Programmation avancée & Symfony

BUT Informatique - 3ème année

TP 1

Install. & config.

Claire BOMBEAUX
Version : **06/09/2025**

Objectifs du TP 1

- Installer et configurer un serveur web local (*WampServer*)
- Installer *Composer*
- Installer *Symfony*, le configurer & créer votre projet
- Activer le *Profiler* de *Symfony*
- Activer le *mode DEBUG* de *Symfony*

Sommaire

1. Installation de WampServer et Composer

1.1 WampServer

Intro : Qu'est-ce que WampServer ?

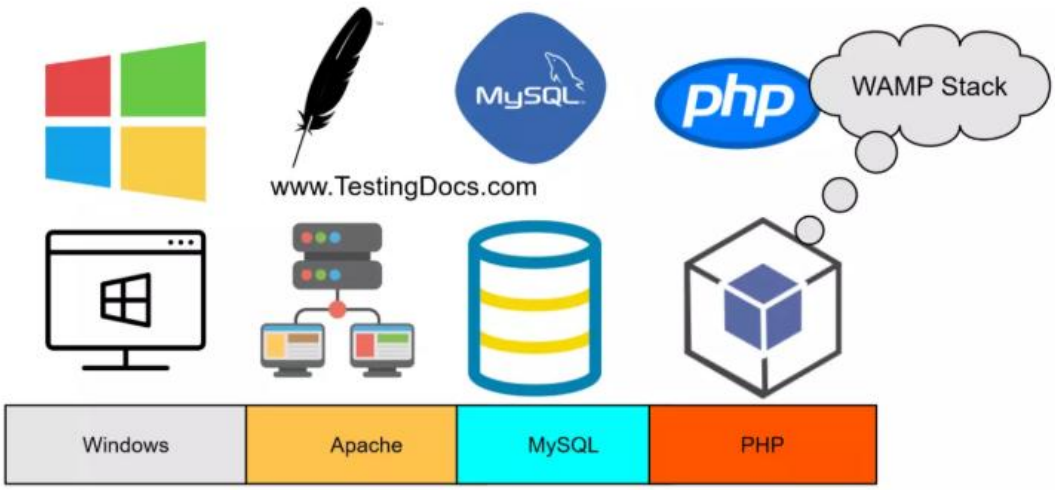
WampServer est un environnement de développement web local pour Windows qui intègre Apache (serveur web), MySQL (base de données), PHP et phpMyAdmin dans un package facile à installer et configurer.

WampServer permet de créer un **serveur web local** sur votre machine **Windows**, vous donnant accès à :

- **Apache** : serveur web (HTTP) pour servir vos pages

Serveur frontal. est « devant » tous les autres et répond directement aux requêtes du client web (navigateur) ;

- **MySQL** : système de gestion de base de données (SGBD). Stockage des données.
- **PHP** : langage de programmation côté serveur. Pages web dynamiques (générées à la demandes. Variables)
- **phpMyAdmin** : interface web pour gérer vos bases de données MySQL

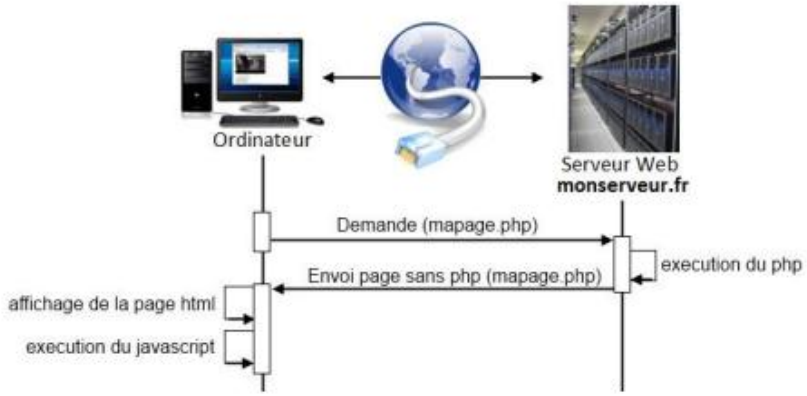


Src. : *https://www.testingdocs.com/testlink-install-requirements/*

Il existe LAMP pour « Linux, Apache, MySQL, PHP ».

Par défaut, les sources de vos projets sont à mettre dans le répertoire `www` de WampServer.

Fonctionnement



Un navigateur seul ne peut pas lire le PHP, seul le serveur le peut. Il faudra donc déposer le site web dans un dossier spécial du serveur, le **répertoire www** ("www directory") accessible via Wamp.

Une fois le site web dans ce dossier, vous pourrez alors visualiser votre œuvre en vous connectant au "localhost". Celui-ci est accessible via le navigateur **en tapant** "localhost" dans la barre de navigation, mais également via WampServer **en cliquant** sur "localhost".

