

TP 1 : Installation de l'environnement

Ce premier TP n'est malheureusement pas le plus intéressant mais il est nécessaire pour bien démarrer le développement web, une fois tous les outils installés et configurés, nous pourrons commencer à coder une vraie application pro avec plaisir 😊.



La plupart des outils sont déjà installés sur les machines de l'IUT, vous pouvez relire ce document pour installer le même environnement sur votre (ou vos) machine(s).

Évitez d'utiliser des outils d'intelligence artificielle comme Copilot ou Chat-GPT, bien qu'ils permettent d'être plus productif ils risquent de nuire à votre apprentissage.

Microsoft VS Code

Sortis en 2015 par Microsoft, VS Code est une alternative gratuite du très populaire Visual Studio, ce dernier étant plus orienté à au développement de grosses applications en C/C++/C#.

Extensions

Voici les extensions à installer sur VS Code :

PHP Intelephense

C'est un incontournable pour le développement PHP, Intelephense (ne pas confondre avec l'extension Intellisense) va scanner tous les fichiers PHP de votre projet afin de proposer une auto-complétion très efficace. Elle affiche également les paramètres à envoyer dans une fonction ou un objet,

⚠️ Attention cette extension est gourmande en ressources (RAM, lecture du SDD) car elle scanne en permanence les fichiers sources (vous avez là un prétexte pour demander un PC gamer à Noël).

Twig Language

Nous utiliserons le moteur de template Twig pour écrire le contenu de nos pages. Cette extension va simplement ajouter de la colorisation syntaxique pour ce langage.

PHP 8 Getter & Setter

Dans une classe, les paramètres sont souvent déclarés en « private » pour des questions de sécurité. Pour lire et écrire dans ces valeurs il est recommandé d'écrire des accesseurs, cette extension permet de les générer d'un simple clic.

Docker

Docker est un outil permettant d'empaqueter des outils dans un conteneur. Le principal avantage est de pouvoir « simuler » un serveur avec les mêmes caractéristiques que le serveur qui va héberger l'application en production. Cela évite des problèmes de compatibilité entre les versions.

Nous utiliserons Docker seulement pour la base de données de l'application. Téléchargez et installez la version AMD64 de Docker Desktop sur votre machine : <https://www.docker.com/products/docker-desktop/>

PHP

Bien entendu, nous aurons besoin d'une version récente de PHP (8.2 ou 8.3). Vous pouvez télécharger PHP sur cette page :
<https://windows.php.net/download#php-8.3>

Choisissez une version ‘Non Thread Safe’ en Zip, décompressez le dossier (par exemple dans C:/php/php-8.3) et configurez la variable d’environnement PATH pour pointer vers ce dossier.

Composer

Composer va servir de chef d’orchestre pour les dépendances PHP. Pour tester si Composer est bien installé sur votre machine, exécutez la commande suivante :

```
composer -v
```

Si Composer n'est pas installé, RDV sur le site :

<https://getcomposer.org/download/>

 Note: après installation la commande peut ne pas être reconnue, dans ce cas fermez votre terminal et ouvrez le à nouveau.

NPM

NPM est développé avec NodeJS, assurez vous dans un premier temps de l'avoir installé sur votre machine.

<https://nodejs.org/en/download/prebuilt-installer>

Prenez-bien l'onglet “Prebuilt Installer”.

Ensuite la commande pour vérifier qu'il est bien installé et à jour:

```
npm -v
```

Symfony CLI

Apporte des commandes pour faciliter le développement d'application, Symfony CLI offre également un serveur web préconfiguré. Tout d'abord, entrez la commande suivante pour vérifier si Symfony CLI est installé :

```
symfony -v
```

Si un message d'erreur « commande non reconnue » s'affiche, il n'est pas installé.

RDV sur la page suivante : <https://symfony.com/download>

Vous pouvez utiliser Scoop sur votre machine ou bien télécharger directement le fichier et **ajouter le lien du dossier dans les variables d'environnements (PATH)**.

Premier projet Symfony

Activer l'extension zip

Pour éviter le chargement long, vérifiez l'activation de l'extension zip de php.

Modifier le fichier *php.ini* en retirant le point-virgule devant la ligne 'extension=zip'.

Il reste maintenant à créer un nouveau projet, il va s'appeler Wiki Coaster, c'est une application qui permet de gérer des fiches sur les coasters du monde.

