Préparer l’environnement et la création d’un projet Symfony

Table des matières

I. Contexte

Durée : 60 minutes.

Environnement de travail :

- Ordinateur avec Windows, macOS ou Linux.
- Éditeur de code *(PHPStorm ou Visual Studio Code sont conseillés).*
- Un navigateur récent : Firefox, Chrome, Safari, Opera.
- Si possible PHP > 7.2.5 installé avec les librairies/extensions suivantes :
    - Iconv,
    - JSON,
    - PCRE,
    - Session,
    - Tokenizer.
- Composer.
- Symfony CLI.

Prérequis : connaître le langage PHP, SQL, et avoir des notions informatiques pour utiliser un terminal.

Objectifs de la partie :

- Savoir installer une stack backend complète.
- Créer et maîtriser l'architecture d'une application Symfony.
- Développer sa première page web.

Contexte

Dans ce cours, nous allons installer tous les prérequis pour développer notre projet sous Symfony.

Mais Symfony, qu'est-ce que c'est ? C'est un framework open source français créé par l'entreprise SensioLabs, dont le but est de transformer une requête HTTP en réponse HTTP.

Le tout, grâce à un ensemble de composants Symfony comportant des classes écrites en PHP objet disponibles directement depuis github.com[1].

Cela permet de développer un projet avec un code plus robuste, plus sécurisé et de disposer d'un cadre de bonnes pratiques.

C'est pourquoi vous installerez *via* ce cours bon nombre d'outils utiles au traitement du langage PHP et à l'accès aux bases de données relationnelles.

II. Installation d'une stack AMP : Apache, MySQL, PHP

A. Introduction

Une stack AMP est un ensemble de logiciels open source à installer, de sorte à pouvoir héberger des sites dynamiques développés en PHP qui interagissent avec des bases de données.

L'acronyme AMP signifie **A**pache, **M**ySQL et **P**HP.

Il existe des stacks similaires comme pour Linux *(LAMP),* Mac *(MAMP)* ou encore Windows *(WAMP)* constituant un seul et même programme à installer, plutôt qu'un par un.

---

1 https://github.com/symfony/symfony

XAMPP quant à lui est cross-platform (X) et possède Perl en plus pour le FTP.

Mais nous allons installer chacun de ces logiciels individuellement pour les configurer correctement et comprendre chacune des étapes.

Il est fort probable que vous ayez déjà installé l'ensemble, mais si ce n'est pas le cas vous pouvez suivre ces différentes parties !

B. Installer Apache2

Apache2 est l'un des serveurs web les plus utilisés avec NGINX.

Open source et connu mondialement, Apache2 est un logiciel qui permet de fournir des contenus web. Cela peut être des pages HTML ou des ressources comme des images et des vidéos.

Le principe est simple, Apache2 se trouvant sur votre ordinateur ou sur un serveur, reçoit la requête HTTP du client *(exemple : un navigateur ou un téléphone)* et la transmet correctement à votre projet Symfony afin de la traiter et de retourner une réponse HTTP.

Il y a différentes méthodes d'installation d'Apache2 selon votre système d'exploitation.

Méthode **Configurer Apache2**

Windows

Si vous êtes sous Windows, il vous suffit de télécharger la dernière version d'Apache2 depuis ce lien : apachelounge.com[1].

Une fois le .zip téléchargé, vous pouvez l'extraire à la racine de votre « *Disque Local (C:)* ».

Il vous faudra ouvrir un invité de commande avec les droits d'administrateur et installer le service « *httpd* » au sein de votre système comme ci-dessous :

```
cd C:\Apache24\bin
httpd -k install
```

Une fois l'installation terminée, sachez que vous pouvez démarrer ou éteindre Apache2 à tout moment en ouvrant le « ***Gestionnaire de services*** » de Windows (appuyez sur la touche « ***Windows + R*** », saisissez « ***services.msc*** » et appuyez sur « ***Entrée*** »). Recherchez le service « *Apache2.4* » et utilisez les options du menu pour contrôler le service. Pour vérifier si Apache2 est allumé, accédez depuis un navigateur à l'adresse http://127.0.0.1[2].

