Fialoux mateo

Compte rendu Entité Symfony

1- Création d'une entité

1.1 Préambule

1.1.1 Créer un projet Symfony nommé Quote (citation) dans votre répertoire de TP. Quelle est la commande à exécuter?

Créer le projet Symfony

echo -e "${GREEN}Création du projet Symfony : $PROJECT_NAME...${RESET}"

"$SYMFONY_CLI" new "$PROJECT_NAME" --webapp

1.1.2 Assurez-vous ensuite que maker-bundle est installé en lançant la commande suivante. Expliquer le rôle de la commande puis de maker-bundle

La commande affiche toutes les bibliothèques installées dans le projet.

symfony/maker-bundle 1.64.0 Symfony Maker helps you create empty commands, controllers, form classes, tests and more so you can forget about writing...

Il est bien installé.

1.1.3 Nous devons maintenant configurer la base de données. Pour ce projet, nous utiliserons SQLite. Localiser le fichier ".env" et éditer le en conséquence. Indiquer son chemin d'accès et les modifications effectuées dans le fichier.

```
DATABASE_URL="sqlite:///%kernel.project_dir%/var/data.db"
```

1.1.4 Lister les autres types de format de base de données supportées par Symfony.

- ➔ **SAP Hana**
- ➔ **IBM DB2**
- ➔ **Oracle**
- ➔ **SQL Server (MSSQL)**
- ➔ **SQLite**
- ➔ **PostgreSQL**
- ➔ **MySQL / MariaDB**

1.2 Création d'une entité Quote (citation)

1.2.1 Lancer la commande suivante et créer les champs text et author

mfialoux@PC-F205-10:~/Documents/php/symfony/Quote$ bin/console make:entity

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

> Quote

Add the ability to broadcast entity updates using Symfony UX Turbo? (yes/no) [no]:

> no

created: src/Entity/Quote.php

created: src/Repository/QuoteRepository.php

Entity generated! Now let's add some fields!

You can always add more fields later manually or by re-running this command.

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

> text

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

> text

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

> yes

updated: src/Entity/Quote.php

Add another property? Enter the property name (or press <return> to stop adding fields):

> author

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

> string

Field length [255]:

>

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

>

updated: src/Entity/Quote.php

Add another property? Enter the property name (or press <return> to stop adding fields):

>

Success!

Next: When you're ready, create a migration with php bin/console make:migration

mfialoux@PC-F205-10:~/Documents/php/symfony/Quote$

Entity il y a un nouveau fichier Quote.php ou a l'intérieur il y a une class Quote et dans le dossier Repository il y a QuoteRepository.php c'est la classe responsable de gérer l'accès aux données de cette entité dans la base de données.

1.2.2 Lister l'ensemble des types disponibles pour la création d'un champ.

- ➔ string
- ➔ text
- ➔ integer
- ➔ smallint
- ➔ bigint
- ➔ float
- ➔ decimal
- ➔ boolean
- ➔ date
- ➔ time

- ➔ datetime
- ➔ datetimetz
- ➔ array
- ➔ json
- ➔ object
- ➔ binary
- ➔ uuid
- ➔ guid

1.2.3 A la fin de la création de notre entité Quote, le message suivant apparaît. Faire une recherche et expliquer en quoi consiste une migration. Lancer ensuite l'opération comme indiqué.:

Une migration permet de passer d'un état A à un état B de ta base de données par exemple, ajouter une table, modifier une colonne, supprimer un champ, etc.

1.2.4 Se rendre dans le répertoire "migrations" du projet. Un fichier commençant par "Version" y a été créé. Analyser et expliquer son rôle.

Dans le dossier /migrations/ ou à l'intérieur on retrouve un fichier php ou il y a des requêtes SQL qui permettent la mise en version et par conséquent des sauvegardes pour la moindre erreur dans la base de données.