Src. : *https://webgburnet.com/wp-content/uploads/2021/05/8-Wampserver.pdf*

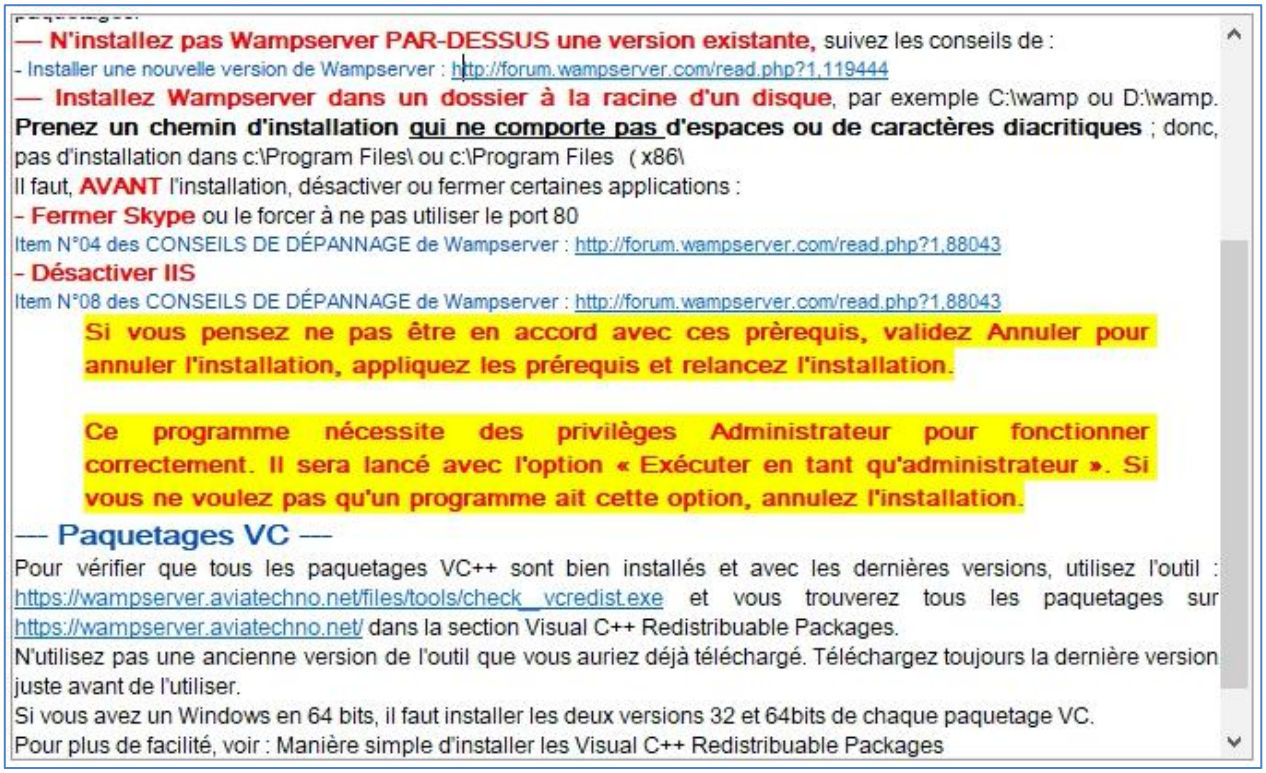
Installer WampServer

Téléchargez WampServer depuis le site officiel : https://www.wampserver.com

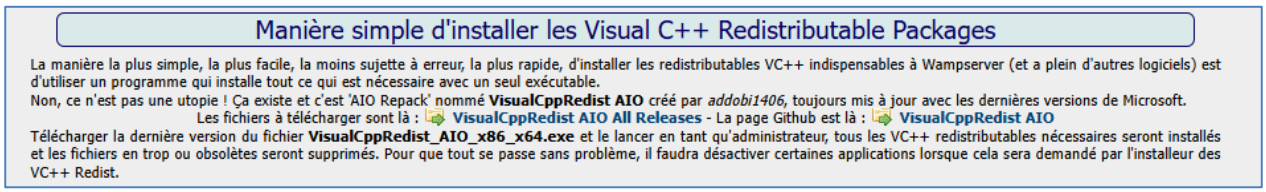
Utiliser « passer au téléchargement direct. » pour ne pas avoir à vous inscrire

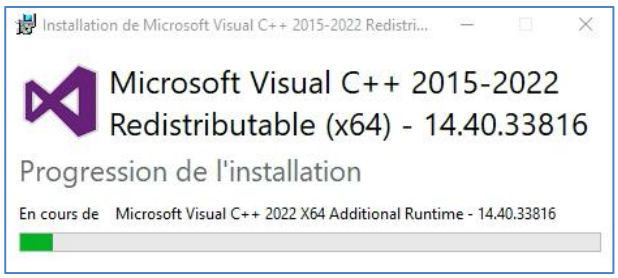
Choisissez la version correspondant à votre système Windows (ex. : 64-bit) : ***wampserver3.3.7_x64.exe*** …

Attention Lire **instructions utilisation.pdf** + **tenir compte des remarques et prérequis affichées lors de l'installation !**

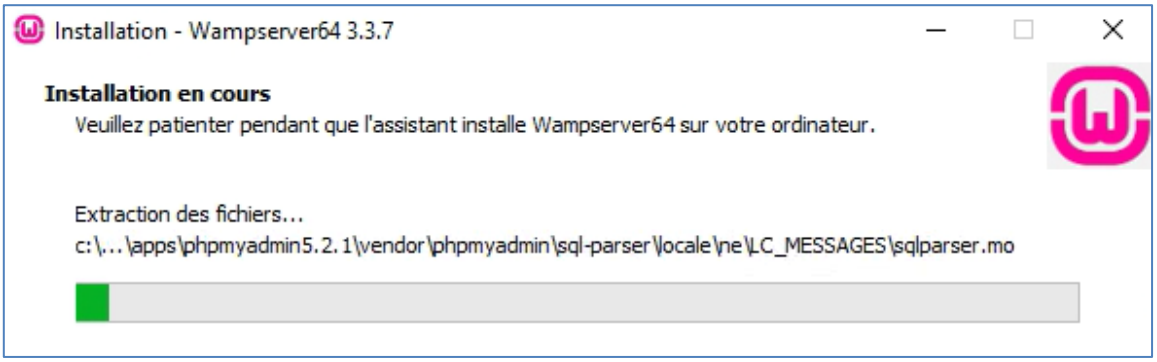


Cela inclut d'installer les packages VC++ nécessaires : https://wampserver.aviatechno.net **avant** l'install. de WAMP.

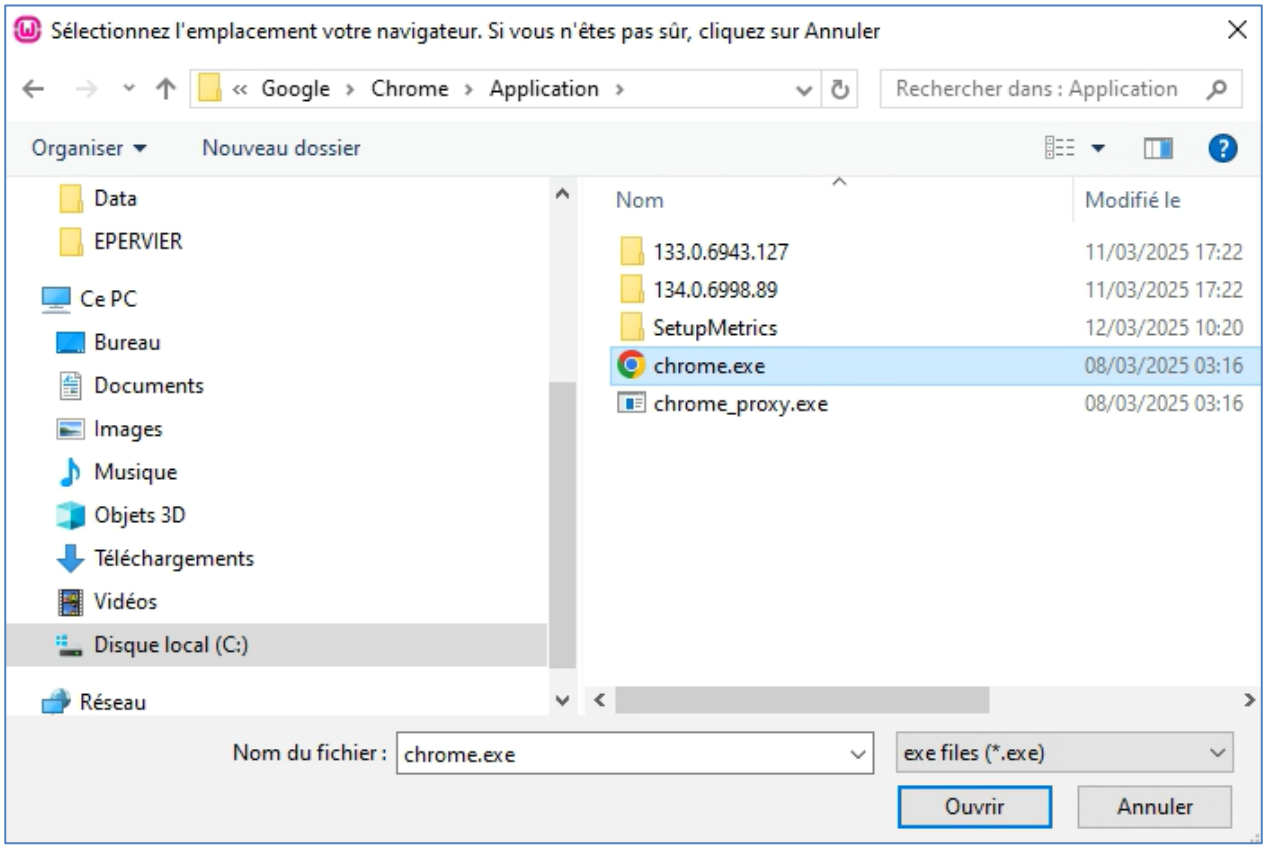




...