Linux ou macOS

Si vous êtes sur Linux ou sur Mac *(en utilisant Homebrew plutôt que le gestionnaire Apt),* il vous faudra renseigner ces quelques lignes de commandes dans votre terminal :

```
sudo apt update
sudo apt install apache2
sudo ufw allow in "Apache"
```

Apache2 étant installé, vous pouvez accéder à l'adresse http://127.0.0.1[3] représentant votre serveur web local.

À ce stade-là, votre serveur ne peut que renvoyer des pages web HTML simples et des ressources.

C'est pourquoi nous allons installer PHP et MySQL pour pouvoir développer notre projet !

1 https://www.apachelounge.com/download/
2 http://127.0.0.1/
3 http://127.0.0.1/

C. Installer MySQL

Méthode **Installer et configurer MySQL**

Maintenant que votre serveur web tourne, il vous faut installer une base de données pour pouvoir y stocker les données de votre site.

Ça tombe bien, c'est exactement le rôle de MySQL !

Étant une base de données relationnelle, MySQL vous permettra de créer des tables, avec différentes colonnes pouvant être reliées entre elles via des relations.

Et avec PHP et Symfony, vous pourrez interagir avec votre base de données pour lire, enregistrer, modifier ou supprimer des informations !

Windows

Si vous êtes sous Windows, vous pouvez vous rendre sur le site dev.mysql.com[1] et télécharger le premier **Windows (x86, 32-bit), MSI Installer**.

Linux ou macOS

Via le gestionnaire de paquet Apt sur Linux ou Homebrew sur macOS, vous pouvez installer MySQL avec la commande suivante :

```
sudo apt install mysql-server
```

Vous serez probablement invité à définir un mot de passe pour l'utilisateur « *root* ».

Assurez-vous donc de renseigner un mot de passe sécurisé et de le conserver.

Pour rappel, « *root* » est un super utilisateur ayant tout pouvoir.

Celui-ci est présent par défaut lors de l'installation, alors attention à ce que vous écrivez !

Après l'installation, vous pouvez exécuter un script de sécurité fourni par MySQL qui vous guidera sur la configuration de votre mot de passe, la désactivation de l'accès à distance pour l'utilisateur « *root* » ou encore la suppression des utilisateurs anonymes.

```
sudo mysql_secure_installation
```

Pour vérifier le statut du service MySQL ou vous connecter à celui-ci, vous pouvez :

```
sudo systemctl status mysql
sudo mysql -u root -p
```

Renseignez alors le mot de passe que vous avez défini.

D. Installer PHP

Méthode **Installer et configurer PHP**

Nous avons le serveur web, la base de données, il ne nous manque plus qu'à installer PHP !

Le logiciel PHP *(avec PHP-fpm)* permettra de lire le code PHP que vous écrivez.

En effet, à la différence d'un langage compilé *(comme C, C++, C#, Java),* PHP ne transformera pas votre code en exécutable, mais l'interprétera en temps réel, c'est l'un des grands atouts d'un langage de scripting !

Windows

Si vous êtes sous Windows, vous pouvez vous rendre sur le site windows.php.net[2] et télécharger le premier ZIP **VS16 x64 Non Thread Safe** d'une trentaine de méga-octets.

