représentées par l'attribut nom d'utilisateur.

Pour chiffrer le mot de passe de nos utilisateurs dans notre base de données, nous devons définir dans le fichier de sécurité le codeur qui sera utilisé. Il se trouve sous la clé `encoders`, dans notre cas, nous utilisons l'encodeur bcrypt.

```
security:
    encoders:
        AppBundle\Entity\User: bcrypt
```

Toutes nos URL sont gérées par le firewall `main`. Sous cette clé, nous pourrons définir certains paramètres, tels que le système d'authentification utilisé ou la route vers laquelle les utilisateurs souhaitant accéder à une ressource protégée devront être redirigés.

```
main:
            anonymous: ~
            pattern: ^/
            form_login:
                login_path: login
                check_path: login_check
```

```
            always_use_default_target_path:  true
            default_target_path:  /
        logout: ~
```

Une fois l'utilisateur authentifié, il est accessible dans un contrôleur via la `$this->getUser()`méthode ou dans une vue Twig via la vue globale `app.user`.

Définir si un utilisateur est authentifié

Par défaut, la sécurité Symfony est utilisée pour définir si un utilisateur est authentifié.
`IS_AUTHENTICATED_ANONYMOUSLY` tous les utilisateurs, même ceux qui ne se sont pas authentifiés
`IS_AUTHENTICATED_FULLY`: utilisateurs authentifiés pendant la session en cours.
Ces attributs peuvent être utilisés dans le fichier security.yml pour sécuriser les modèles d'URL via l'access_control.
Il est également possible d'utiliser ces attributs dans un contrôleur, via la `$this->get('security.authorization_checker')->isGranted('IS_AUTHENTICATED_FULLY')`méthode ou à partir d'une vue Twig`{% if is_granted ('IS_AUTHENTICATED_FULLY')%}`

Attribuer des droits aux utilisateurs

Il est possible de déterminer nos propres rôles et de les attribuer à nos utilisateurs en fonction des droits que nous souhaitons leur attribuer.
Le nom du rôle est libre, cependant, il doit toujours commencer par ROLE_ pour que la sécurité de Symfony le reconnaisse. Dans notre application, nous pouvons attribuer à un utilisateur un ROLE_USER et / ou ROLE_ADMIN, en fonction des droits que l'on souhaite lui accorder.

Seuls les utilisateurs de ROLE_ADMIN peuvent accéder à / users * routes:

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

Ces rôles peuvent être utilisés pour sécuriser des modèles d'URL (sous la `access_control` clé dans le fichier security.yml), une action du contrôleur (à l'aide du `security.authorization_checker` service)

```
>get('security.authorization_checker')->isGranted('ROLE_ADMIN'))
```

Lorsque les rôles sont définis, il est possible d'hériter des rôles. Pour cela, nous utilisons la `role_hierarchy` clé du fichier security.yml.
Dans notre cas, ROLE_ADMIN aura également les droits ROLE_USER.

```
  access_control:
```

```
        - { path: ^/users/create, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/tasks, roles: ROLE_USER }
        - { path: ^/, roles: IS_AUTHENTICATED_ANONYMOUSLY }
```

Les utilisateurs sont chargés depuis la base de données:

```
doctrine:
            entity:
                class: AppBundle:User
                property: username
```