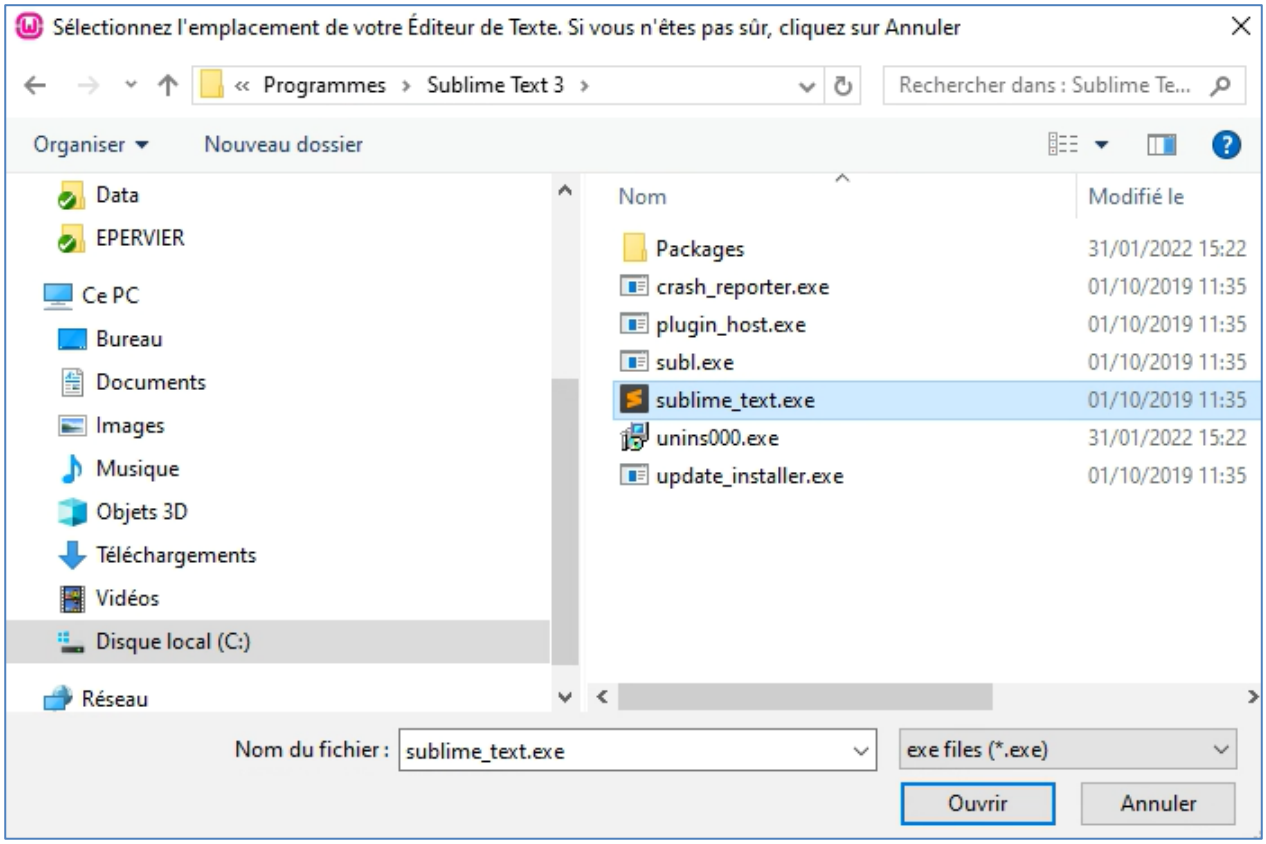


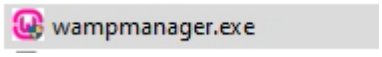
Choisir votre navigateur par défaut :



Choisir votre éditeur de texte par défaut (***Sublime Text*** conseillé)

Téléchargement Sublime Text : https://www.sublimetext.com/download



Démarrez WampServer 

L'icône dans la barre des tâches doit être verte : 

☑ Rappel : Adresse localhost → 127.0.0.1

1.2 Composer

Composer est le gestionnaire de dépendances PHP indispensable pour Symfony.

En effet, il permet d'installer des centaines de packages librement disponibles. On les trouve sur Packagist . Composer permet de gérer les dépendances d'un projet et également de créer le squelette d'une application Symfony.