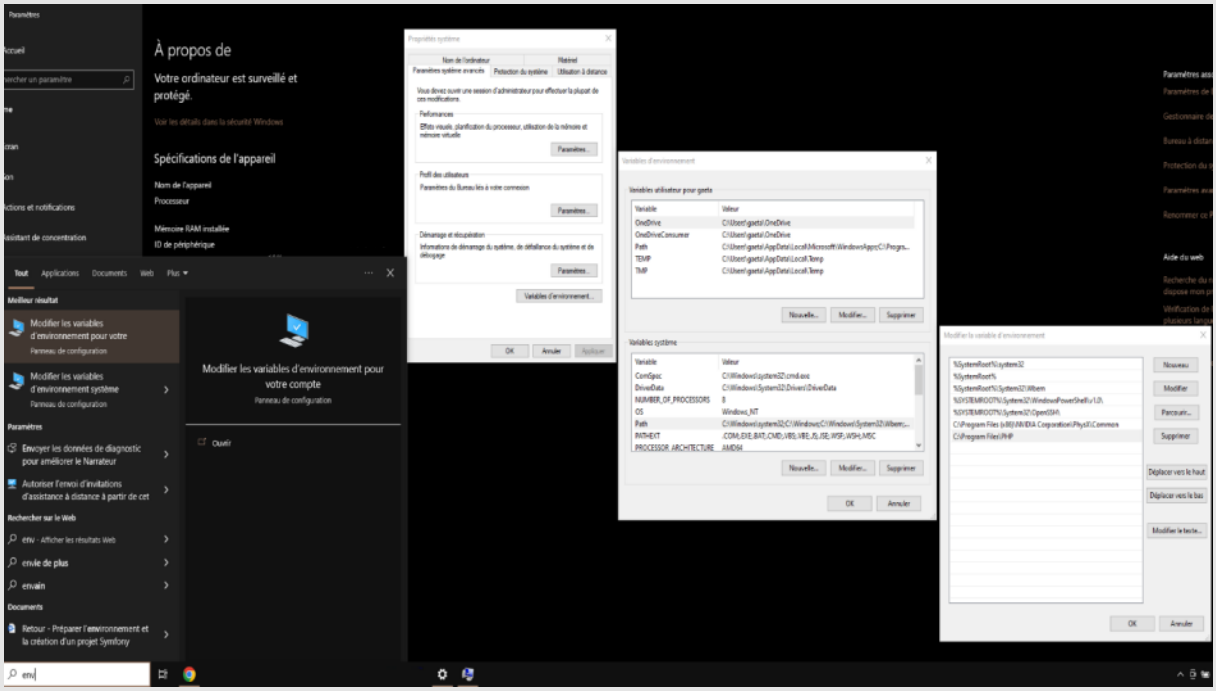
Également, une fois PHP configuré, vous pouvez ajouter à vos variables d'environnement le binaire PHP.

1 https://dev.mysql.com/downloads/installer/
2 https://windows.php.net/download/

Une variable d'environnement permet à votre système Windows, et notamment aux programmes/processus, d'interagir avec elles et leurs valeurs. Ici, notre variable permettra de savoir où chercher lorsque vous vous utiliser PHP.

Nous allons donc, depuis la barre de recherche Windows, réaliser les actions suivantes :

- Taper « ***env*** ».
- Cliquer sur « ***Modifier les variables d'environnement*** ».
- Cliquer sur « ***Variables d'environnement*** ».
- Double-cliquer sur la variable système « ***PATH*** ».
- Ajouter *via* le bouton « ***nouveau*** » le chemin de votre binaire PHP. Par exemple : « *C:\Program Files\PHP* ».
- Puis enregistrer l'ajout !



Linux ou macOS

Via le gestionnaire de paquet Apt sur Linux ou Homebrew sur macOS, vous pouvez installer PHP avec la commande suivante :

```
1 sudo apt install php libapache2-mod-php php-mysql
```

Une fois l'installation terminée, n'hésitez pas à exécuter la commande suivant pour confirmer le numéro de version et l'installation réussie :

```
1 php -v
```

E. Configurer la stack AMP

Vous voilà avec les logiciels nécessaires pour héberger vos projets PHP !

Il ne nous reste plus qu'à finir de configurer l'ensemble et de nous assurer que la stack AMP est correctement installée.

