DOCUMENTATION TECHNIQUE POUR DEVELOPPEUR JUNIOR SYMFONY

Étape 1 :

Commencez par créer un dossier puis à y cloner le repository de l’application en tapant dans le CLI de votre IDE : git clone https://github.com/saro0h/projet8-TodoList.git

Étape 2 :

Puis, placez vous à la racine du projet.

Tapez : composer install

Tous les composants nécessaires au fonctionnement de l’application web vont s’installer.

Étape 3 :

Ensuite tapez : symfony serve

Vous devez voir ce message apparaitre :

[OK] Web server listening

The Web server is using PHP FPM 7.3.19

http://127.0.0.1:8000

Étape 4 :

Rendez vous sur cette adresse http://127.0.0.1:8000

On se retrouve sur la page suivante : <http://localhost:8000/login>

Dans le profiler, on peut voir la version de Symfony utilisé par l’application web.

Il s’agit de la version 3.1.6.

Étape 5 :

Commençons par upgrade cette version vers la version 3.4

A la racine du projet se trouve Composer.Json . Ouvrez le et dans la section require : Changez la version de Symfony/symfony par : 3.4.\*

Puis dans le CLI, tapez : composer update

Étape 6 :

Redemarrez le server et rendez vous sur <http://localhost:8000/login>

Dans le profiler, on peut voir que la version de symfony utilisé est désormais la 3.4.42

Notre but est d’arriver à la version 4 de Symfony.

Mais pour cela, nous devons corrigez les dépreciations.

Étape 7 :

Dans la barre du profiler de Symfony, cliquez sur le bouton pour voir la liste des dépréciations. Il y en a 11.

Étape 8:

Dépréciation  : User Deprecated: Symfony\Component\HttpKernel\Kernel::loadClassCache() is deprecated since Symfony 3.3, to be removed in 4.0.

On peut voir en cliquant sur Trace que la fonction qui est dépréciée se trouve dans web/app\_dev.php .

Ouvrez le fichier et supprimer la ligne 26 :

$kernel->loadClassCache();

Ouvrez aussi web/app.php et supprimez la ligne 10 :

$kernel->loadClassCache();

Si l’on refresh la page, on peut voir qu’il n’y a plus que 9 dépréciations.

Étape 9 :

Dépréciation : Using the unquoted scalar value "**!event**" is deprecated since Symfony 3.3 and will be considered as a tagged value in 4.0. You must quote it in "**/Users/jvjlondon/Desktop/SoutenanceP8/projet8-TodoList/app/config/config\_dev.yml**" on line 20.

Pour corriger celle ci, rendez vous dans app/config/config\_dev.yml et ajoutez des « » autour des valeurs des lignes 20 et 23 comme ceci :

channels: ["!event"]  
console:  
 type: console  
 channels: ["!event", "!doctrine"]

Étape 10 :

Dépréciations : The "**framework.trusted\_proxies**" configuration key has been deprecated in Symfony 3.3. Use the Request::setTrustedProxies() method in your front controller instead.

Pour corriger, celle ci rendez vous dans config/config.yml et supprimez la ligne 26 :

trusted\_proxies: ~

Étape 11 :

Dépréciations :

|  |  |
| --- | --- |
|  | Not setting "**logout\_on\_user\_change**" to true on firewall "**main**" is deprecated as of 3.4, it will always be true in 4.0.  Pour corriger celle ci, allez dans app/config/security.yml et ajoutez cette ligne la dans la section main :  logout\_on\_user\_change: true  Cela devrait être la ligne 17. |

Étape 12 :

Il reste 5 dépréciations

Pour corriger une partie d’entre elles, nous allons nous rendre sur cette adresse et comparer notre composer.json à celui de la version standard de Symfony 3.4

<https://github.com/symfony/symfony-standard/blob/3.4/composer.json>

Donc, nous allons changer ces bundles suivants :

"symfony/swiftmailer-bundle": "^2.6.4"

"symfony/monolog-bundle": "^3.1.0",

"sensio/framework-extra-bundle": "^5.0.0",

"sensio/distribution-bundle": "^5.0.19",

Ensuite à la racine du projet, dans le CLI, tapez : composer update

Étape 13 :

Plusieurs nouvelles dépréciations sont apparues, mais en contrepartie nous sommes à jour dans la version de nos bundles.

Voyons une autre dépréciations :

User Deprecated: The "**Sensio\Bundle\FrameworkExtraBundle\Configuration\Route**" annotation is deprecated since version 5.2. Use "**Symfony\Component\Routing\Annotation\Route**" instead.

