Atelier : Création d'un site e-commerce avec Symfony 6

Prérequis :

- PHP 8 installé
- Composer installé
- Symfony CLI installé

Objectifs de l'atelier :

- Comprendre l'architecture MVC de Symfony
- Configurer un projet Symfony
- Créer des entités et interagir avec une base de données
- Développer des contrôleurs et gérer le routage
- Utiliser Twig pour le rendu des vues
- Gérer les assets avec Webpack Encore
- Mettre en place un système d'authentification
- Mettre en place une navigation dynamique

Plan de l'atelier :

1. Présentation de l'architecture MVC

- Explication du modèle MVC (Modèle-Vue-Contrôleur) et son application dans Symfony.

2. Initialisation du projet Symfony

- Créer un nouveau projet Symfony :
  symfony new my_project_directory --version="6.2.*" --webapp
- Naviguer dans le répertoire du projet :
  cd my_ecommerce

3. Configuration du projet

- Dupliquer le fichier `.env` en `.env.local`.
- Modifier le fichier `.env.local` pour définir la variable `DATABASE_URL`.
- Ajouter les fichiers sensibles au fichier `.gitignore`.

4. Installation des dépendances backend

- Installer les bundles nécessaires :
  composer require symfony/maker-bundle
  composer require doctrine

5. Création des entités et de la base de données

- Créer une entité `Product` :
  php bin/console make:entity Product
- Ajouter les propriétés suivantes : `name`, `description`, `price`, `image`.

- Créer la base de données :
  php bin/console doctrine:database:create
- Générer et exécuter les migrations :
  php bin/console make:migration
  php bin/console doctrine:migrations:migrate

6. Création des contrôleurs et des routes

- Créer un contrôleur pour les produits :
- Définir les routes pour les actions CRUD
  php bin/console make:crud
- Utiliser l'injection de dépendances pour accéder à `EntityManagerInterface`.

7. Utilisation de Twig pour les vues

- Utiliser la fonction `render()` dans le contrôleur pour retourner les vues Twig.
- Utiliser les fonctions `path()` les liens et `asset()` pour les images dans les templates Twig.
- Implémenter des conditions et des boucles pour afficher dynamiquement les données.

8. Gestion des assets avec Webpack Encore

- Installer Webpack Encore :
  composer require symfony/webpack-encore-bundle
- Désinstaller AssetMapper
  composer remove symfony/asset-mapper
- Installer Bootstrap
  npm install bootstrap --save-dev
- Installer les dépendances front-end :
  npm install bootstrap sass-loader@^16.0.1 sass —save-dev
- Configurer Webpack Encore dans le fichier `webpack.config.js et dé-commenter la partie saas`
- Configurer le framework JS stimulus avec WebPackEncore
  npm install @symfony/webpack-encore @hotwired/stimulus —save-dev
- Configurer Webpack Encore dans le fichier `webpack.config.js`.
- Générer les assets :
  npm run watch

9. Générer des formulaire

- Créer une entité contact pour créer un formulaire de contact :
  php bin/console make:entity
- Créer le formulaire (ContactType)
  php bin/console make:form
- Configurer les champs de formulaire dans ContactType
- Créer la vue qui affiche le formulaire :
  php bin/console make:controller ContactController
- Afficher le formulaire dans la vue générée

- Appliquer le thème Bootstrap à votre formulaire
  {% form_theme contactForm 'bootstrap_5_layout.html.twig' %}
- Installer Symfony Validator pour la validation des champs au niveau des classes ou de ContactType :
  composer require symfony/validator
- Afficher un message flash si la soumission du champ est valide sinon afficher les erreurs

10. Mise en place de la sécurité et de l'authentification

- Installer le bundle de sécurité :
  composer require symfony/security-bundle
- Créer une entité `User` :
  php bin/console make:user
- Configurer le système de sécurité dans le fichier `config/packages/security.yaml`.
- Générer les formulaires de connexion et d'inscription :
  composer require symfonycasts/verify-email-bundle
  php bin/console make:registration-form
  php bin/console make:security:form-login
- Vérifier l'utilisateur connecté avec `app.user` et `is_granted()`.
- Installer maildev en local pour recevoir des emails
  npm i -g maildev
  .env.local : MAILER_DSN=smtp://localhost:1025?verify_peer=0
  maildev -v --ip 127.0.0.1

11. Créer une navigation dynamique

- Afficher inscription/connexion si pas connecté ou profile si connecté dans votre nav
  {% if app.user %}

12. Créer un backOffice avec EasyAdmin

- Afficher inscription/connexion si pas connecté ou profile si connecté dans votre nav
  composer require easycorp/easyadmin-bundle
  php bin/console make:admin:dashboard
  php bin/console make:admin:crud

Ressources supplémentaires :

- Documentation officielle de Symfony : https://symfony.com/doc/current/index.html