Documentation développeurs Todolist *application*.


# Gérer vos authentifications.

. Login

. Créer de nouveau utilisateurs.

. Assigner des rôles aux utilisateurs.

. Autorisation des fonctionnalités.

---

**RAPPEL :**

L'authentification sous symfony se fait grâce au système de sécurité intégré au Framework.

Il faut pour cela lors de la création initiale du projet indiquer vouloir

la version –full et non le microservice.

## Powershell cmd :

$ symfony new <projectName> --full *pour un projet standard*

*Documentation officielle*

https://symfony.com/doc/current/setup.html

Sinon il faudra télécharger le bundle de security ultérieurement


voir les instructions détaillées sur la documentation officielle.

https://symfony.com/doc/current/security.html

**Il y à plusieurs façons de se connecter** dans notre cas on souhaite se connecter

Via un formulaire de login :

https://symfony.com/doc/current/security.html#form-login

## Comment se logger ?

Uri : /login

Routename : app_login

Controller : namespace App\Controller SecurityController ::class

---

Explication

Composent security de Symfony (System dauthentification)

Make auth loginformauthenicator

Login et logout

La logique d'authentification ne se trouve pas dans le controller  securitycontroller mais l' authenticator

**Bien comprendre le système d'authentification vous permettre**

**d'avoir une meilleur comprehension des outils et evitera de faire des erreurs d'aproximation.**

Src > Security > **LoginFormAuthenticator.php**

Depuis 2021 Symfony utilise une nouvelle Interface

use Symfony\Component\Security\Http\Authenticator\AbstractLoginFormAuthenticator;

**qui contient le cœur du système.**

Supports()

onAuthenticationFailure()

start()

# L'Authenticator.

Est le point d'entree du système son role est d'intercepter la requete et d'authentifier l'utilisateur

Avec un système de passeport

*LoginFormAuthenticator.php*

class LoginFormAuthenticator extends AbstractLoginFormAuthenticator

La fonction Authenticité(…)

```
public function authenticate(Request $request): Passport
```

Processus simplifie.

Vérification de la requête est ce quelle est supporte et si la route login est appeler

L'email est extrait de la requête et un passeport est généré

```
$request->getSession()->set(Security::LAST_USERNAME, $email);
```

```
        return new Passport(
            new UserBadge($email),
            new PasswordCredentials($request->request->get('password', '')),
            [
                new CsrfTokenBadge('authenticate', $request->request->get('_csrf_token')),
            ]
        );
```

Ce passeport à besoin de 3 parametres (badges)

Qui sont l'email le mot de pass et un token.

## La fonction onAuthenticationSuccess(…)

```
 public function onAuthenticationSuccess(Request $request, TokenInterface
$token, string $firewallName): ?Response
 {
     if ($targetPath = $this->getTargetPath($request->getSession(),
$firewallName)) {
     return new RedirectResponse($targetPath);
        }
        return new RedirectResponse($this->urlGenerator-
>generate('task_list'));
    }
```

La fonction

---

**Passeport.** Vas passer par un listener de security

**Provider.**

**Users checker.** Valider lutilisateur droit ou non de se connecter.

**AccessDeniedhandler.** Gere le type d'Erreur d'acces

### Le formulaire de login.

View : src > templates > security > login.html.twig

Input email value="{{ last_username }}"

Input password

Input _csrf_token value="{{ csrf_token('authenticate') }}"

---

Explication

Voir CSRF Protection in Login Forms

https://symfony.com/doc/current/security.html#csrf-protection-in-login-forms

## Comment créer un nouvel utilisateur dans la bdd ?

**Il nous faut une classe User qui pourra selon le principe du PPO**

**Instancier de nouvel objet à la demande new User().**

### Le Framework Symfony nous dispense d'une commande Maker très utile.

Symfony make :auth

Cette instruction va vous permettre de générer l'entité User

Celuici implémente les services **UserInterface** ainsi que **PasswordAuthentificatedUserInterface**

Cella vas nous permettre de gagner du temps sur la saisie des imports, namespace ect..

Après il faudra modifier et compléter cette classe à votre convenance mais elle respectera les contraintes lie au processus d'authentification mis en place par Symfony.

## RAPPEL :

Ce Maker doit être si vous avez suivi toutes les instructions disponibles

Dans votre projet si cela n'est pas le cas je vous conseille de revoir depuis le

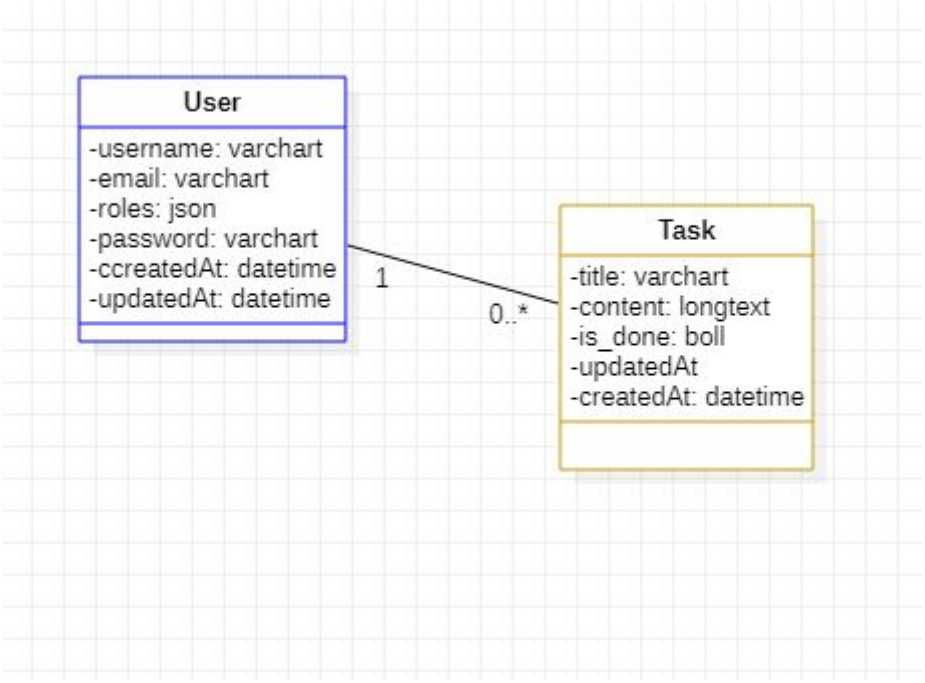
Début de l'installation sinon il faudra le rajouter manuellement

Composer **req maker**

# L’ entité User

Class (constraints)

Table




Referer vous à la class User pour decouvrir toute les contraintes mises en place dans

Les annotations des attributs.

Src > Entity > User.php

```php
#[ORM\Column(type: 'string', length: 25, unique:true)]
#[Assert\NotBlank(message:'Vous devez saisir un nom d\'utilisateur.')]
#[Assert\Length(min: 3, max: 20,
    minMessage:'Your first name must be at least {{ limit }}.',
    maxMessage:'Your first name cannot be longer than {{ limit }}.',
)]
private $username;
```

Ce projet requiert php 8.0.2 au minimum

"require": {"php": ">=8.0.2",

Vous allez retrouver toute les instructions pour installer le projet github directement dans votre machine locale en respectant toute les versions requie pour faire tourner ce projet.

https://github.com/Juju075/Todolist3


Composer.json & composer.lock

# Rappel :

## Composer.json

https://getcomposer.org/doc/01-basic-usage.md#composer-json-project-setup

## Composer.lock (Installation des dépendances)

https://getcomposer.org/doc/01-basic-usage.md#installing-dependencies