Pour le moment seuls les fichiers .html sont compris par Apache2. Il nous faut donc indiquer que PHP doit aussi être géré par Apache2.

Windows

Assurez-vous qu'Apache ne tourne pas.

Pour permettre à Apache de lire vos fichiers PHP, vous devez faire un peu de configuration !

- Accédez au fichier « *C:\Apache24\conf\httpd.conf* ».
- Ajoutez le code suivant tout en bas du fichier :

```
PHPIniDir "C:/php"
LoadModule php_module "C:/php/php8apache2_4.dll"
AddType application/x-httpd-php .php
```

Également :

- Cherchez la ligne <IfModule dir_module> et ajoutez le mot « *index.php* » entre « *Directoryindex* » et « *index.html* ».
- Sauvegardez le fichier et redémarrez Apache2.

```
<IfModule dir_module>
    DirectoryIndex index.php index.html
</IfModule>
```

Linux ou macOS

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

Ensuite :

- Cherchez la ligne <IfModule dir_module> et ajoutez le mot « *index.php* » entre « *Directoryindex* » et « *index.html* ».
- Sauvegardez le fichier et redémarrez Apache2.

```
sudo systemctl reload apache2
```

Sous Windows, pour tester votre installation, vous pouvez créer votre premier fichier au format .php dans votre dossier « *C:\Apache24\htdocs* » nommé « *info.php* » contenant le code suivant ou sous Linux et Mac, directement dans votre dossier « */var/www/info.php* » :

```
<?php
phpinfo() ;
```

Une fois terminé, accédez à votre adresse http://127.0.0.1/info.php pour voir votre nouvelle page PHP renvoyée par Apache2 !

III. Création d'un projet Symfony avec Composer

A. Création d'un projet Symfony avec Composer

Avant d'installer un projet Symfony avec Composer, vous devez vous assurer d'avoir celui-ci installé sur votre machine.

Pour rappel, Composer est un gestionnaire de dépendances PHP. Il vous permettra de télécharger et d'importer n'importe quel composant PHP dans votre projet.

Pour cela, accédez à getcomposer.org[1] vous permettant de télécharger le .exe de Windows ou de jouer les commandes suivantes pour Linux et macOS :

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'e21205b207c3ff031906575712edab6f13eb0b361f2085f1f1237b7126d785e826a450292b6cfd1d64d92e6563bbde02')
 { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php');
} echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Méthode **Créer un projet Symfony**

Pour créer un projet Symfony, vous avez la possibilité d'utiliser Composer ou Symfony CLI symfony.com[2], un binaire fourni par SensioLabs basé sur Composer vous permettant d'avoir quelques commandes en plus.

Si vous disposez de Symfony CLI, vous pouvez également tester votre environnement de travail pour voir s'il est adéquat avec votre projet Symfony avec la commande :

```
symfony check:requirements
```

Dans le cadre de nos cours, nous allons créer un projet ayant, comme base, un projet minimal nommé « ***skeleton*** ». Il existe également la base complète « ***webapp*** » qui permet d'avoir toutes les ressources nécessaires pour développer une application web *(envoi d'emails, création des vues, accès à la base de données),* mais nous allons partir d'un projet pratiquement vide et ajouter nos composants petit à petit.

Pour ce faire, positionnez-vous dans le dossier où vous souhaitez créer votre projet, puis créez votre projet avec Composer :

```
composer create-project symfony/skeleton:"^5.4" mon_repertoire_de_projet
cd mon_repertoire_de_projet
```

« La dépendance ***webapp*** *est un pack de composants à installer pour disposer de tous les outils nécessaires au développement d'une application web. »* Il existe d'autres packs officiels permettant d'étendre Symfony sur plein de sujets (ORM, debug, tests, etc.). On peut retrouver ces packs sur le site de *Composer*[3].

