TP Symfony - Contrôleurs & Twig

Contexte

On va réaliser un site où des posts peuvent être créés par tout le monde (y compris des personnes non-authentifiées), et où des personnes authentifiées peuvent commenter les posts.

On dispose donc de 3 tables:

- La table **Post**, qui contient tous les posts, avec leur auteur, leur contenu leur titre et leur commentaires.
- La table **Comment** contenant tous les commentaires, avec le post visé, l'auteur du commentaire et son contenu.
- La table **User** qui contient tous nos utilisateurs.

Voici un MCD, au cas-où vous êtes perdu:

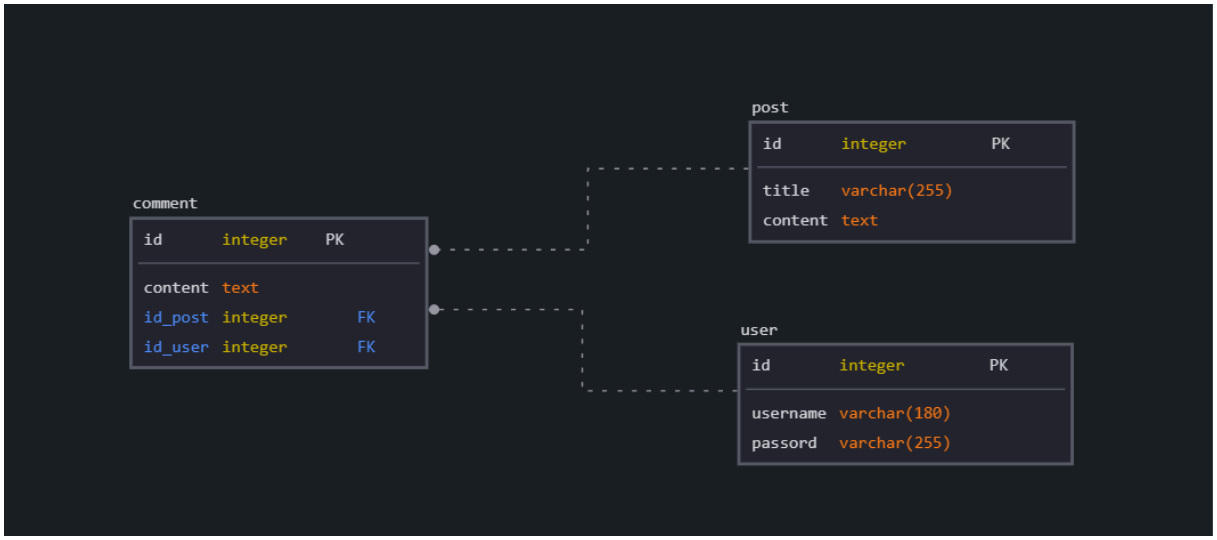

Figure 3: MCD de la base de données.

Mise en place

On va commencer par mettre en place le projet

```
# Télécharge les dépendances
composer update

# Crée la base de données
symfony console doctrine:database:create

# Crée la migration
symfony console make:migration

# Envoie la migration sur le serveur
symfony console doctrine:migrations:migrate
```

Création du formulaire de création d'User

Parce que ce n'est pas le sujet ici, **l'Entity User est déjà créée**. Par contre le formulaire d'inscription et de connexion des User ne sont pas créés. Ca va être à vous de le faire, mais vous ne serez pas largués !

- 1: Créer un formulaire permettant de s'inscrire en tant qu'utilisateur. L'utilisateur doit être identifié par un username, pas un email. On ne souhaite pas vérifier l'email entré, on souhaite que l'utilisateur soit automatiquement authentifié après la création de son compte et après l'enregistrement de son compte, il doit être redirigé vers la page nommée **app_index**. Voici une source pour aiguiller vos recherches:
  - Symfony 4 Doc - Registration Form

- 2: Avoir créé le formulaire c'est bien, mais y avoir accès c'est mieux ! Rajouter le lien vers le formulaire d'enregistrement dans le **<a>** correspondant dans l'index. *indices: regarder la route dans Controller/RegistrationController.php et le template utilisé dans IndexController.php*

- 3: Maintenant créer un formulaire de connexion serait vachement bien ! Heureusement, Symfony peut le faire pour nous. On voudrait aussi une URL permettant de se déconnecter. Ah et ça serait bien que le site se "souvienne" de nous par défaut ! Trouvez comment faire à partir des sources fournies:
  - Symfony Doc - Form Login Setup

- *4: Bonus*: Ca serait vachement drôle si la page d'index nous indiquait notre pseudo si on est connecté ! Ca devrait être faisable à partir des sources fournies ci-dessous et du cours !
  - Twig Doc - If Clauses
  - *indice: dans la fonction index de votre IndexController, vous pouvez demander un ?UserInterface en paramètre, qui possède une méthode getUsername ! Vérifiez par contre qu'il n'est pas null, car si il l'est, alors l'utilisateur n'est pas connecté et vous ne pouvez pas vous en servir.*

- 5: Devinez quoi ? Rajouter un petit bouton de déconnexion dans l'index serait pas mal, non ?

- *6: Bonus*: Ce qui serait hilarant maintenant c'est que le bouton "Se déconnecter" ne soit visible que si l'utilisateur est connecté, et que les boutons "Se connecter" ainsi que "Créer un utilisateur" ne soient pas visibles si on est connectés. Enfin j'imagine qu'on occupe pas nos week-ends de la même façon ?

Création d'un Post

Qui dit création d'un Post dit insertion dans la base de données. Et qui dit insertion dans la base de données dit formulaire ! Nos posts sont très simples:

- Un titre, qui est un varchar(255).
- Le contenu, qui est juste du texte.

