

Formation Symfony 4

4ème séance



Symfony

Retour sur TP



- A ne pas faire :

```
<div><a href="Serie/{ serie.id }">Détails</a></div>
```

Préférez utiliser `{{ path('serie', {'id':serie.id}) }}` (on redirige vers une route avec paramètre et on utilise pas de chemin relatif)

- Utiliser des noms de routes cohérents (vous vous y retrouverez bien mieux lorsque les lignes de code vont s'accumuler)
- Utiliser le camelCase pour le nom des variables (ex: `$maVariable`)

```
/**  
 * @ORM\Column (type="string", length=255)  
 */  
private $Nom;
```

A ne pas reproduire !

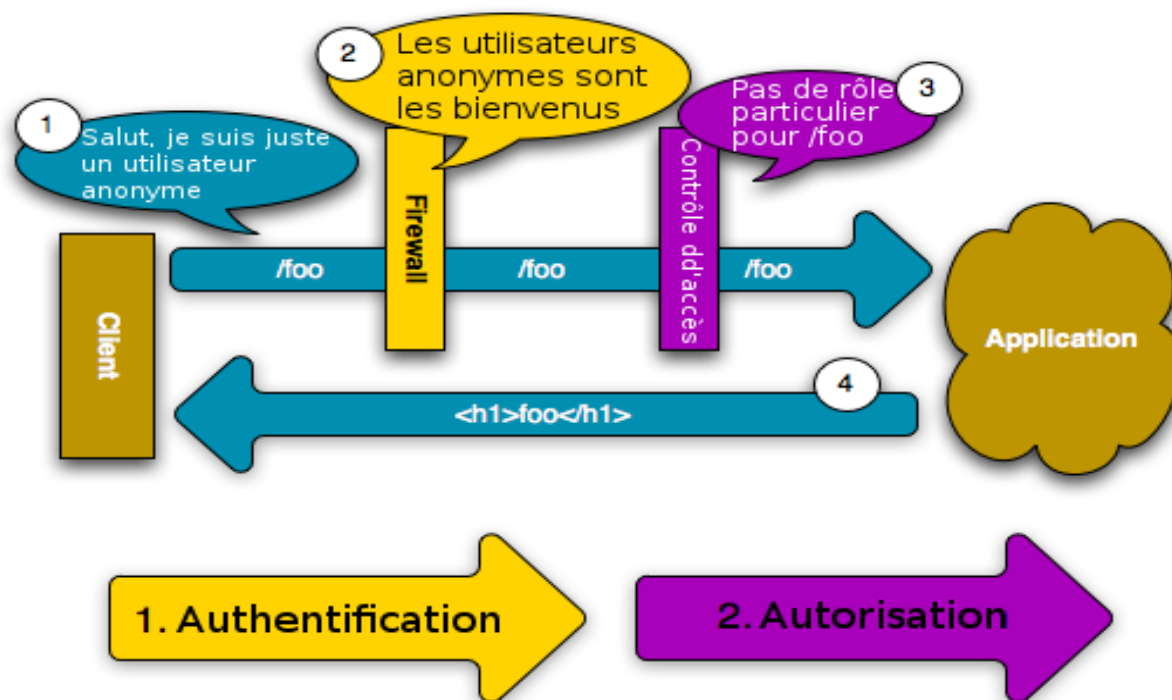
La sécurité



- **\$ composer require security**
- La sécurité est un ensemble de règles qui attribue plus ou moins de liberté aux utilisateurs de votre application
- La sécurité passe par un système d'authentification des utilisateurs de votre application et vous permet de leur donner des autorisations (droits)
- Les configurations sur la sécurité du projet sont définies de manière générale dans le `security.yaml` dans **`/config/packages`**

La sécurité

- L'authentification est le processus permettant de savoir qui on est en tant que visiteur. C'est le rôle du Firewall de gérer ça.
- L'autorisation est le processus qui va déterminer si vous avez le droit d'accéder à la ressource demandée (il agit après le firewall qui peut être vu comme un filtre).