1 https://getcomposer.org/download/
2 https://symfony.com/download
3 https://getcomposer.org

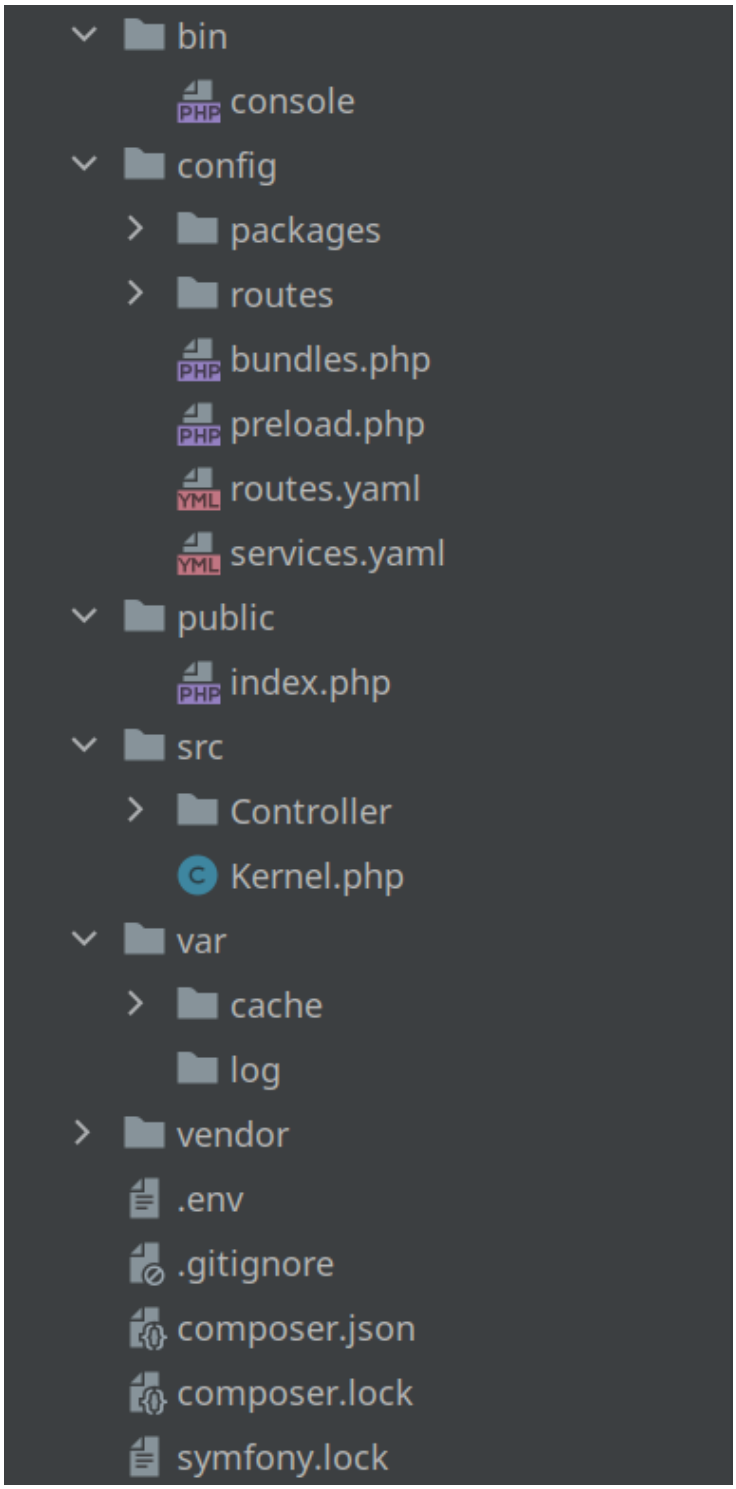
IV. Arborescence d'un projet Symfony et lancement de l'application

A. Structure des dossiers

Maintenant que votre projet a été initialisé, nous allons explorer les différents dossiers et fichiers créés, basés sur le modèle « ***skeleton*** ».

