



P8 - TODOLIST

**DOCUMENTATION TECHNIQUE**

# **AUTHENTICATION**

# Franck Lebeau

## PARCOURS PHP / SYMFONY

**S**ymfony est un framework PHP populaire, conçu pour développer des applications robustes et évolutives. L'authentification des utilisateurs est une fonctionnalité essentielle, et Symfony nous fournit des outils puissants pour gérer ce processus de manière efficace et sécurisée.

## LES FICHIERS CONCERNÉS

Dans Symfony, l'implémentation de l'authentification implique généralement la création / modification de plusieurs fichiers principaux :

1. Le fichier [security.yaml](#)
2. Une entité [User](#)
3. Un [formulaire](#)

## PAS À PAS

Symfony nous fournit un composant de sécurité. Si il n'est pas déjà installé, on peut l'installer via composer :

```
composer require symfony/security-bundle
```

On peut alors créer notre entité User :

```
php bin/console make:user
```

On accepte l'option de hachage du mot de passe lors de la création de l'entité.

Une fois la création terminée, notre entité User est créée, et accessible à : [src/Entity/User.php](#)

En général, on implémente une contrainte d'unicité sur le mail :

```
#[ORM\Column(type: 'string', length: 180, unique: true)]
private ?string $email;
```

**Important :** Symfony nous a créé un champs « Roles », qui nous servira plus tard à attribuer les rôles « Utilisateurs » et « Administrateurs ».

Vous pouvez ajouter / modifier votre entité, lui ajouter des champs, et procéder aux migrations vers votre base de donnée avec :

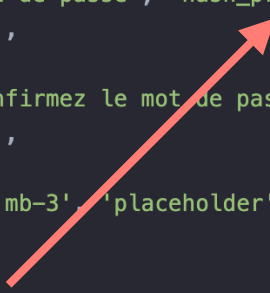
```
$ php bin/console make:migration
$ php bin/console doctrine:migrations:migrate
```

On peut ensuite créer notre formulaire d'inscription avec Symfony Form :

```
php bin/console make:form
```

Dans notre cas, nous avons implémenté le hachage du mot de passe à cette étape, grâce au puissant outil [FormBuilderInterface](#) :

```
53     ->add('plainPassword', RepeatedType::class, [
54         'type' => PasswordType::class,
55         'first_options' => ['label' => 'Mot de passe', 'hash_property_path' => 'password',
56             'class' => 'd-block text-start'],
57     ],
58     'second_options' => ['label' => 'Confirmez le mot de passe', 'attr' => ['class' =>
59         'class' => 'd-block text-start'],
60     ],
61     'attr' => ['class' => 'form-control mb-3', 'placeholder' => "Votre mot de passe"],
62     'required' => true,
63     'mapped' => false
64     ])
```



On peut ensuite créer / configurer son contrôleur qui s'occupera du Login / Logout :

- On utilise l'utilitaire d'authentification fourni par Symfony :  
Symfony\Component\Security\Http\Authentication\AuthenticationUtils
- On écrit une fonction logout pour se déconnecter

Ci-dessous notre implémentation pour TodoList :

```

11  #[Route(name: 'app_')]
    8 references | 0 implementations
12  class SecurityController extends AbstractController
13  {
14      #[Route('/login', name: 'login')]
        16 references | 0 overrides
15      public function login(Request $request, AuthenticationUtils $authenticationUtils): Response
16      {
17          $error = $authenticationUtils->getLastAuthenticationError();
18          $lastUsername = $authenticationUtils->getLastUsername();
19
20          return $this->render('security/login.html.twig', [
21              'error' => $error,
22              'lastUsername' => $lastUsername
23          ]);
24      }
25
26      #[Route('/logout', name: 'logout')]
        8 references | 0 overrides
27      public function logout()
28      {
29
30      }
31  }

```

On poursuit en créant le contrôleur qui s'occupera de l'enregistrement des nouveaux utilisateurs, dans notre cas le [UserController](#).

Notre route [/users/create](#), implémente la logique d'inscription, en récupérant les données du formulaire, les valide, puis elle les enregistre en base de données.

Ci-dessous notre implémentation pour TodoList de la fonction [createUser](#) :

```
#[Route('/users/create', name: 'app_user_create')]
16 references | 0 overrides
public function createUser(Request $request, EntityManagerInterface $entityManager): Response
{
    $user = new User();
    $form = $this->createForm(RegisterUserType::class, $user);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $entityManager->persist($user);
        $entityManager->flush();
        $this->addFlash('success', "Nouvel utilisateur créé");
        return $this->redirectToRoute('app_user_list');
    }

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

On peut enfin configurer notre pare-feu de sécurité, dans le fichier [security.yaml](#) (à config/packages/security.yaml) :

On commence par paramétrer notre [app\\_user\\_provider](#) avec l'email de l'utilisateur :

```
6      providers:
7          # used to reload user from session & other features (e.g. switch_user)
8          app_user_provider:
9              entity:
10                 class: App\Entity\User
11                 property: email
```

On continue avec la configuration de notre pare-feu :

```
12     firewalls:
13         dev:
14             pattern: ^/(_(profiler|wdt)|css|images|js)/
15             security: false
16         main:
17             lazy: true
18             provider: app_user_provider
19             form_login:
20                 login_path: app_login
21                 check_path: app_login
22             logout:
23                 path: app_logout
```

Pour terminer, nous pouvons paramétrer les **restrictions d'accès** au site. Dans le cadre de TodoList :

- Seul les membres peuvent accéder à leur liste de tâches (/tasks)
- Seul les administrateurs peuvent accéder aux parties du site concernant les utilisateurs (/users)

```
33     access_control:  
34         - { path: ^/users, roles: ROLE_ADMIN }  
35         - { path: ^/tasks, roles: ROLE_USER }
```

En complément, nous avons ajouté sur nos routes UserController des annotations isGranted (via Symfony\Component\Security\Http\Attribute\IsGranted), pour sécuriser plus précisément nos routes.

```
#[IsGranted('ROLE_ADMIN')]  
#[Route('/users', name: 'app_user_list')]
```

Pour résumer, nous avons :

1. Créé notre entité User
2. Créé un formulaire sécurisé
3. Créé nos contrôleurs, routes et vues concernant l'inscription et la connexion
4. Configuré notre fichier security.yaml et sécurisé l'accès à nos routes

 **En suivant ces étapes, il vous est possible de configurer une base solide pour l'authentification de votre site Symfony.**