# Symfony : Comprendre le Framework !

### Tout ce qui sera écrit en orange seront des lignes de commande à faire dans le terminal (Windows)

### Quelques commandes pour se déplacer dans le terminal de Windows :

* **cd ou chdir ⬄ Affiche le nom du répertoire courant ou change le répertoire courant**
* **cls ⬄ Efface le contenu de la fenêtre d’invite de commande.**
* **copy ⬄ Copie un fichier d’un emplacement à un autre.**
* **del ⬄ Supprime un ou plusieurs fichiers**
* **dir ⬄Affiche la liste des fichiers et sous-dossiers contenus dans le répertoire.**
* **exit ⬄ Quitte le programme cmd.exe ou le script batch courant.**
* **md ou mkdir ⬄ Crée un répertoire ou un sous-répertoire.**
* **move ⬄ Déplace un fichier d’un dossier à un autre.**
* **rd ou rmdir ⬄ Supprime un répertoire.**
* **ren ou rename ⬄ Change le nom d’un fichier ou d’un ensemble de fichiers.**
* **php -v** ⬄ Permet de voir la version PHP installer sur notre machine
* **composer -V** ⬄ Pour vérifier la version de composer et donc si composer fonctionne bien , qu’il existe et qu’il est bien installer sur la machine.

Une fois que les 2 points précédent soit ok on peut donc commencer à découvrir Symfony.

# Créer une application Symfony avec composer.

Se rendre dans un dossier où l’ont veut créer le projet, on est pas obligé d’aller dans le dossier www de wamp car on ne va pas travailler avec apache pour faire tourner Symfony mais avec un serveur interne.

* **composer create-project symfony/website-skeleton demo**

Create-project ⬄ Créer une structure de dossiers/fichiers à partir d’un modèle.

Symfony/website-skeletin ⬄Modèle d’un projet de site web symfony.

Demo ⬄ Le nom de mon nouveau projet.

Je dois ensuite me rendre (avec le terminal) dans le dossier où j’ai créé mon application.

* **cd demo**

L’ouvrir avec mon éditeur de code (Visuel Studio code)

* **code .**

Une fois ouvert, je ne vais plus utiliser mon invite de commande externe, mais je peux maintenant commencer à utiliser directement au sein même de mon Visuel Studio Code pour taper toute sortes de commande.

Pourquoi toute sortes de commande ? Car Symfony c’est beaucoup de fichiers et beaucoup de codes. On peut tout faire à la main mais les développeurs de Symfony nous ont fournis la CLI ⬄ Command light interface. C’est un petit assistant qu’on peut appeler par des lignes de commandes et qui va nous aider à gérer notre application Symfony.

# Installer un serveur personnaliser pour lancer mon application Symfony

* **composer require server --dev ⬄** Que je ne vais utiliser que lorsque je vais développer, le jour où je vais passer mon site en prod, tu n’auras pas besoin d’installer cette librairie là.

Composer require [xxx]⬄ Permet d’installer une librairie et de l’utiliser dans le projet !

A la fin de l’installation, on va pouvoir lancer notre application dans notre navigateur.

* **php bin/console server:run**

Ça va lancer notre serveur sur localhost :8000

Notre application ne tourne donc pas sur le port 80, le port 80 c’est pour mon apache/wamp si je veux accéder à d’autres projets qui tourne sur wamp par exemple ou à phpmyadmin.

**ATTENTION Dépendamment de la version de Symfony utilisé, la commande précédente affichera une erreur mais on peut toujours lancer le serveur avec la commande suivante :**

**Depuis la version 4.4 de Symfony il faut Aller sur** [**https://symfony.com/download**](https://symfony.com/download) **et télécharger et installer le logiciel Symfony (Bouton Download). Re démarrer le pc, se rendre dans le terminal à l’endroit où se trouve le projet et on peut maintenant lancer le serveur avec :**

* **server :start**

## Commençons par inspecter notre dossier.

On va s’intéressé particulièrement au dossier **src** qui va contenir l’entièreté des **codes** **php** de notre application ainsi qu’au dossier **template** qui lui va nous permettre de **gérer** nos **fichiers d’affichage**/rendu avec **twig**.

***Ouvrir un 2eme onglet de terminal dans Visuel Studio Code pour ce qui va suivre !***

Pour créer un controller qui va se retrouver au sein du dossier controller.

* **Php bin/console make :controller**

make:controller ⬄ Permet de fabriquer un controller simplement !

Choisir un nom pour la classe controller : 1ere lettre en Maj et le reste en camelCase

* **BlogController**

**Un gros bouton vert avec le mot « Success ! » apparait.**

Ma ligne de commande à créer dans le dossier des controllers un fichier qui s’appelle **BlogController.php** et qui **contient** une **classe** et dans notre dossier templates un dossier qui s’appelle **blog** avec déjà un premier fichier qui s’appelle **index.html.twig** pour qu’on puisse tout de suite commencer à travailler.

# Les Annotations

Inspections ce nouveau fichier BlogController.php

N’importe quel commentaire qui commence par un **@** est une **annotation.**

Ici, **cette annotation s’appelle route**, on explique à Symfony que **quand un navigateur appellera mon site.com/blog , voici la fonction que tu dois appeler. La fonction index,** peut importe ce qu’elle fait.

Ici, notre **but dans cette fonction est de traiter la demande et de renvoyer une réponse. Ici, je renvoie le fait d’afficher un fichier html qui s’appel index.html.twig et qui se trouve dans le dossier blog.** On remarque que on a pas préciser que ce dossier se trouve dans le dossier template car Symfony sait que tout ses templates sont dans ce dossier-là. Donc on ne met que blog/index.html et on comprend que c’est bien **ce fichier twig qui nous permet de rendre un affichage.**

## Première page avec Symfony !

Se rendre sur : <http://localhost:8000/blog>

Je vois que Symfony m’a créé une nouvelle page, elle me dit que si je veux **modifier les traitement** pour cette page ça se trouve **dans BlogController.php**, si je veux **modifier l’affichage** pour cette page ça se trouve **dans blog.index.html.twig**: test : Changer le mot Hello en salut !

## Création de l’ensemble des routes qui serviront à ce blog.

Se rendre dans le BlogController.php et créer la fonction public home. Car je veux que ce soit ça qui soit appeler quand je vais sur mon-site.com/**rien** Donc, ça sera la page d’accueil du site.

# Langage Twig

## Simplicité

-Facilite l’écriture des affichages

-Apporte beaucoup de fonctionnalités (que le templating php avait du mal à nous apporter).

## Absence de PHP

-Permet d’abstraire les affichages de balises PHP

-Plus simple pour un intégrateur

Twig existe en dehors de Symfony !