À noter que l'arborescence actuelle est relativement minime et qu'à chaque nouveau composant installé, des dossiers et des fichiers peuvent être créés.

Ouvrez le dossier de votre projet puis *via* un IDE *(un éditeur de code type Visual Studio Code, PHPStorm)* ou votre explorateur de fichier, visualisez les éléments suivants :



- **bin**

  Contient les exécutables de votre projet comme la console ou seulement après avoir installé le composant PHPUnit, le binaire phpunit pour tester votre application.

  Vous utiliserez souvent la console avec la commande «*php bin/console <argument>*», mais ne coderez jamais dans ce dossier.

- **config**

  Contient tous les fichiers de configuration de l'application. Cela peut être la configuration des packages installés par Composer ou bien plus globalement celle de Symfony.

  Il vous arrivera d'adapter certains composants à votre projet en éditant quelques fichiers de configuration.

- **public**

  Contient le « *front controller* » de votre application. C'est-à-dire le premier fichier qui sera appelé lors d'une requête transmise par Apache2.

  Ce dossier contient également tout ce que le visiteur du site sera autorisé à voir : fichiers HTML, CSS, js, les images, etc. Le visiteur du site n'a pas accès à ce qui se trouve en dehors de ce dossier.

  Il est donc fondamental de ne rien mettre dans ce dossier qui serait confidentiel.

- **src**

  Contient la logique métier de votre projet.

  C'est le dossier principal de votre application. Tout le code back que vous développerez se trouvera ici. Vous y passerez donc la plupart de votre temps.

- **var**

  Accessible en écriture par le serveur, ce dossier contient les fichiers temporaires, les fichiers de travail, le cache, les sessions, etc. Vous n'avez généralement pas besoin d'aller dans ce dossier, à moins de consulter vos fichiers de logs ou de nettoyer votre cache.

- **vendor**

  Contient tous les dossiers système nécessaires au fonctionnement interne de Symfony.

  Il contient également les dossiers des packages installés par Composer.

  Vous ne travaillerez jamais dans ce dossier, continuellement mis à jour par Composer.

Par la suite, d'autres fichiers et dossiers seront créés par Symfony Flex lorsque vous importerez de nouveaux composants avec Composer, comme : les traductions, Twig, la sécurité (autorisations et authentifications, etc.).

B. Fichiers importants

À la racine du projet se trouvent quelques fichiers importants dont nous allons voir l'utilité.

Le fichier « .env »

Ce fichier « *modèle* » permet de stocker toute la configuration de base de votre projet Symfony.

Il peut éventuellement contenir des informations sensibles comme des mots de passe, l'accès à votre base de données ou encore des clés API.

C'est pour cela que ce fichier ne sera pas versionné et exposé dans votre dépôt Git.

Si vous souhaitez renseigner des informations sensibles dans ce fichier, vous devrez dupliquer celui-ci avec l'intitulé « *.env.local* » qui sera automatiquement ignoré par Git via le fichier « *.gitignore* » également présent dans votre répertoire de travail.

C'est ce fichier surchargeant son fichier parent « *.env* » qui contiendra les mots de passe et autres.

Les fichiers « composer.json » et « composer.lock »

Ces deux fichiers contiennent la liste des dépendances de votre projet.

La différence entre les deux est subtile, mais importante.

- Le fichier ***composer.json*** contient la liste des dépendances de premier niveau (celles dont votre projet a effectivement besoin) avec comme précision une plage de versions acceptables (ni trop ancienne, ni trop récente par exemple). **C'est une sorte de liste de souhaits**.
- Le fichier ***composer.lock***, lui, contient la liste de l'ensemble de toutes les dépendances réellement installées avec la version précise installées ainsi que toutes les sous-dépendances, c'est-à-dire les dépendances des dépendances. **C'est un registre de toutes les dépendances installées dans votre dossier */vendor***

Le fichier « symfony.lock »

Symfony possède un système de recettes appelé *Flex*. Une recette est un ensemble de dépendances déjà configurées pour fonctionner ensemble.

