

# **Documentation Technique**

## **Sommaire**

Entité	2
Security.yaml	3
Création de l'utilisateur	4
Login/Logout	5

# Entité

Tout d'abord il faut créer l'entité que l'on souhaite connecter ou déconnecter de l'application. Le nom n'est pas important mais il faut implémenter la classe « `UserInterface` »(1).

De plus, il faut créer les fonctions `getter` et `setter` de la variable `$roles`.

Il faut aussi ajouter les fonctions `getSalt()` et `eraseCredentials()` qui vont intervenir dans l'authentification.

Le chemin de `User` : `ToDoList/src/Entity/User.php`

```
src > Entity > User.php > User
1  <?php
2
3  namespace App\Entity;
4
5  use Doctrine\Common\Collections\ArrayCollection;
6  use Doctrine\Common\Collections\Collection;
7  use Doctrine\ORM\Mapping as ORM;
8  use Symfony\Component\Validator\Constraints as Assert;
9  use Symfony\Component\Security\Core\User\UserInterface;
10 use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
11
12 /**
13  * @ORM\Entity(repositoryClass="App\Repository\UserRepository")
14  * @UniqueEntity("email")
15  */
16 class User implements UserInterface 1
17 {
18     /**
19      * @ORM\Id()
20      * @ORM\GeneratedValue()
21      * @ORM\Column(type="integer")
22      */
23     private $id;
24
25     /**
26      * @ORM\Column(type="string", length=25, unique=true)
27      * @Assert\NotBlank(message="Vous devez saisir un nom d'utilisateur.")
28      */
29     private $username;
30
31     /**
32      * @ORM\Column(type="string", length=180, unique=true)
33      * @Assert\NotBlank(message="Vous devez saisir une adresse email.")
34      * @Assert\Email(message="Le format de l'adresse n'est pas correcte.")
35      */
36     private $email;
37
38     /**
39      * @ORM\Column(type="json")
40      */
41     private $roles = [];
42
43     /**
44      * @var string The hashed password
45      * @ORM\Column(type="string")
46     */
47     private $password;
```

```
85
86     /**
87      * @see UserInterface
88      */
89     public function getRoles(): array
90     {
91         $roles = $this->roles;
92         // guarantee every user at least has ROLE_USER
93         $roles[] = 'ROLE_USER';
94
95         return array_unique($roles);
96     }
97
98     public function setRoles(array $roles): self
99     {
100         $this->roles = $roles;
101
102         return $this;
103     }
104 }
```

```
119
120     /**
121      * @see UserInterface
122      */
123     public function getSalt()
124     {
125         // not needed when using the "bcrypt" algorithm in security.yaml
126     }
127
128     /**
129      * @see UserInterface
130      */
131     public function eraseCredentials()
132     {
133         // If you store any temporary, sensitive data on the user, clear it here
134         // $this->plainPassword = null;
135     }
```

# Security.yaml

Le fichier security.yaml doit être configuré afin de mettre en place l'authentification.

La partie encoders indique la manière dont le mot de passe va être chiffré.

La partie providers permet à l'application de savoir quel attribut de la class doit être vérifié. Dans le cas de cette application le visiteur doit se connecter à l'aide de l'username, mais il est possible de modifier cela afin de devoir se connecter avec l'adresse email.

Le role\_hierarchy définit les différents rôles de l'application afin de limiter l'accès à certains rôles.

Concernant les firewalls ils indiquent le chemin pour la connexion, la déconnexion ainsi que la route à suivre après avoir fait une de ces actions.