User provider



- Lorsqu'un utilisateur soumet un nom d'utilisateur et un mot de passe, la couche d'authentification demande au fournisseur d'utilisateur (user provider) configuré de renvoyer un objet utilisateur pour un nom d'utilisateur donné. Symfony vérifie ensuite si le mot de passe de cet utilisateur est correct et génère un jeton (token) de sécurité pour que l'utilisateur reste authentifié pendant la session en cours.
- **Créer une entité User qui implémente UserInterface et Serializable (interfaces)**
- Donner lui des attributs dont username, email, password (mot de passe crypté), plaintextpassword (mot de passe visible)
- Cela nous servira à enregistrer des utilisateurs en bdd mais aussi à charger des utilisateurs lorsque l'authentification sera réussie.

Par défaut

security.yaml

Logged in as anon.

Authenticated **Yes**

Token class AnonymousToken

Firewall name main



anon.



28 ms

```
security:
  providers:
    in_memory: { memory: ~ }
  firewalls:
    dev:
      pattern: ^/(_(profiler|wdt)|css|images|js)/
      security: false
    main:
      anonymous: ~
```

Les providers sont des « fournisseurs d'utilisateurs », ils ont pour but de récupérer les utilisateurs et de les charger.

Types de fournisseurs

- memory => utilise les utilisateurs définis dans le fichier de configuration

Exemple de rendu:

Ouvrir une session

http://localhost:8000

Nom d'utilisateur

Mot de passe

- entity => utilise une entité pour fournir les utilisateurs (ex: User)
- id => permet d'utiliser un service quelconque en tant que fournisseur (WebService par exemple)

UserType

```
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
        ->add( child: 'name', type: TextType::class)
        ->add( child: 'lastname', type: TextType::class)
        ->add( child: 'email', type: EmailType::class)
        ->add( child: 'username', type: TextType::class)
        ->add( child: 'plainPassword', type: RepeatedType::class, array(
            'type' => PasswordType::class,
            'first_options' => array('label' => 'Password'),
            'second_options' => array('label' => 'Repeat Password'),
        ))
    ;
}
```


Récupérer l'utilisateur automatiquement



```
public function loadUserByUsername($username)
{
    try {
        return $this->createQueryBuilder ( alias: 'u' )
            ->where ( predicates: 'u.username = :username OR u.email = :email' )
            ->setParameter ( key: 'username', $username )
            ->setParameter ( key: 'email', $username )
            ->getQuery ()
            ->getOneOrNullResult ();
    } catch ( NonUniqueResultException $e ) {
    }
}
```

register.html.twig

```
{% extends 'base.html.twig' %}

{% block body %}

    {{ form_start(form) }}
    {{ form_row(form.name) }}
    {{ form_row(form.lastname) }}
    {{ form_row(form.username) }}
    {{ form_row(form.email) }}
    {{ form_row(form.plainPassword.first) }}
    {{ form_row(form.plainPassword.second) }}

    <button type="submit">Register!</button>
    {{ form_end(form) }}

{% endblock %}
```

Dans l'entité User :

```
* @return (Role|string)[] The user roles
*/
public function getRoles()
{
    return array('ROLE_USER'); //chaque nouvel utilisateur crée aura le rôle USER
}
```

Créer un utilisateur

```
/**
 * @Route("/newUser", name="newUser")
 */
public function createUser(Request $request, UserPasswordEncoderInterface $passwordEncoder)
{
    $user = new User();
    $form = $this->createForm(type: UserType::class, $user);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $password = $passwordEncoder->encodePassword($user, $user->getPlainPassword());
        $user->setPassword($password);

        $em = $this->getDoctrine()->getManager();
        $em->persist($user);
        $em->flush();
        return $this->redirectToRoute(route: 'home');
    }

    return $this->render(view: 'user/newUser.html.twig', [
        'form' => $form->createView(),
    ]);
}
```

login.html.twig

```
{% extends 'base.html.twig' %}

{% block body %}
    <form action="{{ path('login_check') }}" method="post">
        <label for="username">Login :</label>
        <input type="text" id="username" name="_username" />

        <label for="password">Mot de passe :</label>
        <input type="password" id="password" name="_password" />
        <br />
        <input type="submit" value="Connexion" />
    </form>
{% endblock %}
```

Vérifier le login

```
/**
 * @Route("/loginCheck", name="login_check")
 */
public function loginCheck()
{
    if ($this->get('security.authorization_checker')->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
        return $this->redirectToRoute(route: 'home');
    }
    else{
        return new Response(content: "Authentification ratée");
    }
}
```