- 7: Générer le contrôleur CreatePostController avec la commande:

```
# Crée le contrôleur CreatePostController, avec les fichiers Controller/CreatePostControler.php et templates/create_post/index.html.twig
symfony console make:controller CreatePostController
```

- 8: Changer la route de CreatePostController en **createpost** au lieu de /**create/post**. Parce qu'on aime pas les routes imbriquées.

- 9: Faites votre petit formulaire dans le fichier Twig correspondant. La méthode doit être POST, et l'action sur **createpost**.

Poster un Post

Une route peut être utilisée pour plusieurs méthodes. Perdu ? Prenons le temps de s'expliquer:

- Quand vous essayez de naviguer sur un site, par exemple sur votre **localhost**, votre navigateur effectue une requête GET. C'est la requête la plus simple qui existe, le serveur a juste à renvoyer la ressource correspondante.
- Quand votre navigateur doit renvoyer un formulaire, le développeur, *vous*, privilégiez 99% du temps de passer par la méthode **POST**.

Si vous avez bien spécifié **method=post** dans votre formulaire, et la bonne action, on aura la route **createpost** utilisée pour deux choses à la fois ! L'une pour obtenir le formulaire et l'autre pour le submit ! Heureusement, chez Symfony ils ont pensé à nous.

Les **faibles** penseront à faire deux routes distinctes. Mais nous on est extrêmement puissants, c'est pourquoi nous allons **dupliquer** notre fonction **index**. Votre fichier CreatePostController.php ressemble actuellement à ça:

```
<?php
class CreatePostController extends AbstractController
{
    #[Route('/createpost', name: 'app_create_post')]
    public function index(): Response
    {
        return $this->render('create_post/
index.html.twig', [
            'controller_name' =>
'CreatePostController',
        ]);
    }
} ?>
```

Dupliquez la fonction **index**, avec son attribut route. Sisi je vous jure, faites-le.

Bien, maintenant que c'est fait on va les **tuner** un peu. Renommez la première fonction **index_get**. et changez son attribut **#[Route]** en:

```
    #[Route('/createpost', name:
'app_create_post_get', methods: ["GET"])]
```

Prenez son clone et renommez-là **index_post**. Changez son attribut **#[Route]** en:

```
    #[Route('/createpost', name:
'app_create_post_post', methods: ["POST"])]
```

Et maintenant devinez quoi ? Chaque fois que l'on voudra obtenir, **GET** la page, Symfony appellera la fonction **index_get**, et dès que l'on voudra submit notre formulaire, **POST**er notre Post, il appellera la fonction **index_post**.

Nous ne toucherons pas à **index_get** car il instancie déjà correctement **templates/create_post/index.html.twig**. Par contre **index_post** n'enregistre encore rien. On va y remédier.

Ajoutez dans les paramètres de **index_post** la requête, de façon à avoir:

```
    public function index_post(Request $request):
Response
```

Attention: la classe à importer est Symfony\Component\HttpFoundation\Request;

Vous avez désormais accès aux valeurs transmises par votre formulaire. Pour cela, faites simplement

```
  $request->get(" ... ");
```

Pour récupérer les valeurs envoyées par le formulaire !

- 10: En vous servant de la documentation fournie ci-dessous et des informations données ci-dessus, trouvez un moyen de créer un Post et de l'enregistrer dans votre base de données.
  - Symfony - Persisting Objects to the Database

- *11: Bonus*: Ce qui serait génial c'est qu'en nous renvoyant sur le formulaire après avoir fait persister le Post en base de données, la page nous signale que l'insertion s'est bien passée.

Lister les Posts

- 12: Bon, maintenant ce qui serait bien, c'est que les posts soient visibles ! Créez donc un contrôleur nommé **ListPostsController**. Changez sa route en **listposts**, et affichez-les tous ! Voici des petits liens pour vous aiguiller dans la réalisation d'une telle page:
  - Symfony: Fetching Objects from the Database *Spoiler: EntityManagerInterface et tous les Repository ont une méthode findAll !*
  - Twig: For Loops

Ajout et Listing des Commentaires

- 13: Avec tout ce que vous avez fait jusque-là, bravo ! Vous pouvez désormais vous débrouiller tout seul ! Faites désormais une page permettant d'ajouter un commentaire sur un Post, et ouvrez-y l'accès depuis l'index. Le Post **doit** exister, et l'utilisateur **doit** être connecté afin de pouvoir envoyer un commentaire. Faites ça sur la route **sendcomment**. Une route **GET** et une route **POST pourraient** être nécessaires à chaque fois :^).
  - *indice: dans la fonction index de votre IndexController, vous pouvez demander un ?UserInterface en paramètre, qui possède une méthode getUsername ! Vérifiez par contre qu'il n'est pas null, car si il l'est, alors l'utilisateur n'est pas connecté et vous ne pouvez pas vous en servir.*

- 14: Pour voir ces fameux commentaires, en revanche, pas besoin d'être connecté. Une route **GET** et une route **POST** devraient être encore une fois nécessaires, faites ça sur **listcomments**.

- 15: Maintenant qu'on a accès à tout ce qu'on veut, vous trouvez pas que le site est vachement laid ? Arrangez tout ça, occupez-vous du style du site !
  - *Note: tout ce qui est images/css/js doit être mis dans le dossier **public**, et pour accéder à son contenu rien de plus simple. Admettons que la fiche css est dans **public/style.css**, on y accède via un simple*

    ```
    <link rel="stylesheet" href="style.css">
    ```

    *Vous pouvez d'ailleurs ajouter votre **link rel** directement dans **base.html.twig** comme ceci:*

    ```
    {% block stylesheets %}
      <link rel="stylesheet" href="style.css">
    {% endblock %}
    ```

elle sera ainsi incluse par défaut dans tous les autres templates !