Depuis votre dossier, ouvrez une commande et entrez celle-ci :

```
symfony new WikiCoaster --webapp --php=8.4
```

L'option « webapp » va installer des libs pour créer une application web de base.

Installation des dépendances

Les dépendances PHP devraient être installées automatiquement, si ce n'est pas le cas vous pouvez le faire manuellement avec cette commande :

```
composer install
```

Composer va lire le fichier « composer.json » qui se trouve à la racine du projet et installer ces dépendances dans le dossier « vendor ».

GIT

Installez GIT via ce lien : <https://git-scm.com/>

Maintenant que GIT est installé sur la machine, il faut un endroit pour déposer vos projets, créer un compte sur <https://github.com/> si vous n'en avez pas déjà un, c'est ici que vous pourrez me partager vos travaux.
Il est tant de créer le dépôt pour votre projet, depuis GitHub, créez un nouveau dépôt (repository).

Entrez ensuite ces commandes dans le dossier du projet.

```
git remote add origin <url>  
  
git branch -M main  
  
git push -u origin main
```

git init permet de créer un dépôt local, la commande git remote définit le dépôt distant, git add ajoute tous les fichiers pour l'envoi sur le dépôt et git push envoie les fichiers sur le dépôt distant.

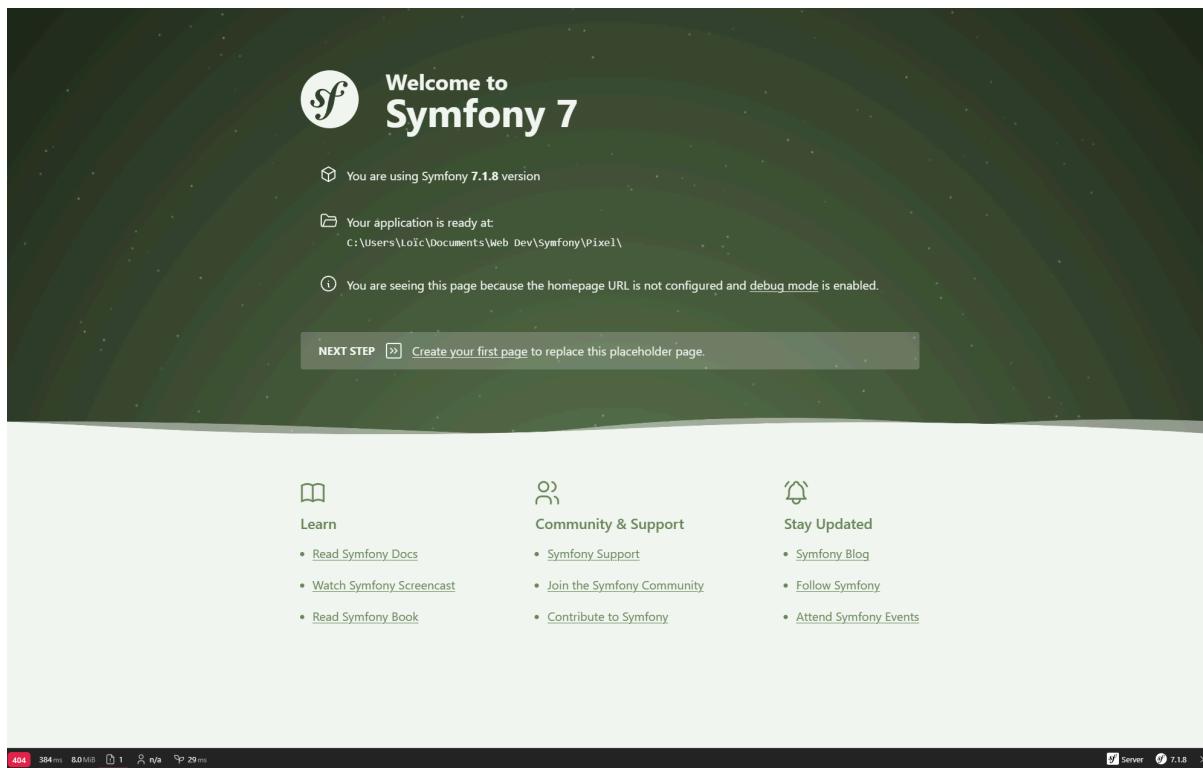
Note: certains dossiers ne sont pas envoyés dans le dépôt distant, c'est le cas du dossier "vendor" car il ne fait pas partie du code de l'application.

