TP2 symfony

Objectifs

Créer un site avec le framework symfony 5 en créant des entités.

Durée

3h

Préambule

Les données de ce TP seront utilisées pour la prochaine séance. Il faudra bien penser à sauvegarder les données de votre base de données **en fin de séance** pour éviter de perdre du temps la prochaine fois.

Sauvegarder : Clic droit sur la base sftp2: Dump with mysqldump en sauvegardant dans votre dossier /media/Qi/$USER

Restaurer : On crée la base de données sftp2 puis clic droit sur la base symfony : Run SQL Script

Préparation du poste de travail

Vous allez utiliser **les machines virtuelles linux** pour réaliser l'ensemble des Tps de ce semestre.

La création d'un projet symfony se fait en ligne de commande.

Placez-vous dans le dossier /media/Qi/$USER.

```
composer create-project symfony/skeleton practiceSymfony
```

Nous utiliserons phpstorm comme IDE pour développer nos sites. Vous trouverez PhpStorm dans les outils installés.

PhpStorm est un outil commercial. En tant qu'étudiant, vous pouvez obtenir une licence gratuite en vous inscrivant avec votre adresse universitaire sur le site de Jetbrain.

Il permet d'utiliser un terminal intégré ainsi qu'un connecteur sur les bases de données. (et pleins d'autres choses ...)

Ouvrez le dossier practiceSymfony avec phpStorm.

Nous utiliserons maintenant que le terminal intégré à phpStorm.

Nous allons installer des composants additionnels pour notre Tp.

```
composer require --dev profiler maker
```

```
composer require annotations twig form validator orm asset
```

Nous pouvons maintenant lancer notre site :

```
symfony server:start
```

Normalement, si vous allez sur http://127.0.0.1:8000, vous devez voir votre site. Vous tombez sur une page 404 de symfony. (La page demandée n'est pas trouvée.)

Mise en place du site

Vous trouverez sous moodle l'ensemble des éléments nécessaires à la réalisation de ce site.

Copiez les dossiers images, css, js, fonts dans public/

Comme dans le TP1, mettez en place le fichier base.html.twig et index.html.twig avec les ressources que vous avez récupérées.

N'oubliez pas les assets.

À ce stade, vous devez avoir la mise en forme du site qui est correcte.

Création du fichier .env.local

Vous allez devoir démarrer le serveur mysql de la machine :

```
sudo service mysql start
```

Un utilisateur spécifique a été créé pour pouvoir accéder au moteur MySQL avec tous les droits nécessaires pour administrer des bases de données
Utilisateur : user
Mot de passe : user

Afin de pouvoir communiquer avec votre base de données, ajoutez la ligne de configuration suivante dans le fichier .env.local (fichier à créer):

```
DATABASE_URL=mysql://user:user@localhost:3306/sftp2
```

Ne mettez pas cette ligne dans le fichier .env car cette configuration correspond à l'environnement local. Si vous changez de serveur, la configuration sera différente. D'autre part, cette ligne contient le login et le mot de passe de l'accès à la base. C'est sensible...

Avec la commande symfony suivante, vous allez créer la base de données de l'application :

```
bin/console doctrine:database:create
```

Configurez le module database de phpstorm pour vous connecter à la base de données afin de visualiser votre base de données.

Création d'une entité

En utilisant la bonne commande symfony, créez une entité "Produit"

- nom : string
- description : text
- prix : float
- image : string

Ajoutez un constructeur à la classe Produit (Dans Phpstorm : Alt+Inser)

Dans le contrôleur, dans une méthode adéquate, créez un nouveau produit.

En cliquant sur le lien exo1, il faut aller vers la page /produit/test qui affiche le produit créé

Pour l'instant, le produit n'est pas issu de la base de données, il est uniquement en mémoire.

Création effective de la table

En utilisant la bonne commande symfony, faites l'association entre l'entité et la table au niveau de la base

Ajoutez des produits en allant sur dans l'onglet Database de Phpstorm (les images seront enregistrées dans le dossier images adéquat en dur)

Dans le champ image, nous écrirons uniquement le nom du fichier image

En cliquant sur le lien exo1bis, il faut aller vers la page /produit qui affiche tous les produits enregistrés sous forme d'un tableau

Ajout d'une seconde entité

Créez une entité "Categorie"

- nom : string

Ajoutez des catégories dans la table. (pensez à rafraîchir la vue)

En cliquant sur le lien exo2, il faut aller vers la page /categorie qui affiche toutes les catégories enregistrées

Modification de l’entité Produit

Ajoutez un champ categorie à la classe Produit en utilisant la commande symfony

En type, vous pouvez utilisez relation

Liaison entre produit et catégorie : un produit appartient à une catégorie (many-to-one)

Visualiser l'entité Produit pour voir les changements opérés.

Ajoutez une catégorie existante à chaque produit

Faites une méthode __toString() dans la classe Categorie

En cliquant sur le lien exo1bis, affichez maintenant les produits avec le nom de la catégorie associée

Création d’un formulaire d’ajout de produit

En cliquant sur le lien exo3, il faut aller vers la page /produit/ajout qui affiche un formulaire permettant d’ajouter un produit en choisissant sa catégorie

```
bin/console make:form
```

Vous devrez modifier légèrement le constructeur de la classe Produit

Le champ prix devra être positif

Création d’une page pour lister, ajouter, supprimer et éditer une catégorie

En cliquant sur le lien exo4, il faut aller vers la page /categorie/ qui affiche la liste des catégories et aura un lien pour ajouter une catégorie

Chaque catégorie aura un lien pour l’éditer et la supprimer

```
bin/console make:crud
```

Observez attentivement l’ensemble du code généré : Controller, Formulaire, Vue…

Pourquoi y a-t-il des fichiers de vue commençant par _ ?

Quel formulaire est utilisé pour la création et la modification ?

Comment fait il pour supprimer un élément ?