Symfony Flex est donc un outil qui rend l'ajout de nouveaux composants très simple.

Car grâce à Composer et lors des installations et mises à jour du projet, *Symfony Flex* utilisera des recettes proposées par Symfony pour ajouter de nouveaux fichiers, lignes de configurations ou autres.

À ce titre, le fichier *symfony.lock* est le fichier qui tient le registre de ces dépendances installées et des recettes exécutées.

C. Lancement de l'application

Voilà, Symfony est installé. Vous avez compris de quoi il était fait, maintenant vous voulez sans doute savoir à quoi il ressemble. Pour cela, vous allez lancer un serveur PHP afin de pouvoir interpréter le code PHP et afficher vos pages dans le navigateur.

Complément **Serveur interne à PHP**

Si vous avez installé Symfony avec Composer et sans l'aide de Symfony CLI, vous pouvez utiliser le serveur interne de PHP de cette manière :

```
1 php -S 127.0.0.1:8000 -t public/
```

`127.0.0.1`, tout comme « *localhost* », correspond à l'adresse IP locale de votre machine.

`8000` correspond au port sur lequel vous voulez lancer votre application. Vous pouvez choisir celui que vous souhaitez, mais attention cependant, certains ports sont réservés pour des usages précis. Le port par défaut est généralement 8000.

`-t public/` précise à Symfony que le point d'entrée de votre application se trouve dans le dossier *« public/ » (implicitement, c'est le fichier « index.php » qui sera sollicité comme expliqué au préalable dans le cours)*. Il est indispensable de préciser ce paramètre pour lancer une application Symfony via le serveur interne de PHP.

Serveur de Symfony CLI

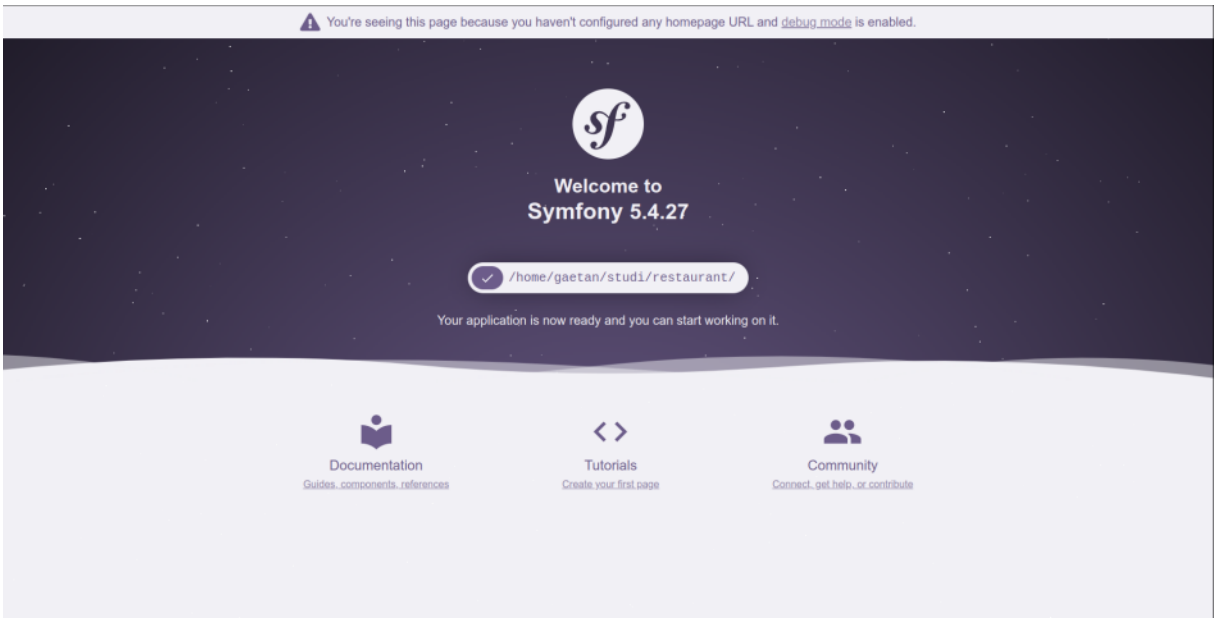
Grâce à Symfony CLI, vous avez, un serveur intégré ainsi que d'autres outils. Pour lancer le serveur, il suffit d'une ligne de commande :

```
1 symfony server:start -d
```

