L’authentification sur le site Todolist

L’authentification est basée sur le composant ‘security’ de symfony, la documentation complète du composant est disponible ici : <http://symfony.com/doc/current/security.html>

L’authentification comporte plusieurs fonctionnalités :

* La connexion et la vérification des identifiants de connexions
* La déconnexion
* La création et la modification d’un utilisateur
* La vérification des droits

Les utilisateurs pouvant se connecter au site sont stockés dans la table user

La table est définie comme suit :

* id : qui représent l’identifiant unique de l’utilisateur dans le site
* username : nom de l’utilisateur
* password : mot de passe crypté de l’utilisateur
* email : son mail
* roles : défini les droits de l’utilisateur sur le site.

Dans le système, les utilisateurs sont gérés par l’entité doctrine : AppBundle\Entity\User.

## Les instructions pour connaitre les utilisateurs connectés

L’instruction suivante permet de connaître à tout moment l’utilisateur connecté

$this->container->get('security.token\_storage')

->getToken()->getUser();

L’instruction ‘twig’ équivalente:

app.user

Pour tester les droits de l’utilisateur connecté:

$this->get('security.authorization\_checker')

->isGranted('ROLE\_ADMIN')

L’instruction ‘twig’ équivalente :

is\_granted("ROLE\_ADMIN")

## La connexion

La route ‘/login’ permet d’afficher le formulaire de connexion.

Si des modifications sont à faire elles doivent être faites ici :

SecurityController::loginAction

Et ici :

app/Resources/views/security/login.html.twig

La route /login\_check permet de valider la connexion.

A priori rien à faire ici : la validation de la connexion est géré par le composant symfony.

## La déconnexion

Pour se déconnecter il suffit de faire un ‘GET’ sur la route /logout.

## La création et la modification d’un utilisateur

Pour créer un nouvel utilisateur il suffit de créer une instance de la classe  ‘User’ et enregistrer cette instance en base de données, en utilisant ‘entityManager’ du composant doctrine.

Actuellement, c’est le contrôleur ‘UserController’ qui se charge de la création et de l’édition des utilisateurs.

L’action ‘createAction’ reliée à la route ‘users/create’ permet de créer un nouvel utilisateur.

Il faut créer une instance $user de classe User

$user = new User();

Il faut instancier un formulaire de saisie relié à l’instance $user

$form = $this->createForm(UserType::class, $user);

Si le formulaire de saisie est validé

$form->handleRequest($request);

if ($form->isValid()) {

$em = $this->getDoctrine()->getManager();

Scriptage du mot de passe saisi

$password = $this->get('security.password\_encoder')

->encodePassword($user, $user->getPassword());

$user->setPassword($password);

Gestion des rôles associés à l’utilisateur

$role = $form->get('roles')->getData();

$roles = [];

foreach ($role as $key => $value) {

$roles[] = $value;

}

$user->setRoles($roles);

L’action se termine par la sauvegarde en base de données du nouvel utilisateur :

$em->persist($user);

$em->flush();

$this->addFlash('success', "L'utilisateur a bien été ajouté.");

return $this->redirectToRoute('user\_list');

}

Affichage de la page de connexion :

return $this->render('user/create.html.twig', ['form' => $form->createView()]);

L’action ‘editAction’ reliée à la route /users/{id}/edit permet de modifier les rôles d’un utilisateur déjà existant.

On instancie un formulaire de saisi prérempli .

       $form = $this->createForm(UserEditType::class, $user);

Si le formulaire est validé

$form->handleRequest($request);

if ($form->isValid()) {

Les rôles saisis

$role = $form->get('roles')->getData();

$roles = [];

foreach ($role as $key => $value) {

$roles[] = $value;

}

$user->setRoles($roles);

Enregistrement des modifications

$this->getDoctrine()->getManager()->flush();

$this->addFlash('success', "L'utilisateur a bien été modifié");

return $this->redirectToRoute('user\_list');

}

Affichage de la page d’édition d’un utilisateur :

return $this->render('user/edit.html.twig', ['form' => $form->createView(), 'user' => $user]);

}

## La configuration de la sécurité

Le fichier security.yml contient le paramétrage de la sécurité du site

security:

encoders:

Définition du cryptage utilisé pour crypter les mots de passe

AppBundle\Entity\User: bcrypt

Paramétrage de la classe php sur laquelle se base l’authentification

providers:

doctrine:

entity:

class: AppBundle:User

property: username

…

firewalls:

dev:

pattern: ^/(\_(profiler|wdt)|css|images|js)/

security: false

main:

anonymous: ~

pattern: ^/

Paramétrage du formulaire de login

form\_login:

login\_path: login

check\_path: login\_check

always\_use\_default\_target\_path: true

default\_target\_path: /

logout: ~

Paramétrage des règles d’accès au site

access\_control:

Les routes /login… sont accessibles à tout le monde

- { path: ^/login, roles: IS\_AUTHENTICATED\_ANONYMOUSLY }

Les routes /users sont accessibles uniquement aux utilisateurs ayant un rôle admin

- { path: ^/users, roles: ROLE\_ADMIN }

Les routes / sont accessibles uniquement aux utilisateurs ayant un rôle user

- { path: ^/, roles: ROLE\_USER }