2- Découverte d'une entité

2.1 Examiner la classe Quote.php. Quels sont les accesseurs et mutateurs dont-elle dispose?

- ➔ Propriété : $id
  - ◆ Accesseur : public function getId(): ?int
  - ◆ Mutateur : Aucun car l'id est auto générée
- ➔ Propriété : $text
  - ◆ Accesseur : public function getText(): ?string
  - ◆ Mutateur : public function setText(?string $text): static
- ➔ Propriété : $author
  - ◆ Accesseur : public function getAuthor(): ?string
  - ◆ Mutateur : public function setAuthor(string $author): static

2.2 Que vaut l'id d'un objet Quote lors de son instanciation?

Par défaut l'id vaut null

```
private ?int $id = null;
```

2.3 Que signifie le point d'interrogation devant la déclaration de certaines variables comme l'id?

Ça signifie que la variable peut être de ce type ex : int ou alors null.

2.4 A quoi servent des attributs ORM (#[ORM...) dans le code et peut-on les supprimer de la classe Quote?

```
#[ORM\GeneratedValue]
```

sont des annotations en PHP 8+ utilisées par Doctrine pour comprendre comment relier la classe PHP à une table de base de données.
Non on ne peut pas les supprimer car sinon Doctrine ne saura plus comment gérer la base de données et elle perdra malheureusement son travail.

2.5 Quel est le rôle joué par la classe QuoteRepository?

La classe QuoteRepository permet d'accéder aux données de la table quote en base de données, via des méthodes de recherche, de filtre, de tri, etc.

3- Utilisation d'une entité

3.1 Maintenant que nous disposons de notre entité, il nous faut encore créer un contrôleur pour la gérer. Là encore, la console de Symfony nous permet de le faire automatiquement avec la commande suivante: Exécuter la commande (laissez les paramètres par défaut). Quels sont les fichiers créés?

On a deux nouveau fichiers /templates/quote/index.html.twig et /src/Controller/QuoteController.php

3.2 Que nous indique le débogage des routes concernant create_first_quote?

```
+--------------+-------------------------------------------------------------------+
| Property     | Value                                                             |
+--------------+-------------------------------------------------------------------+
| Route Name   | create_first_quote                                                |
| Path         | /create-first-quote                                               |
| Path Regex   | {^/create\-first\-quote$}sDu                                      |
| Host         | ANY                                                               |
| Host Regex   |                                                                   |
| Scheme       | ANY                                                               |
| Method       | ANY                                                               |
| Requirements | NO CUSTOM                                                         |
| Class        | Symfony\Component\Routing\Route                                   |
| Defaults     | _controller: App\Controller\QuoteController::createFirstQuote()   |
| Options      | compiler_class: Symfony\Component\Routing\RouteCompiler           |
|              | utf8: true                                                        |
+--------------+-------------------------------------------------------------------+
```

Ça donne des informations précises sur la route create_first_quote.

3.3 Il est indiqué dans un commentaire que l'on récupère le singleton de Doctrine. Expliquer ce qu'est le design pattern Singleton en langage objet.

Le Singleton est un design pattern qui garantit que seule une instance d'une classe peut exister à la fois dans une application.

3.4 Lancer le serveur et accéder à l'url de notre page Ouvrir la base de données dans l'éditeur de votre choix. Recharger là page plusieurs fois et expliquer ce qu'il se passe.

A chaque rafraichissement de la page cela me crée un nouveau champ dans la base en incrémentant de 1 à chaque fois

3.5 Ajouter la méthode suivante à votre contrôler Survoler l'icône "cible" en bas, que se passe-t-il?

array:7 [▼

0 => App\Entity\Quote {#839 ▸}

1 => App\Entity\Quote {#841 ▸}

2 => App\Entity\Quote {#842 ▸}

3 => App\Entity\Quote {#843 ▸}

4 => App\Entity\Quote {#844 ▸}

5 => App\Entity\Quote {#845 ▸}

6 => App\Entity\Quote {#846 ▸}

]

C'est le résultat du dump($quotes) dans ta méthode listQuotes(). Affichage avec un tableau