Pour corriger celle-ci, rendez vous dans src/AppBundle/Controller et remplacez dans tous les controller :

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;

par :

use Symfony\Component\Routing\Annotation\Route;

Étape 14 :

Voyons une autre dépréciations :

The "**framework.trusted\_proxies**" configuration key has been deprecated in Symfony 3.3. Use the Request::setTrustedProxies() method in your front controller instead.

Pour la corriger, rendez vous dans un premier temps dans app/config/config.yml et supprimez la ligne 26 :

trusted\_proxies: ~

Puis allez dans web/app.php et ajoutez la ligne suivante :

Request::setTrustedProxies()

en dessous de la ligne 9 :

$kernel = new AppKernel('prod', false);  
Request::*setTrustedProxies*()

Étape 15 :

Voyons une autre dépréciations :

User Deprecated: Creating Doctrine\ORM\Mapping\UnderscoreNamingStrategy without making it number aware is deprecated and will be removed in Doctrine ORM 3.0.

Pour corriger celle-ci, rendez vous dans app/config/config.yml et changez la ligne 60 par :

naming\_strategy: doctrine.orm.naming\_strategy.underscore\_number\_aware

Étape 16 :

Voyons la dernière dépréciation :

|  |  |
| --- | --- |
|  | Enabling the "**sensio\_framework\_extra.router.annotations**" configuration is deprecated since version 5.2. Set it to false and use the "**Symfony\Component\Routing\Annotation\Route**" annotation from Symfony itself. |

Pour la corriger rendez vous dans app/config/config.yml et ajoutez ces lignes de codes à la fin :

sensio\_framework\_extra:  
 router:  
 annotations: false

Voila il n’y a plus aucune dépréciations.

Nous allons pouvoir upgrade la version de Symfony à la version 4.

Étape 17 :

Rendez vous dans composer.json et changez dans require :

"symfony/symfony": "^4.0",

Et aussi dans require-dev :

"symfony/phpunit-bridge": "^4.0"

Puis supprimer dans require :

"incenteev/composer-parameter-handler": "^2.0",

"symfony/swiftmailer-bundle": "",

Et supprimer aussi :

"sensio/distribution-bundle": "^5.0.19",

Et dans require-dev :

Supprimer :

"sensio/generator-bundle": "^3.0",

Ensuite dans le CLI, depuis la racine du projet tapez : composer update

Étape 18 :

Nous sommes maintenant sur Symfony 4 ! Super !

Nous allons pouvoir installer la structure Flex :

Son but ? **Automatiser l’installation et la suppression de vos dépendances** en fournissant une **configuration par défaut** sans avoir à aller lire la doc pour trouver quelle configuration écrire, quelles routes charger ou autre tâche rébarbative à effectuer. Et dès Symfony 4.0, Flex sera le moyen par défaut pour développer une application Symfony.

Commencez par vérifier que votre composer est à la derniere version possible.

Ensuite installez flex : composer require symfony/flex

Vidons désormais notre dossier vendor pour le reremplir avec et configurer sa librairie avec flex.

Dans le CLI, tapez : rm -rf vendor

Puis : composer install

Étape 19 :

Rendez vous dans composer.json et dans require supprimez :

Symfony/symfony

Et ajoutez :

"php": "^7.2",  
"symfony/console": "^4.0",  
"symfony/framework-bundle": "^4.0",  
"symfony/lts": "^4@dev",  
"symfony/yaml": "^4.0"

Puis dans require-dev, ajoutez :

"symfony/dotenv": "^4.0",

Puis tout en haut de composer.json, ajoutez les lignes suivantes :

"config": {  
 "preferred-install": {  
 "\*": "dist"  
 },  
 "sort-packages": true  
},

Puis remplacez la section scripts et extra par celles-ci :

"scripts": {  
 "auto-scripts": {  
 "cache:clear": "symfony-cmd",  
 "assets:install --symlink --relative %PUBLIC\_DIR%": "symfony-cmd"  
 },  
 "post-install-cmd": [  
 "@auto-scripts"  
 ],  
 "post-update-cmd": [  
 "@auto-scripts"  
 ]  
},  
"conflict": {  
 "symfony/symfony": "\*"  
},  
"extra": {  
 "symfony": {  
 "allow-contrib": true  
 }  
}

Remplacez les lignes suivantes aussi , pour autoload et autoload-dev

|  |
| --- |
|  |

"autoload": {  
 "psr-4": {  
 "App\\": "src/"  
 }  
},  
"autoload-dev": {  
 "psr-4": {  
 "App\\Tests\\": "tests/"  
 }  
},

Puis tapez dans le CLI, composer update .

