# Symfony


**Symfony** est un framework PHP. Il permet d'aider à développer les sites web plus rapidement et plus efficacement. C'est avant tout un outil qui permet de ne pas devoir écrire continuellement les mêmes codes répétitifs et de se concentrer sur les fonctionnalités principales d'une application web. Par exemple, Symfony nous aide à créer la connexion de l'utilisateur. Symfony utilise la programmation orientée objet qui est un modèle qui permet de définir ainsi que de faire interagir des objets ensembles. On va alors pouvoir créer des classes qui vont définir des méthodes et qui pourront être utilisées tout au long du projet.

Lorsque l'on crée un projet Symfony, des dossiers et des fichiers sont crées avec une arborescence qui est déjà définie. Symfony utilise une architecture MVC.

- M pour **Model** qui va contenir le traitement des données
- V pour **View** qui va consister à donner une vue à l'utilisateur grâce à des templates
- C pour **Controller** qui va permettre d'écrire du code et ainsi faire un lien avec le Model en obtenant des données et la View pour afficher un visuel.

# Installer Symfony

## Installation de Composer

Composer est un gestionnaire de dépendances pour PHP. Il facilite la gestion des bibliothèques et des dépendances externes dans les projets PHP.

**Documentation** :

https://getcomposer.org/download/

**Étapes d'installation** :

Ligne de commande à écrire dans le terminal.

```
php -r "copy('https://getcomposer.org/installer', 'composer-s
php -r "if (hash_file('sha384', 'composer-setup.php') === 'e2
php composer-setup.php
php -r "unlink('composer-setup.php');"
sudo mv composer.phar /usr/local/bin/composer
```

Tester d'écrire **composer** dans votre console.

## Installation du CLI de Symfony

```
curl -sS https://get.symfony.com/cli/installer | bash
```

Après cette commande, il est possible que votre console vous dise de bouger vos fichiers pour que vous puissiez utiliser la commande symfony de partout, si c'est le cas, faites-le.

### Créer un nouveau projet

```
symfony new nom_de_app --version="7.0.*@dev" --webapp
```

Ici on précise la version que l'on souhaite utiliser et que l'on veut être en mode dev.

Vous avez maintenant un nouveau dossier qui porte le nom que vous avez donné à l'app.

# Mettre Bootstrap sur Symfony

```
composer require symfony/webpack-encore-bundle
```

```
npm -v

Si npm n'existe pas encore :
sudo apt install npm
```

```
npm install @symfony/webpack-encore --save-dev
```

```
npm install bootstrap
```

Ajouter dans le fichier webpack.config.js en dessous du .addEntry déjà existant.

```
.addEntry('vendor', [
        'bootstrap/dist/css/bootstrap.css',
        'bootstrap'
        // ... autres dépendances
    ])
```

Ajouter dans le <head> de la base.html.twig.

```
<link rel="stylesheet" href="{{ asset('build/vendor.css') }}"
<script src="{{ asset('build/vendor.js') }}" defer></script>
```

On va build tout ça.

```
npm run build
```

Si on a bootstrap d'installer, lancer le server symfony ne sera pas suffisant, il va falloir lancer aussi npm dans un autre terminal.

```
npm run watch
```

## A savoir

Maintenant, il est possible de changer facilement les couleurs de Bootstrap.

Aller dans node-module/bootstrap/scss/_variables.scss. La dedans vous avez toutes les couleurs de bootstrap et vous pouvez les changer à votre guise.

## Bootstrap pour formulaire

```
# config/packages/twig.yaml
twig:
```

```
    form_themes:
        - 'bootstrap_5_layout.html.twig'
```

# Structure de Symfony

Voyons un peu la structure de notre dossier crée par Symfony.

bin :  contient les dossiers d'exécution, on n'y touche pas.

config : comme son nom l'indique, il contient des fichiers de configuration. Ce dossier nous sera utile.

migrations : contient les différentes requêtes passées à la base de données.

public : répertoire racine. Contient les images, le CSS et le JS.

src : contient les codes de l'application. Notamment les controllers, les entités de la BDD et les form.

templates : les fichiers Twig qui vont servir à la Vue du site.

translations : contient les traductions, si vous en faites.

var : stock des données. On y touche pas.

vendor : dépendances externes du projet. On y touche pas non plus.

.env : fichier de configuration de l'environnement. C'est ici que vous donnez les infos sur la BDD.

## Le Controller et la Vue

Symfony possède des templates qui lui servent à rendre une vue à l'utilisateur, **Twig** nous permet en partie d'intégrer la maquette. C'est un langage de template qui facilite l'intégration du langage PHP, il est rapide, sécurisé et flexible. Grâce à Twig, il est possible de créer un fichier de base de template, qui est nommé nativement par Symfony base.html.twig et qui va nous permettre de créer des blocs

que l’on va pouvoir mettre sur toutes les pages que l’on souhaite seulement en les appelant.

Sur cette base, il est donc possible de construire la barre de navigation ainsi que le pied de page, comme cela, on peut les intégrer sur tous les autres templates facilement et rapidement. Twig permet aussi de faire de l’algorithmique de base en faisant des boucles et des conditions, ce qui est très pratique lorsque l’on crée un site web d’e-commerce. Les **controllers** permettent de gérer une partie de code. Ces controllers sont ainsi reliés la plupart du temps à des templates Twig auquels ils pourront transmettre des variables qui pourront être utilisées sur les templates Twig.