L'access\_control limite l'accès de certaines routes à des utilisateurs qui n'ont pas le rôle indiqué. Par exemple pour cette application, uniquement les utilisateurs avec le rôle 'ROLE\_ADMIN' peuvent gérer les utilisateurs liés aux chemins « /users »

Le chemin : [ToDoList/config/package/security.yaml](#)

```
config > packages > ! security.yaml
1  security:
2      encoders:
3          AppBundle\Entity\User: bcrypt
4          App\Entity\User:
5              algorithm: auto
6
7
8      providers:
9          app_user_provider:
10             entity:
11                 class: App\Entity\User
12                 property: username
13
14     role_hierarchy:
15         ROLE_USER: ROLE_USER
16         ROLE_ADMIN: [ROLE_USER, ROLE_ADMIN]
17
18     firewalls:
19         dev:
20             pattern: ^/(_(profiler|wdt)|css|images|js)/
21             security: false
22
23         main:
24             anonymous: ~
25             pattern: ^/
26             form_login:
27                 login_path: login
28                 check_path: login_check
29                 always_use_default_target_path: true
30                 default_target_path: /
31             logout: ~
32
33     access_control:
34         - { path: ^/tasks/, roles: ROLE_USER }
35         - { path: ^/users/, roles: ROLE_ADMIN }
```

# Création de l'utilisateur

La fonction de création d'un utilisateur va créer un utilisateur en base de donnée. Les informations de cet utilisateur seront comparées aux informations contenues dans le formulaire de connexion pour permettre la création d'une session.

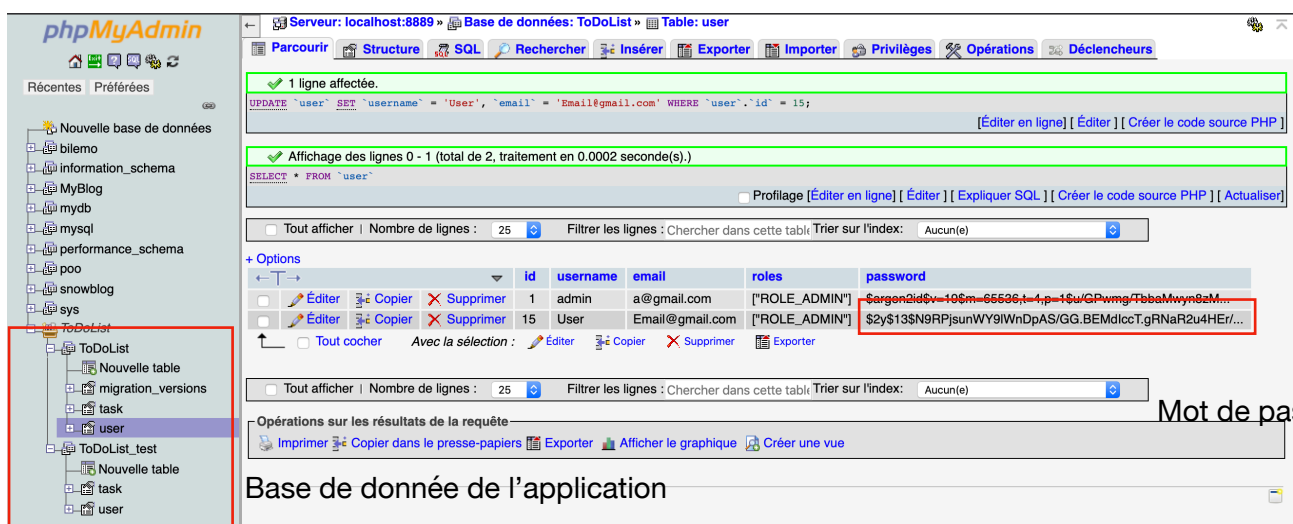
Dans cette fonction se trouve une ligne de code permettant de chiffrer le mot de passe afin que le mot de passe soit irrécupérable même si la base de donnée est accessible par des personnes extérieures.

