BACKOFFICE SYMFONY

Lancer tout d'abord la commande suivante pour télécharger le composant easyAdminBundle de composer :

```
composer require admin
```

2 fichiers ont été créés :

config/packages/config.yaml (Nom du BackOffice)

config/routes/easy_admin.yaml (chemin vers le BackOffice (/admin) et le Controller qui gère)

config/packages/easy_admin.yaml (appel des différentes entités dans le BackOffice)

De base dans **config/packages/easy_admin.yaml :**

```
entities:
    - App\Entity\Article
    - App\Entity\Category
    - App\Entity\User
```

Ce sont toutes les entités appelés automatiquement dans le BackOffice. **EasyAdminBundle** fait tout le travail !!! Décommetter le code pour avoir toute les tables dans le BackOffice.

Voici le code dans le fichier **config/routes/easy_admin.yaml :**

```
easy_admin_bundle:
    resource: '@EasyAdminBundle/Controller/EasyAdminController.php'
    prefix: /admin
    type: annotation
```

Ce code ne changera pas, nous garderons la route **/admin** pour le BackOffice.

Taper dans l'URL http://localhost:8000/admin , ça fonctionne !!!

Lorsque l'on souhaite ajouter un nouvel article ou une catégorie, avec les clés étrangères cela pose problème, nous avons des données de type ARRAY ou OBJECT provenant de la BDD, donc dans le fichier Entity/Article.php ajouter la méthode suivante :

```
public function __toString()
    {
        return $this->title;
    }
```

Ainsi que dans le fichier Entity/Category.php :

```
public function __toString()
    {
        return $this->title;
    }
```

Si nous observons le BackOffice, il y a des champs qui ne sont pas très pertinent, comme dans la table article, la clé étrangère de la table catégorie et inversement… Nous allons modifier le fichier **config/packages/easy_admin.yaml** afin de définir nos propres champs et définir une liste beaucoup plus complète :

```
easy_admin:
  entities:
    Article:
        class: App\Entity\Article
        form:
          fields:
            - {property: 'title'}
            - {property: 'content'}
            - {property: 'image'}
            - {property: 'category'}
            - {property: 'createdAt'}
Category:
        class: App\Entity\Category
        form:
          fields:
            - {property: 'title', label: 'Titre'}
            - {property: 'description'}
User:
        class: App\Entity\User
        form:
          fields:
            - {property: 'email'}
            - {property: 'username'}
```

Pour gérer le menu et personnaliser le BackOffice ajouter les lignes suivantes au début du fichier :

```
easy_admin:
    design:
      # brand_color: '#1ABC9C'
      menu:
        - label: 'Blog'
          icon: 'newspaper-o'
          children:
            - {entity: 'Article', icon: 'folder-open', label: 'Articles'}
            - {entity: 'Category', icon: 'folder-open', label: 'Catégories'}

        - label: 'Accès'
          icon: 'user'
          children:
            - {entity: 'User', icon: 'folder-open', label: 'Utilisateurs'}
```

Pour plus de clarté dans le code nous allons décomposer et répartir les différentes parties du BackOffice.

Dans le dossier packages créer la hiérarchie suivante :

Config/packages/admin/entities :

➔ Article.yaml :

```yaml
easy_admin:
  entities:
    Article:
        class: App\Entity\Article
        form:
          fields:
            - {property: 'title'}
            - {property: 'content'}
            - {property: 'image'}
            - {property: 'category'}
            - {property: 'createdAt'}

        list:
          sort: 'createdAt'
          actions:
            - {name: 'edit', icon: 'pencil', label: false, css_class: 'btn btn-primary mb-
1'}
            - {name: 'delete', icon: 'trash', label: false, css_class: 'btn btn-danger'}
```

➔ Category.yaml :

```yaml
easy_admin:
  entities:
    Category:
        class: App\Entity\Category
        form:
          fields:
            - {property: 'title', label: 'Titre'}
            - {property: 'description'}

        list:
          actions:
            - {name: 'edit', icon: 'pencil', label: false, css_class: 'btn btn-primary'}
            - {name: 'delete', icon: 'trash', label: false, css_class: 'btn btn-danger'}
```

➔ User.yaml :

```yaml
easy_admin:
  entities:
    User:
        class: App\Entity\User
        form:
          fields:
            - {property: 'email'}
            - {property: 'username'}
            - {property: 'Roles', type: 'collection'}

        list:
          actions:
            - {name: 'edit', icon: 'pencil', label: false, css_class: 'btn btn-primary'}
            - {name: 'delete', icon: 'trash', label: false, css_class: 'btn btn-danger'}
```

List : permet de personnaliser le BackOffice (ajouter des boutons bootstrap, des icônes etc…)

Pour donner la bonne direction des composants, dans le fichier **config/packages/easy_admin.yaml** ne laisser que :

```
imports:
    - {resource: admin/}
```

On précise que l'on importe les différents fichiers .yaml directement ici !!

Maintenant nous allons paramétrer les droits d'utilisateurs, nous allons définir 2 rôles :

- ROLE_USER : accès à toute la partie front (membre connecté ou visiteur)
- ROLE_ADMIN : accès au BackOffice mais aussi à la partie front

Pour cela ajouter une entité (un champ) à la table User.

```
php bin/console make:entity
```

De la table User :

Roles → type array json → not null

Générer maintenant le fichier de migration

```
php bin/console make:migrate
```

Lancer maintenant la migration :

```
php bin/console doctrine:migrations:migrate
```

Observer la BDD, nous avons maintenant un champ 'Roles', ajouter manuellement un role ADMIN à un membre :

```
["ROLE_ADMIN"]
```

Et un rôle USER à un autre membre :

```
["ROLE_USER"]
```

Modifier maintenant le fichier **config/packages/security.yaml** afin de gérer les rôles des utilisateurs :

```
role_hierarchy:
        ROLE_ADMIN: ROLE_USER

    # 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: ^/admin, roles: ROLE_ADMIN }
        # - { path: ^/profile, roles: ROLE_USER }
        - { path: ^/login$, roles: ROLE_USER }
```

Penser à modifier la méthode dans le fichier Entity/User.php pour renvoyer les bons rôles de la BDD :

```
public function getRoles() {
        return $this->Roles;// utilisateur classique
    }
```

Afin de pouvoir ajouter et modifier les rôles dans le BackOffice, modifier la ligne suivante dans le fichier **packages/admin/entites/User.yaml**, les rôles sont renvoyés sous forme de tableau ARRAY, il n'est pas possible d'afficher un tableau ARRAY avec une simple instruction d'affichage :

```
 - {property: 'Roles', type: 'collection'}
```

Pour ajouter le lien BackOffice dans la nav, ajouter dans le fichier **template/base.html.twig** :

```
{% if is_granted('ROLE_ADMIN') %}
                <li class="nav-item">
                    <a class="nav-link" href="{{ path('easyadmin') }}">BackOffice</a>
                </li>
                {% endif %}
```

Afin de rediriger vers la page connexion en cas de tentative d'accès au BackOffice par un ROLE_USER, ajouter cette ligne à la fin du fichier **config/packages/security.yaml** :

```
access_denied_url: /connexion
```