Déconnexion

```
/**
 * @Route("/logout", name="logout")
 */
public function logout() {
    //pas besoin de retourner quelque chose lorsque l'utilisateur
    //se déconnecte, il suffit de préciser le target dans security.yaml
    $this->get('security.token_storage')->setToken(null);
    $this->get('request')->getSession()->invalidate();
}
```

Paramétrer les Firewalls

```
security:
  encoders:
    App\Entity\User:
      algorithm: bcrypt # chiffrage
  providers:
    provider1: #nom du provider
      entity: #provider lié à une entité
      class: App\Entity\User
      property: username
  firewalls:
    visiteur:
      anonymous: true
      pattern: ^/login$
      provider: provider1

    main:
      anonymous: false #utilisateurs non authentifiés sont redirigés vers le login
      pattern: ^/ #tout ce qui suit cette url doit passer le firewall
      provider: provider1 #définir quel provider est utilisé pour ce firewall
      form_login:
        login_path: login #route vers le login
        check_path: login_check #route vers la vérification du login
      logout:
        path: logout #route vers la déconnexion
        target: login #route retournée après la déconnexion
  dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security: false
```

Processus d'authentification

Login : azef

Mot de passe :

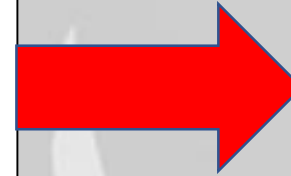
Connexion

Logged in as anon.

Authenticated **Yes**

Token class AnonymousToken

Firewall name login



Logged in as

Authenticated **Yes**

Token class UsernamePasswordToken

Firewall name main

Récupérer l'utilisateur

- `$user = $this->getUser();`

Si `$user == null` alors la requête n'est pas derrière un pare-feu ou l'utilisateur courant est anonyme

Si `$user` est une instance de `User` alors l'utilisateur est derrière le pare-feu et authentifié

- On peut accéder aux attributs de l'utilisateur sur twig via la superglobale : `{{ app.user }}`

Ex: `{{ app.user.username }}`


Accès de contrôle et Autorisation





- On peut limiter l'accès a certaines pages selon le rôle de l'utilisateur
- On définit des rôles (voir une hiérarchie de rôles)
- La configuration de l'autorisation se fait au cas par cas suivant les ressources : sécuriser une méthode de controller, un affichage twig ou une URL

Exemple

localhost:8000/admin (authentifié avec ROLE_USER)

 Symfony Exception

 Symfony Docs

 Symfony Support

[AccessDeniedException](#) > [AccessDeniedHttpException](#)

HTTP 403 Forbidden

Access Denied.



```
role_hierarchy:
```

```
    ROLE_ADMIN:          ROLE_USER
    ROLE_SUPER_ADMIN:    ROLE_ADMIN
```

```
access_control: #définition des droits (rôles) pour pouvoir accéder aux urls
```

- { path: ^/register\$, role: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/login\$, role: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/admin, roles: ROLE_ADMIN }
- { path: ^/, roles: ROLE_USER }

Exemple 2

- Sécuriser une méthode de controller avec l'annotation :

```
/**
```

```
* @Security("has_role('ROLE_AUTEUR') and has_role('ROLE_AUTRE')")
```

```
* /
```

- Depuis un vue twig : `{% if is_granted('ROLE_ADMIN') %}....{% endif %}`
- Depuis le controller : `if ($this->get('security.authorization_checker')->isGranted('ROLE_ADMIN')){ }`

FOSUserBundle



- Bundle permettant de gérer les utilisateurs et le système d'authentification (connexion et inscription) facilement. Il nous provisionne toutes les routes, il ne faut plus que les personnaliser (en surchargeant les méthodes et vues du bundle).
- Possibilité de confirmer l'inscription par mail et bien plus encore...
- Documentation:
<https://symfony.com/doc/current/bundles/FOSUserBundle/index.html>



Projet : Site web

Implémentation du système de gestion des utilisateurs et de la sécurité



- Système de connexion/déconnexion + inscription
- 3 rôles à définir : l'utilisateur qui peut voir des films et séries, l'annonceur qui peut poster des films et séries, et enfin l'admin qui a tous les droits (ceux mentionnés avant + gérer les autres utilisateurs)
- Une partie gestion du profil (changer ses infos pour un utilisateur)