

# AUTHENTIFICATION

## 1. Le firewall sous Symfony

L'authentification pour un système informatique est un processus permettant au système de s'assurer de la légitimité de la demande d'accès faite par un utilisateur et ça afin d'autoriser l'accès à certaines ressources du système. dans notre application créer avec le framework symfony c'est le firewall qui ce charge de l'authentification de notre utilisateur le visiteur doit être un membre authentifié pour accéder à l'ensemble des pages de notre application, sinon le firewall le redirigera automatiquement sur la page de connexion qui est au passage la seule partie de l'application qui n'est pas gerer par le systeme de firewall.

```
security:
    # https://symfony.com/doc/current/security/experimental_authenticators.html
    enable_authenticator_manager: true
    password_hashers:
        App\Entity\User:
            algorithm: auto

    # https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers
    providers:
        # used to reload user from session & other features (e.g. switch_user)
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            lazy: true
            provider: app_user_provider
            custom_authenticator: App\Security\CustomAuthenticator
            logout:
                path: app_logout
                # where to redirect after logout
                # target: app_any_route

            # activate different ways to authenticate
            # https://symfony.com/doc/current/security.html#firewalls-authentication

            # https://symfony.com/doc/current/security/impersonating_user.html
            # switch_user: true

    # Easy way to control access for large sections of your site
    # Note: Only the *first* access control that matches will be used
    access_control:
        - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/users, roles: ROLE_ADMIN }
        - { path: ^/tasks, roles: ROLE_USER }
```

voici le fichier qui gère le firewall de notre application (security.yml)

## 2. la class d'utilisateur User

Dans notre application les utilisateurs sont une instance de la class User, c'est le système d'autentification de symfony par default. Comme vous pouvez le remarquer la class User implémente le UserInterface qui implémente lui même une configuration pour nos utilisateur comme par exemple le systeme d'encodage des mot de passe ou encore l'access control qui permet de gerer accés à nos routes par role d'utilisateur nous verrons ça un peu plus en detail dans les pages suivante de ce document technique.

```
security:
    # https://symfony.com/doc/current/security/experimental_authenticators.html
    enable_authenticator_manager: true
    password_hashers:
        App\Entity\User:
            algorithm: auto
```

type d'encodage des mots de passe (security.yml)

### 3. L'encoders

l'encodeur est un objet qui encode le mot de passe de notre utilisateur. dans notre application nous utilisons l'algorithme d'encrypage bcrypt (algorithme:auto dans la nouvelle version de symfony) qui est une fonction de hachage créée par Niels Provos et David Mazières. Elle est basée sur l'algorithme de chiffrement

```
security:  
    # https://symfony.com/doc/current/security/experimental\_authenticators.html  
    enable_authenticator_manager: true  
    password_hashers:  
        App\Entity\User:  
            algorithm: auto
```

exemple de configuration du type d'encrypatage (security.yml)

### 4. Le Providers

un provider est un fournisseur d'utilisateurs. le firewall de notre application s'adressent aux providers pour récupérer les utilisateurs et les identifiants via une propriété unique. Dans notre application par exemple, nous allons identifier notre utilisateur via son username qui est unique mais, nous pourrions le faire via son email qui serait bien plus pertinent dans notre cas mais, nous verrons ça un peu plus tard dans l'audit de notre application.

```
8     providers:  
9         doctrine:  
10            entity:  
11                class: AppBundle:User  
12                property: username
```

exemple de configuration du provider (security.yml)

### 5. Les firewalls

Lors du processus d'autentification l'utilisateur utilise la page de login (formulaire de connexion). l'utilisateur peut ensuite renseigner son nom d'utilisateur et son mot de passe il est ensuite transmis vers le check\_path pour vérifier l'identification de l'utilisateur en cas d'erreur d'authentification le système reinvite l'utilisateur à s'authentifier et dans le cas d'un succès créer un token d'accès unique et permet l'utilisateur d'utiliser l'application.

```
firewalls:  
    dev:  
        pattern: ^/(_(profiler|wdt)|css|images|js)/  
        security: false  
  
    main:  
        anonymous: ~  
        pattern: ^/  
        form_login:  
            login_path: login  
            check_path: login_check  
            always_use_default_target_path: true  
            default_target_path: /  
        logout: ~
```

exemple de configuration du firewalls (security.yml)

## 6. le formulaire de connexion

le formulaire de connexion est definie dans le contrôleur **SecurityController** via la methode **login()** c'est l'arborecence est celle par default dsur le framework **symfony**, le système de sécurité est gerer automatiquement via au gestionnaire d'évènements de **symfony**. en cas de succès, le visiteur sera authentifié. Dans le cas contraire le systeme renverra celui-ci vers le formulaire de connexion pour le réinvité a rentrer ces informations d'authentification.

```
class SecurityController extends AbstractController
{
    /**
     * @Route("/login", name="app_login")
     * @CodeCoverageIgnore
     */
    public function login(AuthenticationUtils $authenticationUtils): Response
    {
        if ($this->getUser()) {
            return $this->redirectToRoute('homepage');
        }

        // get the login error if there is one
        $error = $authenticationUtils->getLastAuthenticationError();
        // last username entered by the user
        $lastUsername = $authenticationUtils->getLastUsername();

        return $this->render('security/login.html.twig', ['last_username' => $lastUsername, 'error' => $error]);
    }

    /**
     * @Route("/logout", name="app_logout")
     * @CodeCoverageIgnore
     */
    public function logout()
    {
        throw new \LogicException('This method can be blank - it will be intercepted by the logout key on your firewall.');
    }
}
```

route login (SecurityController.php)

## 7. le rendu du formulaire (vue twig)

```
1  {% extends "base.html.twig" %}%
2
3  {% block body %}
4      {% if error %}
5          <div class="alert alert-danger" role="alert">{{ error.messageKey|trans(error.messageData, 'security') }}</div>
6      {% endif %}
7
8      <form action="{{ path('login_check') }}" method="post">
9          <label for="username">Nom d'utilisateur :</label>
10         <input type="text" id="username" name="_username" value="{{ last_username }} />
11
12         <label for="password">Mot de passe :</label>
13         <input type="password" id="password" name="_password" />
14
15         <button class="btn btn-success" type="submit">Se connecter</button>
16     </form>
17  {% endblock %}
18
```

vue twig pour la procedure d'autentification de l'utilisateur via formulaire de connexion (login.html.twig)