1. **Installer PHP**
2. **Installer COMPOSER**

DL Composer sur <https://www.getcomposer.org/download>

Exécuter le setup tout accepter

Vérifier la version sur VScode -> **composer –v**

1. **Installer Xampp**
2. **Installer Symfony**
3. **symfony check:requirements** -> permet de vérifier si tt est ok, sinon il faut installer ce qu’il demande dans php.ini

**Initialiser votre projet symfony :**

Dans le terminal insérer cette ligne et modifier my\_project\_directory par le nom de votre appli :

symfony new my\_project\_directory --version="7.1.\*" –webapp

Symfony est structurée sur un pattern MVC (Model View Controller), constitué par ces dossiers :

Dans **src** :

* **Entity/Repository** -> Model
* **Controller** -> Controller

Dans racine du projet :

* **Templates** -> les Views

Le fichier **composer.json** : contient le nom des librairies dont dépend le projet pour fonctionner avec leur version : Les modes d’emplois de ces librairies sont installées dans **composer.lock**. NE JAMAIS Y TOUCHER.

Le dossier **config** contient tous les fichiers de config des librairies (bundles) propres à Symfony et les librairies tierces.

Le fichier **.env** est un fichier de configuration de « secrets », ou de données de connexion aux API, à votre bdd....

Une fois le dossier de votre appli créé, dans les onglets de navigation de VS Code, cliquer sur Fichiers puis ouvrir le dossier et choisissez votre appli.

Ouvrez Xampp et lancez apache(serveur) et MySQL (sera notre gestionnaire de bdd).

Dans le terminal insérez : symfony serve pour lancer le serveur Symfony, puis cliquez sur l’url du serveur local -> ouvre la page d’accueil de votre projet Symfony dans votre navigateur par défaut.

Page de présentation de symfony classique car le symfony debug (composant natif de symfony installé avec –webapp) est activé et qu’il n’y a pas encore de controller pour la homepage.

* Ici parler de git et faire l’installation SB.

**Créer un controller symfony :**

La doc : <https://symfony.com/bundles/SymfonyMakerBundle/current/index.html>

Le maker est un composant symfony permettant de créer des composants pour le projet, ici ce sera un controller :

**Symfony console make :controller** -> HomepageController

Cela va créer un fichier HomepageController.php dans le dossier src/Controller et un dossier Homepage dans le dossier Templates avec un fichier nommé index.html.twig.

Un controller est un fichier php (< ?php) , situé dans le dossier src (chemin : App\Controller), qui utilise des « use », des méthodes disponibles dans des Class (ex : AbstractController) situées dans des fichiers internes de symfony (trouvables dans le dossier vendor).

Un controller est aussi une Class qui porte le nom de votre controller et qui peut étendre d’autres classes.

Dans cette Class, on peut y trouver des méthodes pour exécuter une/des action(s) sur une route spécifique (Voir requête http pour infos) et restituer une vue une vue au navigateur après la réponse du serveur -> La vue c’est le fichier twig situé dans le template associé.

**Twig c’est quoi ?**

La doc : <https://symfony.com/doc/current/templates.html#twig-templating-language>

Pour faire simple twig permet de créer des templates html concis, d’imbriquer des templates ensembles, d’utiliser des variables php et d’autres fonctions...

Les écritures spécifiques à twig :

**{% %}**  -> défini un block twig.

**{{ }}** -> permet l’affichage d’une variable définie en global ou en paramètre dans la route de rendu.

**{# #}** -> permet d’écrire un commentaire.

Si on rafraichit la page d’accueil rien ne se passe c’est normal -> Modifier /homepage en / (route classique des homepage) et on envoi shopname en variable -> Voir le résultat localhost/

**Création d’un article :**

Pour créer un article, on va utiliser les CRUD (des méthodes create, read, update et delete dans les controllers). Sauf que pour utiliser CRUD, il faut avoir un modèle, donc une entity (on a besoin des propriétés des objets).

On commence donc par créer une première Entity -> Brand (Marque des chaussures).

**Symfony console make :entity** Brand et suivre les instructions (nom de propriété, type de propriété, possible d’être null en bdd...).

Observer la structure du fichier avec les props, les getter et setter de chaque propriété. (Ici voir PHP POO)

Voir le Repository associé à l’Entity.

La spécificité de symfony c’est auussi les ORM (Object Relational Mapping) avec Doctrine, gestionnaire de datas entre symfony et la bdd.

La doc : <https://symfony.com/doc/current/doctrine.html>

Avant de créer nos tables en bdd, il faut configurer la bdd dans le .env :

DATABASE\_URL="mysql://root@127.0.0.1:3306/shopshoes?serverVersion=10.4.28-mariaDB"

Puis créer cette bdd via symfony -> **Symfony console doctrine:database:create**

Vérifier la présence de votre bdd dans admin de mySql de Xampp.

Ensuite on va pouvoir injecter cette nouvelle entity dans notre bdd-> **Symfony console make:migration** -> voir le dossier migrations pour expliquer le fichier créer -> fichier de requête Sql.

Ensuite effectuer la migration -> **Symfony console doctrine:migrations :migrate**

En rafraichissant phpmyadmin, on constate la création de table Brand dans notre bdd.

**Utilisation de CRUD** :

**Symfony console make:crud** -> entity => Brand

Controller : Admin**\**BrandController

Expliquer les dossiers et fichiers créés : Form, controller et templates associés.

Montrer le rendu de la route admin/brand -> aucun css appliqué, on va commencé a styliser avec le built theme de BootStrap inclus de symfony.

La doc : <https://symfony.com/doc/current/form/form_themes.html#symfony-built-in-form-themes>

Bootstrap5\_layout le mieux, ajouter dans le base.twig le css et js de bootstrap sinon ca marche pas.

On stylise un peu le create -> create c’est new + form -> donc expliquer maintenant le form, et le twig.

**REGLES CSS :**

EN BOOTSTRAP :

-1 = 0.25 rem ; = 4px ;

-2 = 0.5 rem ; = 8px ;

-3 = 1 rem ; 1rem = 16px ;

-4 = 1.5 rem ; = 24px ;

-5 = 3 rem ; = 48px ;

On ajoute une class container de BS et un mt-5 pour ajuster le tout.

*Pour styliser, on utilise le dossier assets/style et app.js pour les imports. (pour comprendre voir le log de la console dans le navigateur).*

Enfin on aborde le Form de Brand et ses props.

Enfin on créer une marque sans photo, bien mettre imageName en required false et **expliquer les # méthodes du controller** -> et le retour à l’index.

I**ci aborder le role des controllers et leur facon de renvoyer a la vue les variables avec les methodes du repository (voir CategoryRepository -> findAll, find, findOneBy.... pour évoquer la boucle faite dans la vue)**