📖 **Documentation :** https://getcomposer.org/doc/

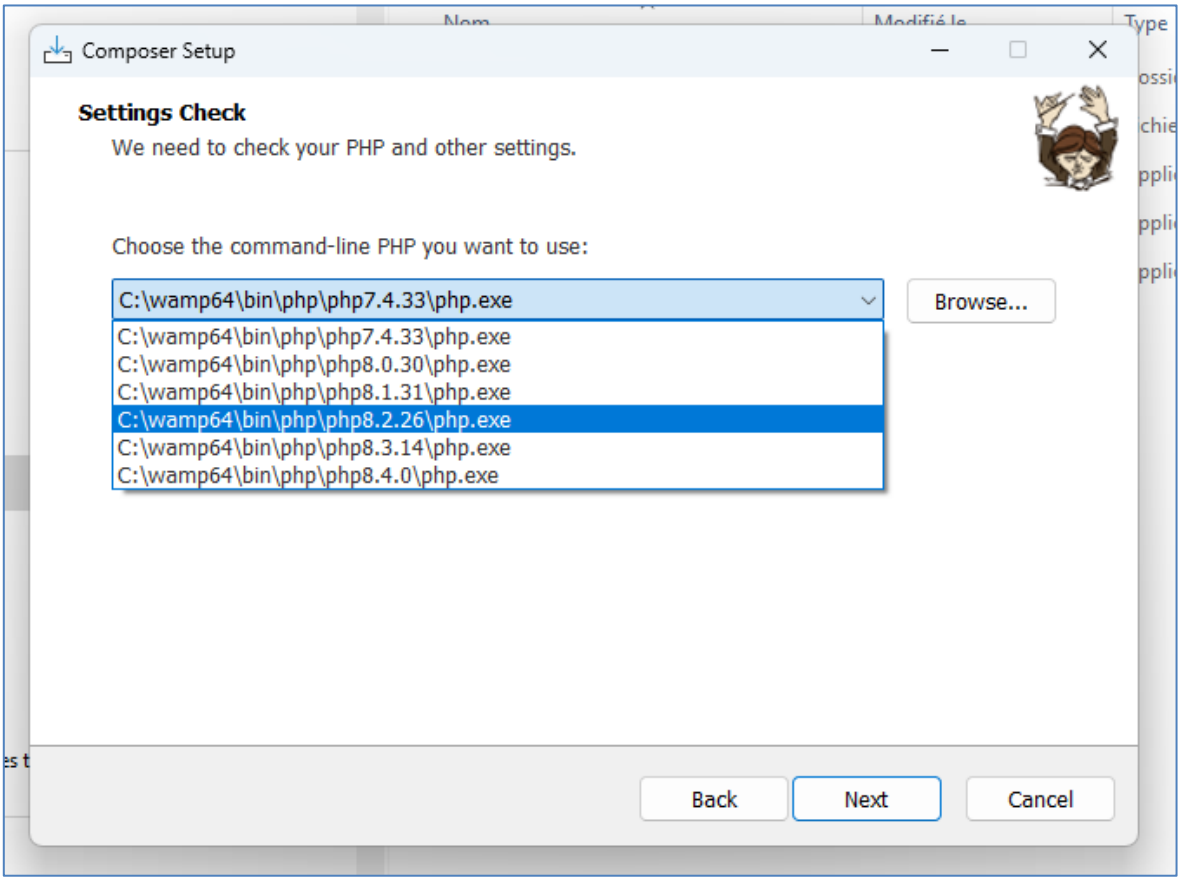
Installer Composer

Plusieurs méthodes sont possibles, en voici quelques-unes :

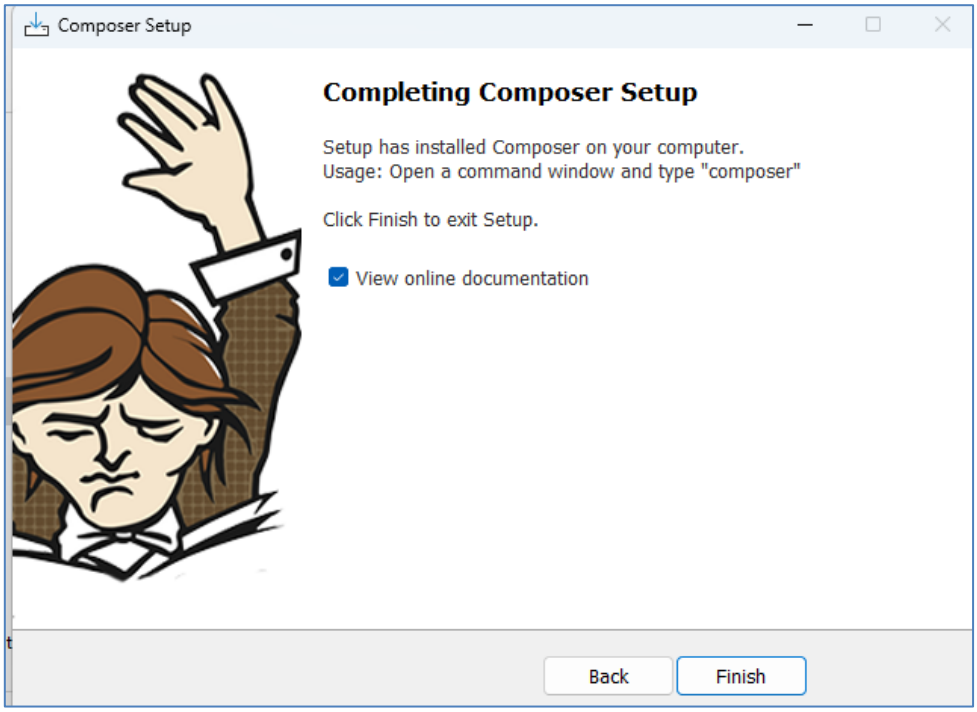
Méthode 1 : Installateur Windows (recommandé)

Téléchargez `Composer-Setup.exe` depuis le site officiel : getcomposer.org

Exécutez l'installateur et configurer Composer



Choisir la version de PHP utilisée (Pour cela voir la *Partie 2. Création d'un projet Symfony > 2.1 Releases et Requierements* de ce TP1).



Attention si Internet KO (type VM) : tenter de saisir votre proxy. Si aucun accès Internet, voir méthode manuelle

Méthode 2 : Installation en ligne de cmde

Command-line installation

To quickly install Composer in the current directory, run the following script in your terminal. To automate the installation, use the guide on installing Composer programmatically.

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') === 'dac665fdc30fdd8ec78b38b9800061b4150413ff2e3b6f88543c
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

This installer script will simply check some `php.ini` settings, warn you if they are set incorrectly, and then download the latest `composer.phar` in the current directory. The 4 lines above will, in order:

- Download the installer to the current directory
- Verify the installer SHA-384, which you can also cross-check here
- Run the installer
- Remove the installer

Most likely, you want to put the `composer.phar` into a directory on your PATH, so you can simply call `composer` from any directory (*Global install*), using for example:

```
sudo mv composer.phar /usr/local/bin/composer
```

For details, see the instructions on how to install Composer globally.

WARNING: Please do not redistribute the install code. It will change with every version of the installer. Instead, please link to this page or check how to install Composer programmatically.

Src. : *https://getcomposer.org/download/*

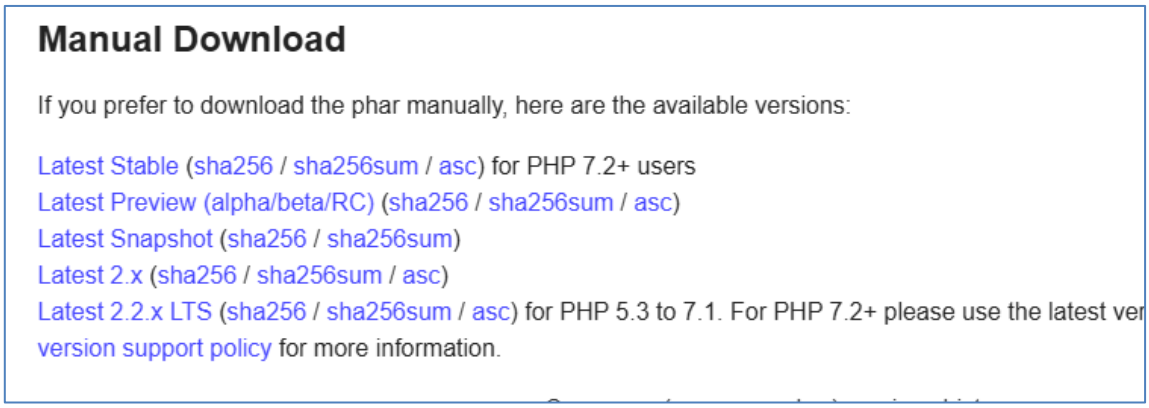
Détails des commandes :

- Chargement de l'installateur
- Vérification de l'intégrité (hash)
- Exécution de l'installateur
- Retire l'installateur