3.6 Mettre la ligne dump() en commentaire et recharger la page. Que constatez vous?

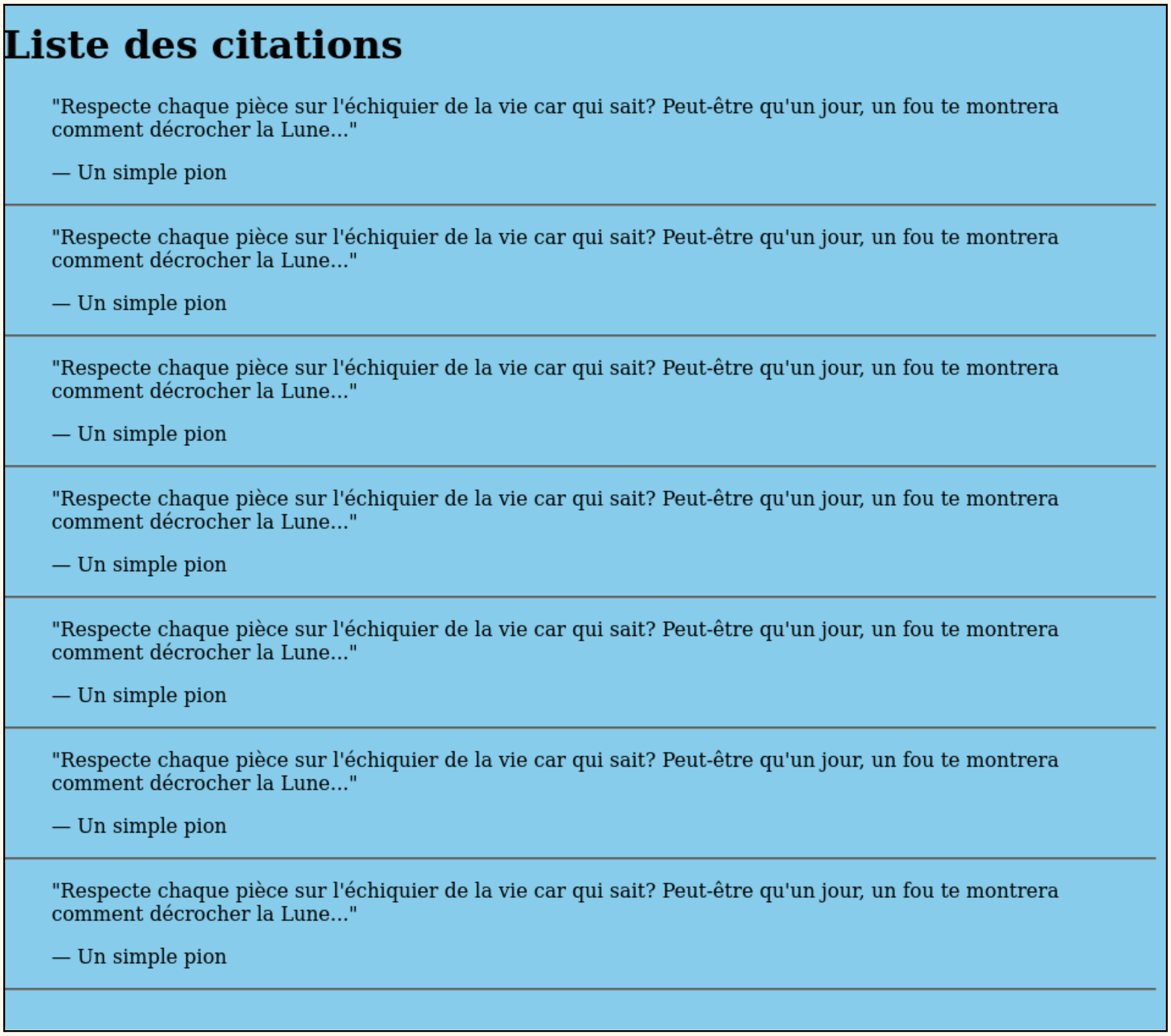
Je n'ai plus accès à la cible.