Étape 20 :

Allez à l’adresse suivante :

<https://github.com/symfony/recipes/blob/master/symfony/console/4.4/bin/console>

Puis copiez le contenu et coller le dans notre projet dans bin/console .

Étape 21 :

Nous allons migrer tous le contenu de app/ vers la nouvelle strucuture symfony4 petit à petit.

On peut s’occuper de nouveaux composants pour commencer avec les formulaires et validator

Tapez dans le CLI :

composer require form

et

composer require validator

Grâce à flex tout est déjà configurer pour nous.

Passons à la suite.

Nous allons installer le composant de sécurité pour crsf\_protection

Tapez dans le CLI :

composer require security\_csrf

Étape 22 :

Ensuite occupons nous de twig :

Composer require twig

Un nouveau dossier template a été créer.

Migrons toutes nos anciennes vues dans ce dossier.

Pour cela, allez dans app/Resources/view, copiez tous les dossiers et collez les dans template et overwrite le base.html.twig .

Vous pouvez ensuite supprimer le dossier Resources.

Étape 23 :

Dans config/framework.yaml

Changez la ligne 14 par celle-ci :

fragments: ~

Puis voici le tour des asset, tapez dans le CLI :

Composer require asset

Étape 24 :

Allez dans src/AppBundle et copiez le contenu des dossiers Controller et Entity pour le copier dans src/Controller et src/Entity.

Bougez le dossier Form aussi dans src/

Puis supprimez src/AppBundle.

Étape 25 :

Changez tous les namespaces et les use contenant AppBundle par App dans les controllers et les entités ainsi que dans le dossier Form

Puis dans les controllers changer la classe extends par AbstractController ainsi que le use par

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

et supprimer celui qui n’est plus utilisé.

Étape 26 :

Migrons ensuite la sécurité en installant le bundle suivant :

Dans le CLI, tapez : composer require security

Copiez le contenu de app/config/security.yml et collez le dans config/security.yaml

En changeant par la suite la ligne 8 :

class: App:User

Étape 27 :

On va installer le nouveau web profiler avec la commande suivante, dans le CLI : composer require profiler

Étape 28 :

Dans config/routes.yaml, décommentez les 3 lignes :

index:  
 path: /  
 controller: App\Controller\DefaultController::index

Étape 29 :

Avec Flex, parameters.yaml n’existe pas, on référence plutôt des variables d’environnement.

On passe les variables de dev dans .env .

Renseignez la base de donnée dans le .env :

DATABASE\_URL=mysql://root:root@127.0.0.1:8889/projet8Todolist?serverVersion=5.7

Ici, j’utilise mysql , le user est root, le password root et le host : l27.0.0.1 avec le port 8889 et le nom de la base de donnée todolist.

Étape 30 :

Nous allons créer la database, en tapant la commande suivante dans le CLI,

./bin/console doctrine:database:create

Vous devez avoir la réponse suivante qui s’affiche :

Created database `projet8Todolist` for connection named default

Étape 31 :

Supprimez maintenant le dossier app/ .

Dans le dossier var, supprimez tout sauf le dossier cache et log

Ensuite Dans le dossier web/ : prenez css/ , img/, js/ et robots.txt et placez les dans le dossier public/

Supprimez le dossier web/

Puis tapez dans le CLI (si vous utilisez Apache) :

Composer require symfony/apache-pack

Ceci va créer un .htaccess dans le dossier /public

Étape 32 :

Vérifions qu’il n’y a plus de AppBundle dans nos fichiers en tapant la commande suivante dans le CLI :

git grep AppBundle

Dans TaskController et UserController , remplacez le AppBundle par App.

Puis dans le dossiers tests, bouger le dossier Controller à la racine de tests et supprimer le dossier AppBundle.

Le but du dossier tests est de refléter comme un miroir le dossier src/.

Puis dans test/Controller/DefaultControllerTest.php, remplacer AppBundle par App dans le namespace.