Méthode 3 : Installation hors ligne (pour information uniquement)

Sur votre machine hôte (avec internet) :

Téléchargez le fichier `composer.phar` depuis getcomposer.org



Téléchargez aussi `composer-setup.php` si nécessaire

Transférez ces fichiers vers votre VM (via Dossier partagé entre hôte et VM

ou Clé USB ou Réseau local si disponible)

Dans votre VM (sans Internet) :

Placez `composer.phar` dans le dossier PHP de WampServer :

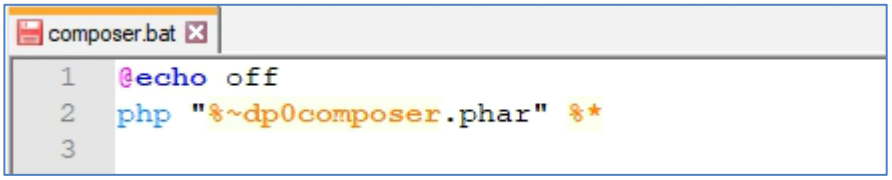
cmd

```
C:\wamp64\bin\php\php8.2.26\composer.phar
```

Créez un fichier batch `composer.bat` dans le même dossier :

batch

```
@echo off
php "%~dp0composer.phar" %*
```

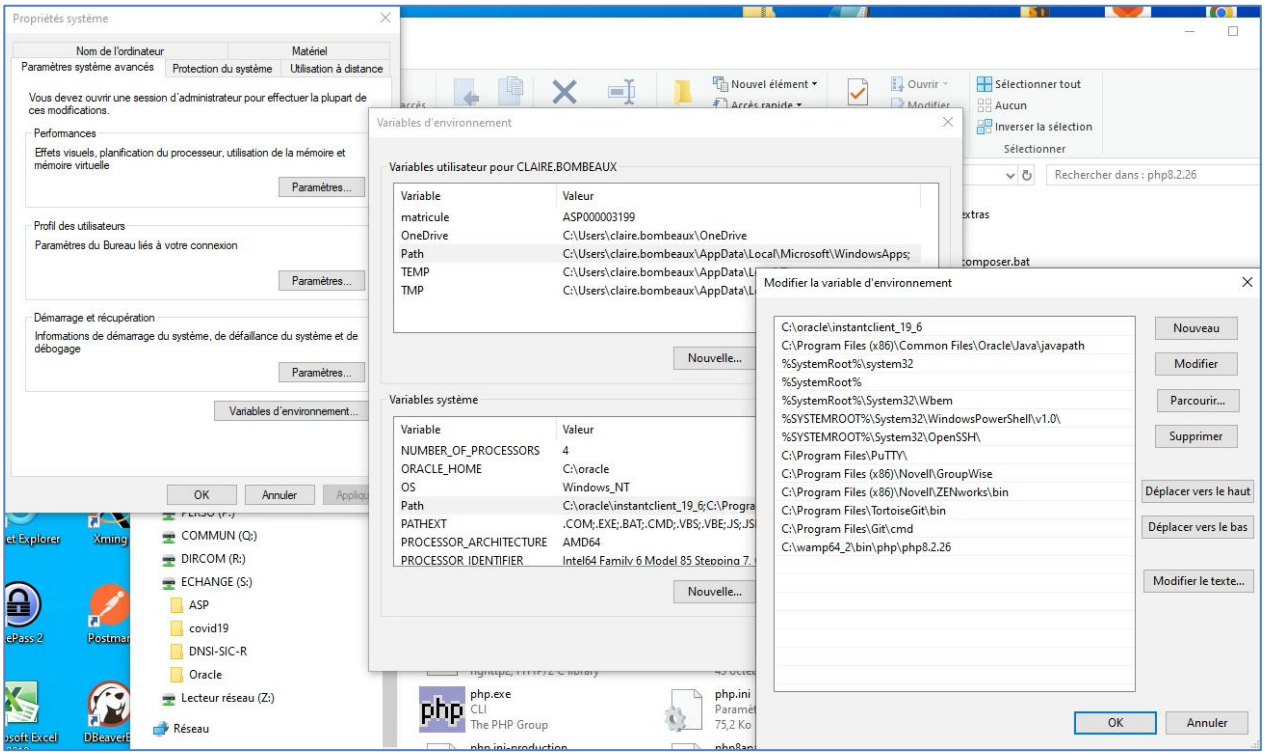


Ajoutez ce dossier au PATH Windows de votre VM (Cf. partie suivante)

Ajouter PHP au PATH

Requis si non fait via l'install. de Composer

→ Variables d'env. système



Ajouter le chemin de PHP (Ex. : `C:\wamp64\bin\php\php8.2.26`) au ***PATH***

Redémarrer car les variables système se chargent au démarrage

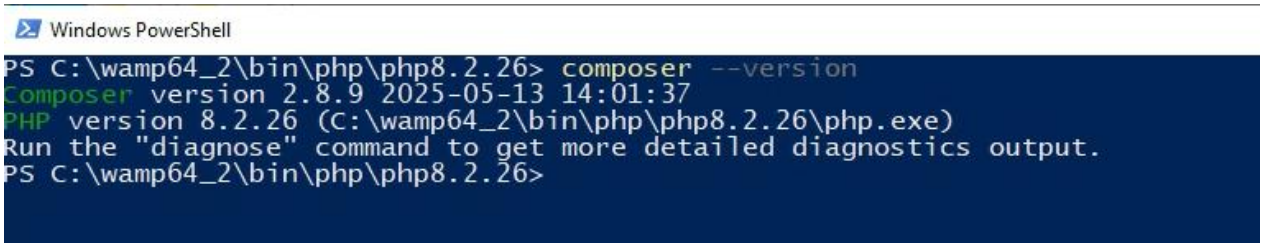
cmd

```
php -v
```

Vérifier l'installation de Composer

cmd

```
composer --version
```



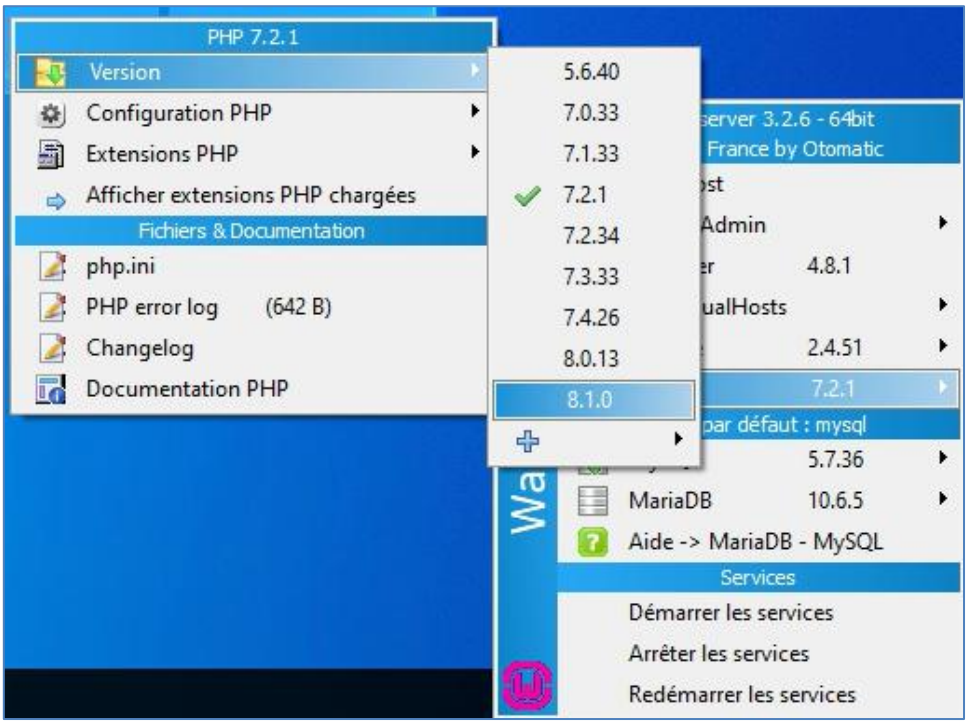
1.3 Configuration de PHP (via WampServer)