L'option `-d` (pour *daemon)* lance le serveur en arrière-plan.

Dans tous les cas, en vous rendant sur l'adresse http://127.0.0.1:8000[1] vous devriez obtenir cette page, félicitations !

1 http://127.0.0.1:8000/



V. Développer sa première page

A. Développer sa première page

Définition

Symfony est un framework HTTP dont le but principal est de transformer une requête en réponse. Pour cela, plusieurs architectures sont possibles, mais on utilisera le plus répandu d'entre eux : le modèle MVC, pour ***M**odel, **V**iew, **C**ontroller*, qui convient parfaitement pour réaliser des applications web classiques.

C'est une organisation du code qui permet de correctement séparer les responsabilités des portions de codes de notre application. Les models servent à interagir avec notre base de données, les views contiennent le rendu visuel de nos pages et les controllers servent d'intermédiaire et pilotent les interactions entre toutes nos portions de code. Il est en quelque sorte le chef d'orchestre de notre application.

Méthode

Ici, le controller est chargé de connaître les routes disponibles sur votre application et d'exécuter du code lorsque vous vous connectez à cette route.

C'est quoi une route ? C'est un mécanisme liant une URL à une méthode *(ici de controller)* dont l'objectif est de retourner une réponse.

Pour créer votre première route, il vous faut créer votre premier Controller. Pour cela :

1. Rendez-vous dans le dossier *« /src/Controller »* et créez un fichier *« HomeController.php »*.

2. Collez le code suivant dans ce fichier :

```php
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/')]
```

```
11     public function home() : Response
12     {
13   return new Response('Bienvenue sur votre accueil!') ;
14      }
15 }
```

3. Désormais, si vous rafraîchissez la page initiale, vous devriez voir le message « *Bienvenue sur votre accueil !* » s'afficher.

Félicitations, vous venez de créer votre première route avec votre première réponse HTTP !

Exemple

VI. L'essentiel

Pour développer Symfony, il vous faut impérativement avoir un environnement de travail correctement configuré. C'est-à-dire une stack AMP : un serveur web, une base de données et PHP.

De là, il suffit de lancer des commandes avec Composer ou Symfony CLI pour créer une nouvelle application web, démarrer un serveur PHP et interagir avec notre projet via un navigateur.

La plus grande partie de **votre application** trouvera sa place dans le répertoire « *src/* ».

Celui-ci contiendra la logique métier de votre projet.

Pour lancer le serveur de Symfony il suffit de taper la commande `symfony server:start -d` ou de lancer un serveur local avec PHP.

Les routes de notre application seront rattachées à des controllers et retourneront obligatoirement un réponse HTTP.

VII. Pour aller plus loin

Désormais, grâce à votre stack préparée et vos dépendances installées, vous êtes en mesure de pouvoir créer un projet Symfony.

Vous pouvez donc, par la suite, créer d'autres routes et contrôleurs pour structurer votre projet.

- **Contrôleurs** : créez des contrôleurs dont le nom des classes correspond à la logique métier.
- **Routes** : créez plusieurs routes dans ces contrôleurs afin de disposer de plusieurs pages sur votre projet. Vous dessinerez petit à petit la structure de votre application web.