Étape 33 :

Rendez vous dans le dossier src/Repository et créons nos deux repository :

UserRepository.php

<?php  
  
namespace App\Repository;  
  
use App\Entity\User;  
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;  
use Doctrine\Persistence\ManagerRegistry;  
  
*/\*\*  
 \** ***@method*** *User|null find($id, $lockMode = null, $lockVersion = null)  
 \** ***@method*** *User|null findOneBy(array $criteria, array $orderBy = null)  
 \** ***@method*** *User[] findAll()  
 \** ***@method*** *User[] findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)  
 \*/*class UserRepository extends ServiceEntityRepository  
{  
 public function \_\_construct(ManagerRegistry $registry)  
 {  
 parent::*\_\_construct*($registry, User::class);  
 }  
}

Et

TaskRepository.php

<?php  
  
namespace App\Repository;  
  
use App\Entity\Task;  
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;  
use Doctrine\Persistence\ManagerRegistry;  
  
*/\*\*  
 \** ***@method*** *Task|null find($id, $lockMode = null, $lockVersion = null)  
 \** ***@method*** *Task|null findOneBy(array $criteria, array $orderBy = null)  
 \** ***@method*** *Task[] findAll()  
 \** ***@method*** *Task[] findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)  
 \*/*class TaskRepository extends ServiceEntityRepository  
{  
 public function \_\_construct(ManagerRegistry $registry)  
 {  
 parent::*\_\_construct*($registry, Task::class);  
 }  
}

Étape 34 :

Nous allons maintenant implémenter le nouveau Maker, tapez dans le CLI :

Composer require maker –dev

Nous allons nous servir de ce service pour créer notre système d’authentification, tapez dans le CLI :

Composer require make :auth

Choisissez Empty authenticator, puis appelez l’authenticator AppAuthenticator.

Un dossier et fichier src/Security/appAuthenticator.php vient d’être créer et config/packages/security.yaml vient d’être mis à jour.

Cependant, remplacer le code par celui-ci :

security:  
 encoders:  
 App\Entity\User:  
 algorithm: bcrypt  
  
 providers:  
 in\_memory: { memory: null }  
 in\_database:  
 entity:  
 class: App\Entity\User  
 property: username  
  
 firewalls:  
 dev:  
 pattern: ^/(\_(profiler|wdt)|css|images|js)/  
 security: false  
 main:  
 anonymous: lazy  
 provider: in\_database  
  
 form\_login:  
 login\_path: login  
 check\_path: login\_check  
 always\_use\_default\_target\_path: true  
 default\_target\_path: /tasks  
  
 logout:  
 path: /logout  
 target: /  
 invalidate\_session: true  
 guard:  
 authenticators:  
 - App\Security\AppAuthenticator  
  
 access\_control:  
 *# - { path: ^/login, roles: IS\_AUTHENTICATED\_ANONYMOUSLY }  
 #- { path: ^/users, roles: ROLE\_ADMIN }  
 #- { path: ^/tasks, roles: IS\_AUTHENTICATED\_FULLY }*

Étape 35 :

Dans le dossier src/Controller/SecurityController.php

Remplacez totalement la function loginAction par celle-ci :

*/\*\*  
 \** ***@Route****("/login", name="login")  
 \*/*public function login(AuthenticationUtils $authenticationUtils): Response  
{  
 // if ($this->getUser()) {  
 // return $this->redirectToRoute('target\_path');  
 // }  
  
 // 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]);  
}

En effet, avec Symfony 4, nous injectons directement la classe :

use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;

Étape 36 :

Pour remplir notre base de données, nous avons besoin de faire des migrations.

Cependant le package n’est pas encore présent.

Tapez dans le CLI, composer require migrations

Puis après l’installation, tapez :

./bin/console make:migration

Puis

php bin/console doctrine:migrations:migrate

Super cela fonctionne

Étape 37 :

Nous allons faire évoluer l’entité User. En effet nous devons être capable, lors de la création d’un utilisateur, de choisir son rôle : Administrateur ou Utilisateur.

Commençons par éditer l’entité User :

Tapez dans le CLI ,

Php bin/console make :entity

Class name of the entity to create or update (e.g. FiercePuppy):

> User

Ajoutons la propriétée roles :

New property name (press <return> to stop adding fields):

> roles

Renseignez le field type : array

Field type (enter ? to see all types) [string]:

> array

Et repondez que le field peut être null par : yes

Can this field be null in the database (nullable) (yes/no) [no]:

>yes

Puis Migrer ceci dans la base de donnée avec les commandes :

php bin/console make:migration

et

php bin/console doctrine:migrations:migrate

Remplacez dans src/Entity/User, le get et set de Roles par cela :

public function getRoles()  
{  
 $roles = $this->roles;  
 $roles[] = 'ROLE\_USER';  
 return array\_unique($roles);  
}

et

public function setRoles(array $roles): self  
{  
 $this->roles = $roles;  
  
 return $this;  
}

Cela sert à attribuez le rôle User par défaut à un utilisateur, si il n’a pas de rôle déjà attribué.