Après avoir installé WampServer, vous devez vous assurer que PHP est configuré correctement pour Symfony.

Pour cela, consultez les ***requierements*** de Sf : *Cf. Partie 2. Création d'un projet Symfony > 2.1 Releases et Requierements de ce cours.*

Vérifiez que PHP est en version 8.2 ou supérieure (requis pour Symfony 6+)

Allez dans **PHP** → **Version** et sélectionnez PHP 8.2 ou supérieur



Voir aussi partie précédente de ce TP : *Ajouter PHP au PATH* (*cmd* : `php -v`)

Vérifier / activez les extensions PHP suivantes, nécessaires à SF , dans WampServer :

- ✓ `php_curl`
- ✓ `php_intl`
- ✓ `php_mbstring`
- ✓ `php_openssl`
- ✓ `php_pdo_mysql`
- ✓ `php_xml`

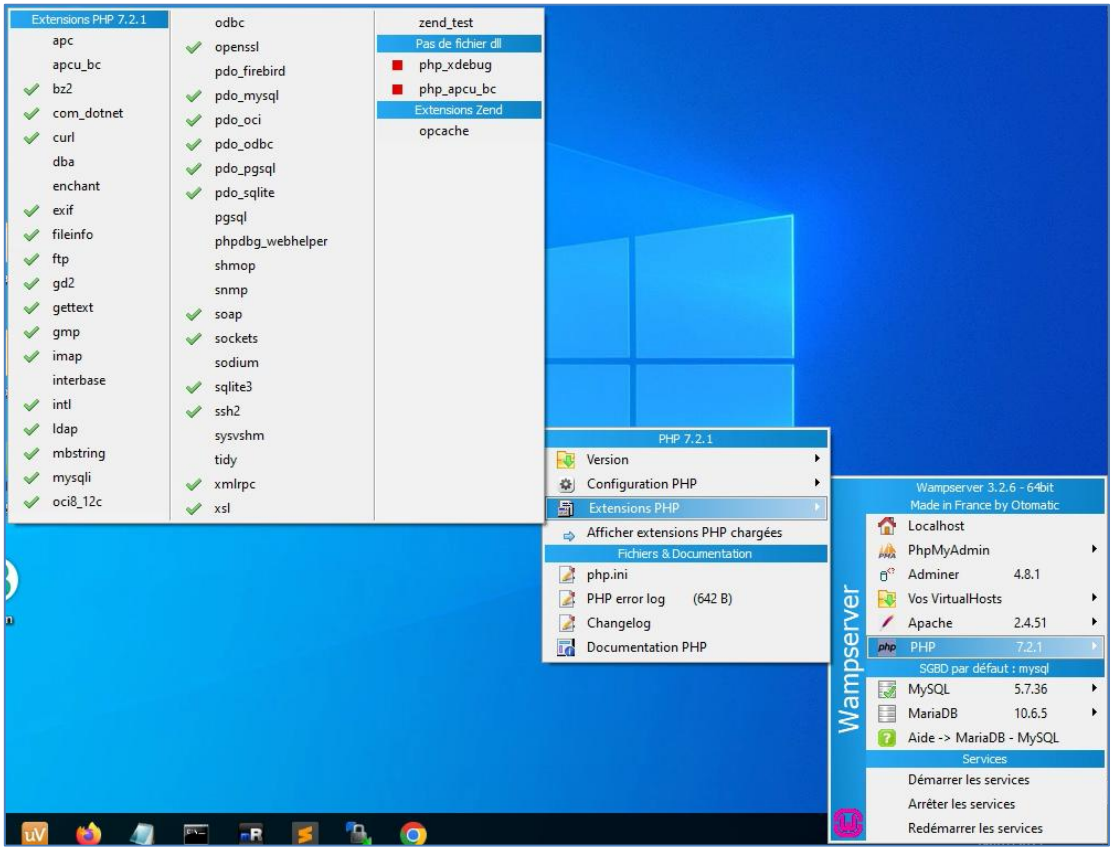
Redémarrer WampServer.

*Une extension PHP est un **module complémentaire** qui ajoute des fonctionnalités supplémentaires au langage PHP de base. C'est comme un plugin qui étend les capacités du langage.*

Activation d'une extension => fichier **php.ini**

A noter : il existe une fonctionnalité de WAMP permet de voir quelles extensions sont activées.

Clic gauche sur l'icône WampServer → **PHP** → **Extensions PHP**



Familiarisiez-vous avec WAMPServer et ses fonctionnalités. Comprenez comment accéder aux fichiers de configurations et aux fichiers de logs.

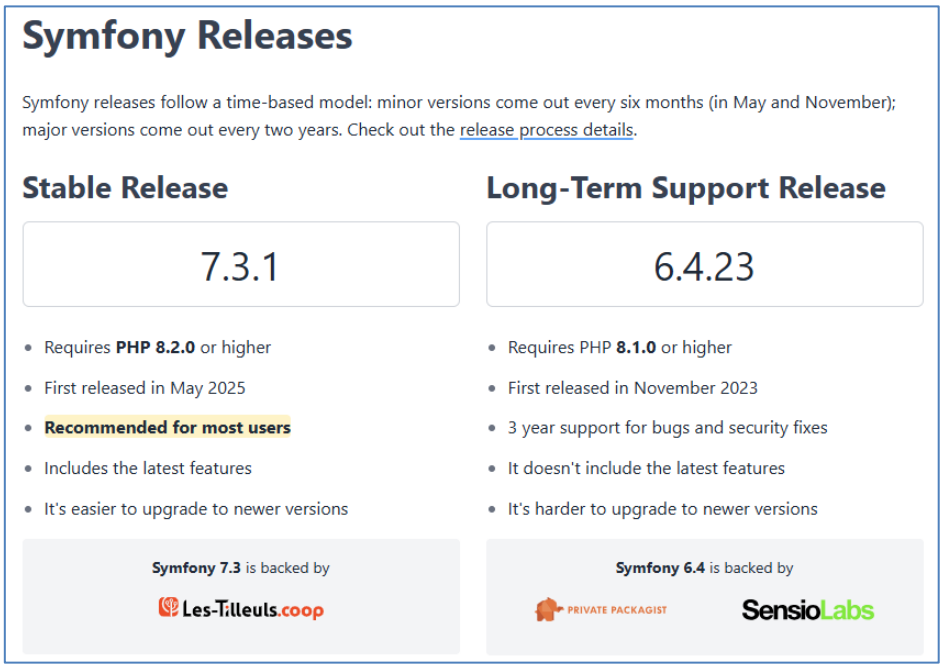
2. Création et config. d'un projet Symfony

2.1 Intro. : Versions et prérequis

Avant toutes installations, il est généralement nécessaire d'identifier la version souhaitée et d'en connaitre les prérequis logiciels et physiques.

