Consignes Séquence 6 : Doctrine

Savoir utiliser Doctrine pour administrer une base de données et ses données

Exercice 7

Toujours via le terminal, créer l'entité « **Cars** ». Elle devra être composée des champs suivants :

- **brand**, string, 255 caractères, non nul
- **model**, string, 255 caractères, non nul
- **year**, integer, non nul
- **engine**, string, 255 caractères, non nul
- **paint**, string, 255 caractères, non nul

Une fois l'entité générée, effectuer la **migration** vers la base de données.

Exercice 8

A l'aide de **Composer, installer orm-fixtures**.

Le but de cet exercice est de **créer des jeux de données** à insérer directement en base.

Dans un premier temps, il faudra créer une **nouvelle fixture** dans le **terminal** grâce à la commande « **php bin/console make:fixtures** ».
Elle portera le nom « **CarFixtures** ».

Ensuite, à l'intérieur du fichier nouvellement créé « **src/DataFixtures/CarFixtures** », créer dans la méthode **load()** 5 objets représentant 5 voitures différentes à ajouter en base.

Enfin, effectuer la **migration en base** des données du fichier fixtures.

Exercice 9

Récupérez les données que vous venez d'entrer en base via fixtures grâce à la **méthode findAll()** de Doctrine.
Une fois ces données réceptionnées, **affichez-les** sur la vue « **Car/index.html.twig** ».

Exercice 10

A l'aide de **Composer**, installer **cocur/slugify**.
Cette bibliothèque vous permettra de "jouer" avec vos **URL** plus facilement.

Une fois **slugify** installé, créer dans l'entité Cars le **getter** permettant la récupération d'un **slug**.

Puis, dans le **CarController**, créer une méthode **show()**, qui permettra la visualisation des détails d'une voiture.

Exercice 11

Créer une **nouvelle entité « EnergyOption »** via le **terminal**. Cette entité permettra l'ajout du type d'énergie utilisé par la voiture (essence, gasoil, diesel, électrique etc...).

Elle devra contenir les champs :

- **name**, string, 255 caractères, non nul
- **car**, relation avec la classe **Cars**, type ManyToMany.

ATTENTION : Une fois l'entité générée et **AVANT** de faire la migration, **vérifier** si le sens de la relation est correct (grâce au **mappedBy** et **inversedBy** dans les 2 entités en jeu).

Effectuer ensuite la **migration** vers la base de données.

Exercice 12

Maintenant que la nouvelle entité a été créée, vous allez **générer automatiquement tout le CRUD** correspondant à ces options.

Dans le terminal, tapez la commande « **php bin/console make:crud** » et renseignez la classe **EnergyOption**.

Une fois le CRUD généré, faire les **adaptations nécessaires** afin de conserver votre style récurrent dans les nouveaux templates **créés par symfony**.

Exercice 13

Après avoir **généré et vérifié** le bon fonctionnement du **CRUD EnergyOption**, ajouter différents types de carburant grâce au **formulaire d'ajout créé automatiquement par Symfony** et faire en sorte d'afficher les différents types d'énergie **dans la liste des voitures (Car/index.html.twig).**

Une fois cette étape réalisée, faire en sorte d'avoir une **recherche** sur la page permettant de sélectionner un type précis d'énergie et de sortir les voitures **reliées** à celui-ci.

Exercice 14

Avec les connaissances acquises jusqu'à présent, ajouter 2 nouvelles options aux voitures :

- Le nombre de km
- Le nombre de places

Toutes les caractéristiques des voitures devront montrer à l'utilisateur les voitures disponibles en fonction de la caractéristique choisie.

Contrainte : Le nom de la caractéristique choisie devra figurer en **haut de la page** en question.