Étape 38 :

Allons éditer maintenant le formulaire de création de User et implémentons la nouvelle fonctionnalité :

<?php  
  
namespace App\Form;  
  
use Symfony\Component\Form\AbstractType;  
use Symfony\Component\Form\FormBuilderInterface;  
use Symfony\Component\Form\Extension\Core\Type\EmailType;  
use Symfony\Component\Form\Extension\Core\Type\TextType;  
use Symfony\Component\Form\Extension\Core\Type\PasswordType;  
use Symfony\Component\Form\Extension\Core\Type\RepeatedType;  
use Symfony\Component\Form\Extension\Core\Type\ChoiceType;  
use Symfony\Component\Form\Extension\Core\Type\CollectionType;  
  
class UserType extends AbstractType  
{  
 public function buildForm(FormBuilderInterface $builder, array $options)  
 {  
 $builder  
 ->add('username', TextType::class, ['label' => "Nom d'utilisateur"])  
 ->add('password', RepeatedType::class, [  
 'type' => PasswordType::class,  
 'invalid\_message' => 'Les deux mots de passe doivent correspondre.',  
 'required' => true,  
 'first\_options' => ['label' => 'Mot de passe'],  
 'second\_options' => ['label' => 'Tapez le mot de passe à nouveau'],  
 ])  
 ->add('email', EmailType::class, ['label' => 'Adresse email'])  
 ->add('roles', CollectionType::class, [  
 'label' => false,  
 'entry\_type' => ChoiceType::class,  
 'entry\_options' => [  
 'label' => 'Définir le role',  
 'choices' => [  
 'Administrateur' => 'ROLE\_ADMIN',  
 'Utilisateur' => 'ROLE\_USER',  
 ],  
 ],  
 ]);  
 }  
}

Remplacez le code par celui-ci, On peut voir que nous utilisons le CollectionType dans lequel nous implémentons un ChoiceType.

Allons tester ce code et la création d’un utilisateur.

Étape 39 :

Nous allons commencer par corriger une dépréciation, rendez vous dans src/Controller/UserController et remplacez le code suivant à la ligne 40 et 56 :

if ($form>isSubmitted() && $form>isValid()) {

Puis faites la méme chose dans src/Controller/TaskController et remplacez le code à la ligne 31 et 54

En effet, avec Symfony 4, nous devons vérifiez que le formulaire a bien été submit.

Étape 40 :

Rendez vous dans src/Controller/UserController.

Nous devons injecter l’encoder dans le code et ne plus utiliser le service :

security.password\_encoder

Remplacez la ligne 33 par celle-ci :

$password = $encoder->encodePassword($user, $user->getPassword());

Et ajoutez ceci dans les paramètres de la fonction :

UserPasswordEncoderInterface $encoder

Ce qui doit donner ceci :

public function createAction(Request $request,UserPasswordEncoderInterface $encoder)

ajoutez ensuite :

use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;

Étape 41 :

Tapez dans le CLI : symfony serve

Et rendez vous ici : http://localhost:8000/users/create

On voit le nouveau formulaire cependant il est bien moche.

Nous allons le rendre plus joli.

Rendez vous dans config/packages/twig.yaml et ajoutez cette ligne :

form\_themes: ['bootstrap\_4\_layout.html.twig']

Puis rendez vous dans templates/user/create.html.twig et ajoutez cette ligne juste après la première ligne :

{% form\_theme form 'bootstrap\_4\_layout.html.twig' %}

Refresh la page et admirez le travail.

Ce formulaire est bien plus joli. Super. Testons le et créons un administrateur.

Tout fonctionne : **Superbe !** L'utilisateur a bien été ajouté.