Versions (*Releases*) de Symfony

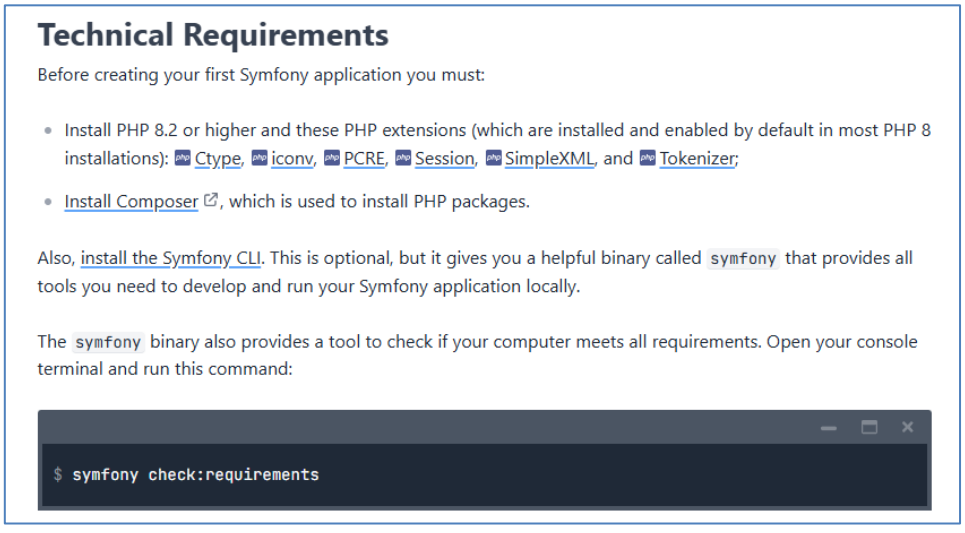
https://symfony.com/releases



LTS : version stable d'un logiciel informatique, maintenue pendant une période plus longue que l'édition standard.

Prérequis techniques (*requierements*)

https://symfony.com/doc/current/setup.html#technical-requirements



2.2 Création du projet Symfony

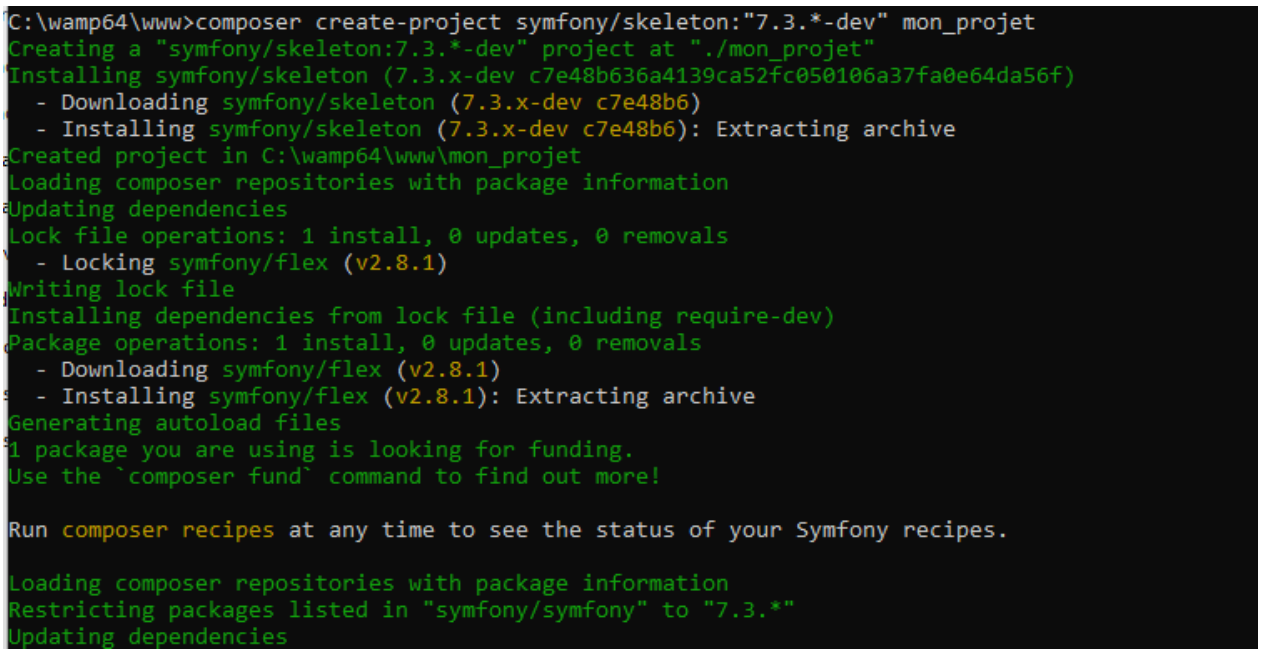
Ouvrez l'invite de commande Windows

Placez-vous dans le répertoire `www` de WampServer et créez votre projet :