Exécuter l'application

Pour voir le fonctionnement de l'application sur navigateur, il est nécessaire de lancer le serveur web en utilisant Symfony CLI:

```
symfony serve
```

Il suffit ensuite d'aller sur le lien indiqué (<http://127.0.0.1:8000>).



Structure d'un projet Symfony

Lors de la création d'un projet, Symfony ajoute une structure composée des dossiers suivants

assets : Il va contenir les fichier css, js et image de l'application, Webpack va ensuite les compacter pour créer des fichiers qui seront mis dans le dossier public

bin : Contient des applications utiles pour le développement comme la console

config : La configuration de l'application se fait dans ce dossier, il contient principalement des fichiers yaml

migrations: c'est dans ce dossier qu'il y aura des fichiers de migrations de base de données, ils permettent de modifier la structure de la base de données lors de mise à jour de l'application

public : Comme dans la plupart des projet il y a le fichier index.php dans ce dossier ainsi que les assets générés par Webpack

src : Le dossier src contient le code source PHP de l'application il contient les contrôleurs, entités, formulaires etc.

templates : Ce dossier est la partie "vue" de l'application, il contient tous les fichiers de template qui seront générés par le moteur Twig

tests : Dans une application importante, il est utile de développer des tests fonctionnels ou unitaires, ils seront écrit dans ce dossier

translations : Symfony possède un outils pour traduire une application, les traductions sont placées dans ce dossier sous forme de fichier yaml

var : Ce dossier contient les fichiers de cache et de log

Dossier "src"

Tous les fichiers PHP dans ce dossier doivent avoir un espace de nom commençant par "App".

Controller : Les contrôleurs ont des méthodes qui sont appelées pour chacune des pages de l'application, ces méthodes retournent une réponse serveur (html, json, pdf, file etc.)

Entity : Les entités représente la base de données sous forme d'objet PHP

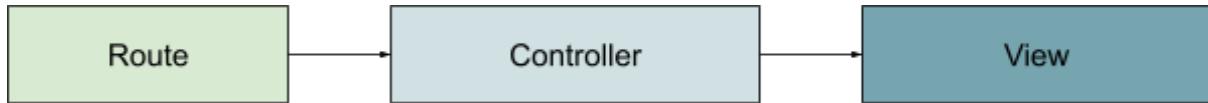
Repository : Les repository sont des classes qui vont permettre de lire dans la base de données, elles contiennent les requêtes.

Form : Les formulaires de l'application sont décrits dans ce dossier (il n'est pas encore créé)

Première page

Pour créer une page simple, il faut suivre 3 étapes:

- Définir une route
- Définir une action dans le controller
- Créer une vue



```

1 <?php
2
3 // src/Controller/AppController.php
4
5 namespace App\Controller;
6
7 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
8 use Symfony\Component\HttpFoundation\Response;
9 use Symfony\Component\Routing\Attribute\Route;
10
11 class AppController extends AbstractController
12 {
13     #[Route('/', name: 'app_home')]
14     public function index(): Response
15     {
16         return $this->render('app/index.html.twig');
17     }
18 }
19

```

```

1 {# templates/app/index.html.twig #-}
2 {% extends "base.html.twig" %}
3
4 {% block body %}
5 <h1>Welcome to Pixel !</h1>
6 {% endblock %}

```

Vous pouvez maintenant admirer votre première page faite en Symfony.



Installation de libs css/js

Tailwind & DaisyUI

Installez Tailwind en suivant les instructions du lien suivant:

<https://tailwindcss.com/docs/installation/framework-guides/symfony>

Sur les machines de l'IUT c'est PHP8.0 qui est défini par défaut, composer affichera une erreur par rapport à ça, il faut ajouter l'option '--ignore-platform-reqs' sur chaque commande.

Installez maintenant la lib DaisyUI: <https://daisyui.com/>

```
npm install daisyui@latest
```

Modifiez le fichier assets/styles/app.css afin d'ajouter la lib DaisyUI:

```
@import "tailwindcss";
@plugin "daisyui";
```

Et enfin, compilez les assets:

```
npm run dev.
```