Étape 42 :

Nous allons maintenant customiser la fonction create et y incorporer la possibilité de modifier un utilisateur existant, remplacer le code de la fonction par celui-ci :

*/\*\*  
 \** ***@Route****("/users/create", name="user\_create")  
 \** ***@Route****("/users/{id}/edit", name="user\_edit")  
 \*/*public function createAction(User $user = null, Request $request,UserPasswordEncoderInterface $encoder)  
{  
 if (!$user) {  
 $user = new User();  
 }  
  
 $form = $this->createForm(UserType::class, $user);  
 $form->handleRequest($request);  
  
 if ($form->isSubmitted() && $form->isValid()) {  
  
 $em = $this->getDoctrine()->getManager();  
 $password = $encoder->encodePassword($user, $user->getPassword());  
 $user->setPassword($password);  
 $em->persist($user);  
 $em->flush();  
  
 if (!$user->getId()) {  
 $this->addFlash('success', "L'utilisateur a bien été ajouté.");  
 } else {  
 $this->addFlash('success', "L'utilisateur a bien été modifié.");  
 }  
  
 return $this->redirectToRoute('user\_list');  
 }  
  
 return $this->render('user/create.html.twig', [  
 'form' => $form->createView(),  
 'editMode' => $user->getId() !== null  
 ]);  
}

Vous pouvez supprimer du coup, la fonction editAction qui n’est plus d’actualité ainsi que la vue dans templates/user/edit.html.twig

Étape 43 :

Nous allons devoir éditer la vue templates/user/create.html.twig, afin qu’elle puisse répondre aux deux éventualités : création ou modification d’utilisateur.

{% extends 'base.html.twig' %}  
  
{% form\_theme form 'bootstrap\_4\_layout.html.twig' %}  
  
{% block header\_title %}  
 {% if editMode %}  
 <h1>Modifier un utilisateur</h1>  
 {% else %}  
 <h1>Créer un utilisateur</h1>  
 {% endif %}  
{% endblock %}  
{% block header\_img %}{% endblock %}  
  
{% block body %}  
 <div class="row">  
  
 {{ form\_start(form) }}  
  
 {{ form\_row(form.username) }}  
 {{ form\_row(form.email) }}  
 {{ form\_row(form.password) }}  
 {{ form\_row(form.roles.0) }}  
 {{ form\_row(form.\_token) }}  
  
 <button type="submit" class="btn btn-success">  
 {% if editMode %}  
 Enregistrer les modifications  
 {% else %}  
 Ajouter  
 {% endif %}  
 </button>  
 {{ form\_end(form, {'render\_rest': false}) }}  
  
  
 </div>  
{% endblock %}

Étape 44 :

Nous allons ensuite ajouter l’interdiction de se rendre dans ce formulaire, ainsi que vers la liste des utilisateurs si l’utilisateur n’a pas le rôle Admin.

On se rend dans src/Controller/UserController et ajoutons dans les use :

use Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;

Puis dans nos annotations :

*/\*\*  
 \** ***@Route****("/users", name="user\_list")  
 \** ***@IsGranted****("ROLE\_ADMIN")  
 \*/*

Et

*/\*\*  
 \** ***@Route****("/users/create", name="user\_create")  
 \** ***@Route****("/users/{id}/edit", name="user\_edit")  
 \** ***@IsGranted****("ROLE\_ADMIN")  
 \*/*

On remarque que l’on a ajouté :

***@IsGranted****("ROLE\_ADMIN")*

Essayer désormais de vous rendre sur la page de création, modification ou sur la liste des utilisateurs sans être Administrateur et vous serez tout de suite redirigez vers le formulaire de login.

Étape 45 :

Créons un lien pour se rendre sur la page des utilisateur et cachons le bouton de création si l’utilisateur n’est pas admin.

Dans templates/base.html.twig, ajoutons une condition à la ligne 42 :

{% if app.user and app.user.roles.0 == 'ROLE\_ADMIN' %}  
<a href="{{ path('user\_create') }}" class="btn btn-primary">Créer un utilisateur</a>  
{% endif %}

Modifions ici la route et le nom de ce bouton qui n’est pas vraiment adapté.

Remplaçons le par BackOffice Administrateur

Ce qui va donner ceci :

{% if app.user and app.user.roles.0 == 'ROLE\_ADMIN' %}  
<a href="{{ path('users') }}" class="btn btn-primary">BackOffice Administrateur</a>  
{% endif %}

Puis allez dans template/user/list.html.twig

Et ajoutez au début du block body, la ligne suivante :

<a href="{{ path('user\_create') }}" class="btn btn-primary">Créer un utilisateur</a>

Étape 46 :

Les utilisateurs sont stockés dans la base de donnée, on peut observer qu’il y a maintenant seulement un utilisateur et qu’il a le rôle admin :