Le chemin du contrôleur : `ToDoList/src/Controller/UserCreateController.php`

Le chemin du chiffrement de mot de passe : `ToDoList/src/Service/UserManager.php`

```
src > Controller > UserCreateController.php > UserCreateController > createAction
1 <?php
2
3 namespace App\Controller;
4
5 use App\Entity\User;
6 use App\Service\UserManager;
7 use Symfony\Component\HttpFoundation\Request;
8 use Symfony\Component\Routing\Annotation\Route;
9 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
10
11 class UserCreateController extends AbstractController
12 {
13     /**
14      * @Route("/users/create", name="user_create")
15      */
16     public function createAction(Request $request, UserManager $userManager)
17     {
18         $user = new User();
19
20         $form = $userManager->form($user, $request);
21
22         if ($form->isSubmitted() && $form->isValid()) {
23             $userManager->create($user);
24
25             $this->addFlash('success', 'L'utilisateur ajouté avec succès !');
26
27             return $this->redirectToRoute('user_list');
28         }
29
30         return $this->render('user/create.html.twig', ['formUser' => $form->createView()]);
31     }
32 }
33 }
```

```
40
41 public function encode($user)
42 {
43     $password = $this->encoder->encodePassword($user, $user->getPassword());
44
45     $user->setPassword($password);
46
47 }
```



Base de donnée de l'application

Mot de passe chiffré

# Login/Logout

Après la configuration du fichier security.yaml, la création de l'entité et la création d'un utilisateur en base de donnée il faut créer un controller contenant les fonctions de connexion et déconnexion. Il faut ajouter à cela un formulaire de connexion et un bouton déconnexion.

Le chemin du controller : `ToDoList/src/Controller/SecurityController.php`

Le chemin de la vue formulaire de connexion : `ToDoList/templates/security/login.html.twig`

Le chemin de la vue du bouton de déconnexion/connexion : `ToDoList/templates/base.html.twig`

```
src > Controller > SecurityController.php > SecurityController > loginAction
1  <?php
2
3  namespace App\Controller;
4
5  use Symfony\Component\Routing\Annotation\Route;
6  use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
7  use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;
8
9  class SecurityController extends AbstractController
10 {
11     /**
12      * @Route("/login", name="login")
13      */
14     public function loginAction(AuthenticationUtils $authenticationUtils)
15     {
16         $error = $authenticationUtils->getLastAuthenticationError();
17         $lastUsername = $authenticationUtils->getLastUsername();
18
19         return $this->render('security/login.html.twig', array(
20             'last_username' => $lastUsername,
21             'error'         => $error,
22         ));
23
24     /**
25      * @Route("/login_check", name="login_check")
26      */
27     public function loginCheck()
28     {
29         // This code is never executed.
30     }
31
32     /**
33      * @Route("/logout", name="logout")
34      */
35     public function logoutCheck()
36     {
37         // This code is never executed.
38     }
39 }
40 }
```

```
27 <body>
28 <nav class="navbar navbar-light navbar-fixed-top" style="background-color: #e3f2fd;" role="navigation">
29     <div class="container">
30         <div class="navbar-header">
31             <a class="navbar-brand" href="/">To Do List app</a>
32         </div>
33         
34     </div>
35 </nav>
36
37 <!-- Page Content -->
38 <div class="container">
39     <div class="row">
40         <a href="{{ path('user_create') }}" class="btn btn-primary">Créer un utilisateur</a>
41
42         {% if app.user %}
43         <a href="{{ path('logout') }}" class="pull-right btn btn-danger">Se déconnecter</a>
44         {% endif %}
45
46         {% if not app.user and 'login' != app.request.attributes.get('_route') %}
47         <a href="{{ path('login') }}" class="btn btn-success">Se connecter</a>
48         {% endif %}
49     </div>
50
51     <div class="row">
52         <div class="col-md-12">
53             {% for flash_message in app.session.flashBag.get('success') %}
54             <div class="alert alert-success" role="alert">
55                 <strong>Superbe !</strong> {{ flash_message }}
56             </div>
57             {% endfor %}
58
59             {% for flash_message in app.session.flashBag.get('error') %}
60             <div class="alert alert-danger" role="alert">
61                 <strong>Oops !</strong> {{ flash_message }}
62             </div>
63         </div>
64     </div>
65 </div>
```

```
templates > security > login.html.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 %}
```