3.7 Documenter le code de la méthode.

```
public function quotes(ManagerRegistry $doctrine): Response

    {

        // Récupération de toutes les citations depuis la base de données

        $quotes = $doctrine->getRepository(Quote::class)->findAll();
```

```
        // Affichage de la page HTML avec les citations (à transmettre à la
vue)
        return $this->render('quote/index.html.twig', [
            'controller_name' => 'QuoteController',
            // Il manque l'envoi de la variable $quotes ici pour afficher
les citations dans le template
            'quotes' => $quotes
        ]);
    }
```

3.8 Nous souhaitons maintenant afficher la liste de citations afin d’obtenir une page de la forme suivante



Modifier la méthode quotes() du contrôleur afin de retourner à la page twig, en plus du titre du contrôleur, la liste des citations

```
{% extends 'base.html.twig' %}



{% block title %}Liste des citations{% endblock %}

```

```
{% block body %}

   <h1>Liste des citations</h1>



   {% for quote in quotes %}

       <blockquote>

           <p>"{{ quote.text }}"</p>

           <footer>— {{ quote.author }}</footer>

       </blockquote>

       <hr>

   {% else %}

       <p>Aucune citation enregistrée.</p>

   {% endfor %}

{% endblock %}
```

3.9 Modifier la page du template Twig afin d'afficher la liste comme sur la capture (faire des recherches sur la syntaxe de balisage Twig).

```
{% extends 'base.html.twig' %}



{% block title %}Liste des citations{% endblock %}



{% block body %}

   <h1>

       Voici les citations
```

```
        <span style="color: green;">&#x2705;</span> {# ✅ icône "check"
verte #}
    </h1>


    {% if quotes is not empty %}
        <ul style="list-style-type: disc; padding-left: 20px;">
            {% for quote in quotes %}
                <li style="margin-bottom: 15px;">
                    <strong>"{{ quote.text|e }}"</strong><br>
                    <em> {{ quote.author|e }}</em>
                </li>
            {% endfor %}
        </ul>
    {% else %}
        <p>Aucune citation enregistrée.</p>
    {% endif %}
{% endblock %}
```

J'ai également enlevé le fond bleu dans le css.

4- Aller plus loin

Pour le formulaire j'ai du créer un formulaire avec cette commande php bin/console make:form QuoteType ou j'ai entrée l'entité Quote ou on a nos citation

The name of Entity or fully qualified model class name that the new form will be bound to (empty for none):

> Quote

puis une fois le fichier créé partie model j'ai attaqué la modification du contrôleur avec une nouvelle route

```
#[Route('/add-quote', name: 'add_quote')]

   public function addQuote(Request $request, ManagerRegistry $doctrine):
Response

   {

       $quote = new Quote();

       $form = $this->createForm(QuoteType::class, $quote);

       $form->handleRequest($request);


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

           $entityManager = $doctrine->getManager();

           $entityManager->persist($quote);

           $entityManager->flush();


           $this->addFlash('success', 'Citation ajoutée avec succès !');

           return $this->redirectToRoute(route: 'quote-listing');
```

```
        }

        return $this->render('quote/add.html.twig', [

            'form' => $form->createView(),

        ]);

    }
```

Une fois ma route crée j'ai crée le fichier add.html.twig ou j'ai écrit le formulaire sur du html

```
{% extends 'base.html.twig' %}


{% block title %}Ajouter une citation{% endblock %}


{% block body %}

    <h1>Ajouter une nouvelle citation</h1>


    {{ form_start(form) }}

        {{ form_row(form.text) }}

        {{ form_row(form.author) }}

        <button class="btn">Ajouter la citation</button>

    {{ form_end(form) }}


    {# --<a href="{{ path('/quotes') }}">Retour à la liste des
citations</a> #}

{% endblock %}
```