cmd

```
cd C:\wamp64\www
```

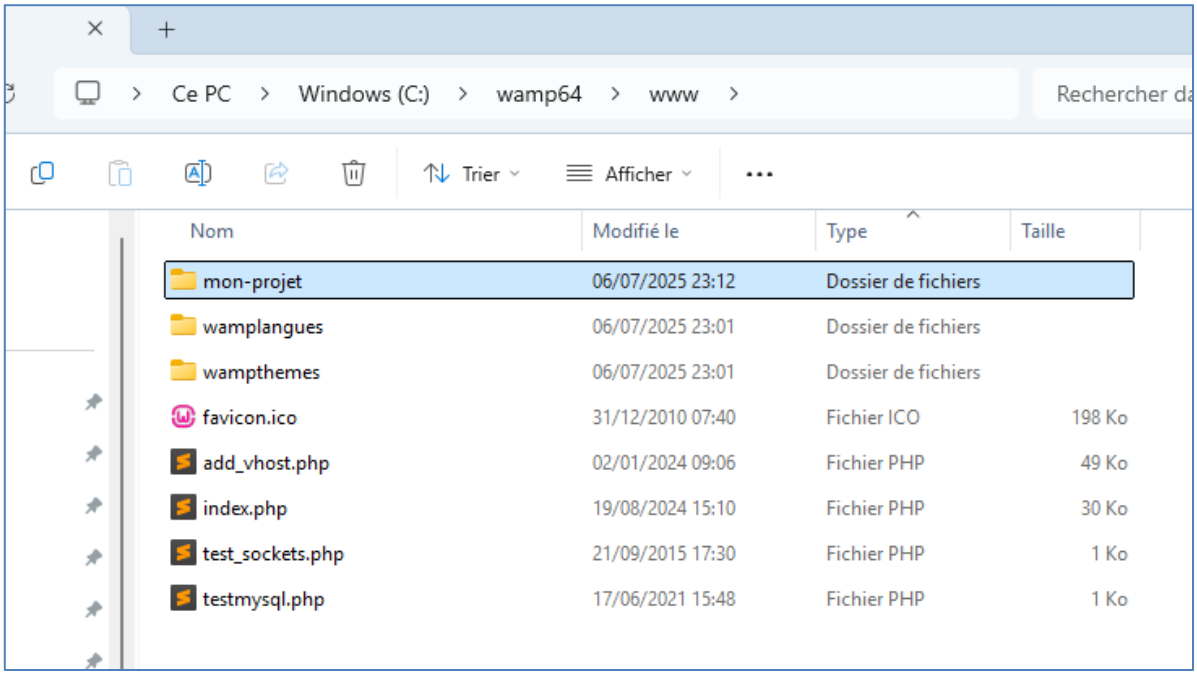
`composer create-project symfony/skeleton:7.3.*-dev mon_projet`



Note si proxy, le setter avant

```
set HTTP_PROXY=http://username:password@proxy.company.com:8080
set HTTPS_PROXY=http://username:password@proxy.company.com:8080
```

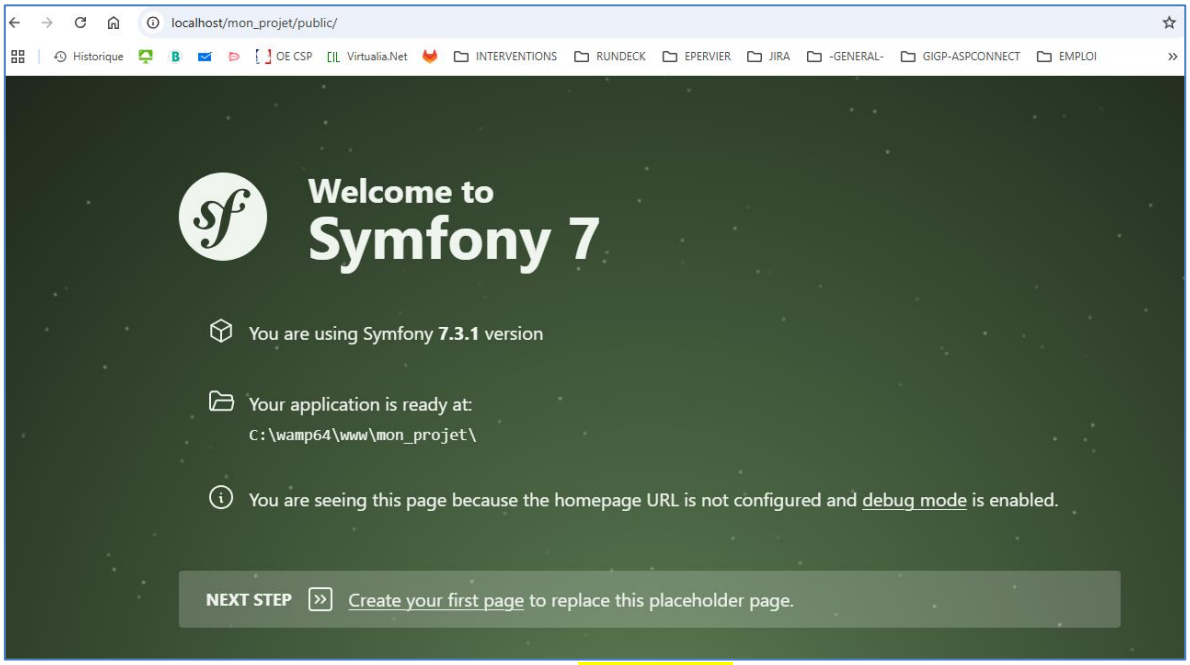
```
cd mon_projet
```



📖**Documentation** : https://symfony.com/doc

Accédez directement à `http://localhost/mon_projet/public/` dans votre navigateur.

Cette page qui s'affiche est la page d'accueil de Symfony pour toutes les applications qui viennent d'être créées.



http://127.0.0.1/mon_projet/public/

2.3 Activer le Profiler de Symfony

Le **Profiler** de Symfony est un outil de débogage et d'analyse des performances qui collecte des informations détaillées sur l'exécution de votre application.

C'est un système qui capture et stocke des données sur chaque requête HTTP traitée par votre application Symfony. Il fournit une interface web pour examiner ces données de manière détaillée.

Le profiler permet d'utiliser la Web Debug **Toolbar**, une barre d'outils qui apparaît en bas de chaque page en mode développement :

Installation et configuration du Profiler

Le Profiler est généralement installé automatiquement dans l'environnement de développement.

Si ce n'est pas le cas, installez-le :

`cd C:\wamp64\www\mon_projet` → *se placer dans son répertoire projet*

```
composer require --dev symfony/profiler-pack
```

```
PS C:\wamp64\www\mon_projet> composer require --dev symfony/profiler-pack
./composer.json has been updated
Running composer update symfony/profiler-pack
Loading composer repositories with package information
Restricting packages listed in "symfony/symfony" to "7.3.*"
Updating dependencies
Lock file operations: 6 installs, 0 updates, 0 removals
  - Locking symfony/profiler-pack (v1.0.6)
  - Locking symfony/translation-contracts (v3.6.0)
  - Locking symfony/twig-bridge (v7.3.0)
  - Locking symfony/twig-bundle (v7.3.1)
  - Locking symfony/web-profiler-bundle (v7.3.1)
  - Locking twig/twig (v3.21.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 6 installs, 0 updates, 0 removals
  - Downloading twig/twig (v3.21.1)
  - Downloading symfony/translation-contracts (v3.6.0)
```

Assurez-vous que le Profiler est activé dans votre fichier `config/packages/[dev]/web_profiler.yaml` :

yaml

```yaml
web_profiler:
    toolbar: true
    intercept_redirects: false
```

Le profiler se configure dans le fichier `config/packages/dev/web_profiler.yaml`. Vous pouvez

personnaliser quels collecteurs de données activer, définir des conditions d'activation, ou modifier l'apparence de la Toolbar.
Dans `config/packages/dev/framework.yaml`, vérifiez que le profiler est activé :

yaml

```yaml
framework:
    profiler:
        only_exceptions: false
```

Assurez-vous que vous êtes bien en environnement de développement, dans le fichier `.env`

```
APP_ENV=dev
```

Et/ou commande **`php bin/console --version`**

Cf. partie Environnements Symfony de ce TP

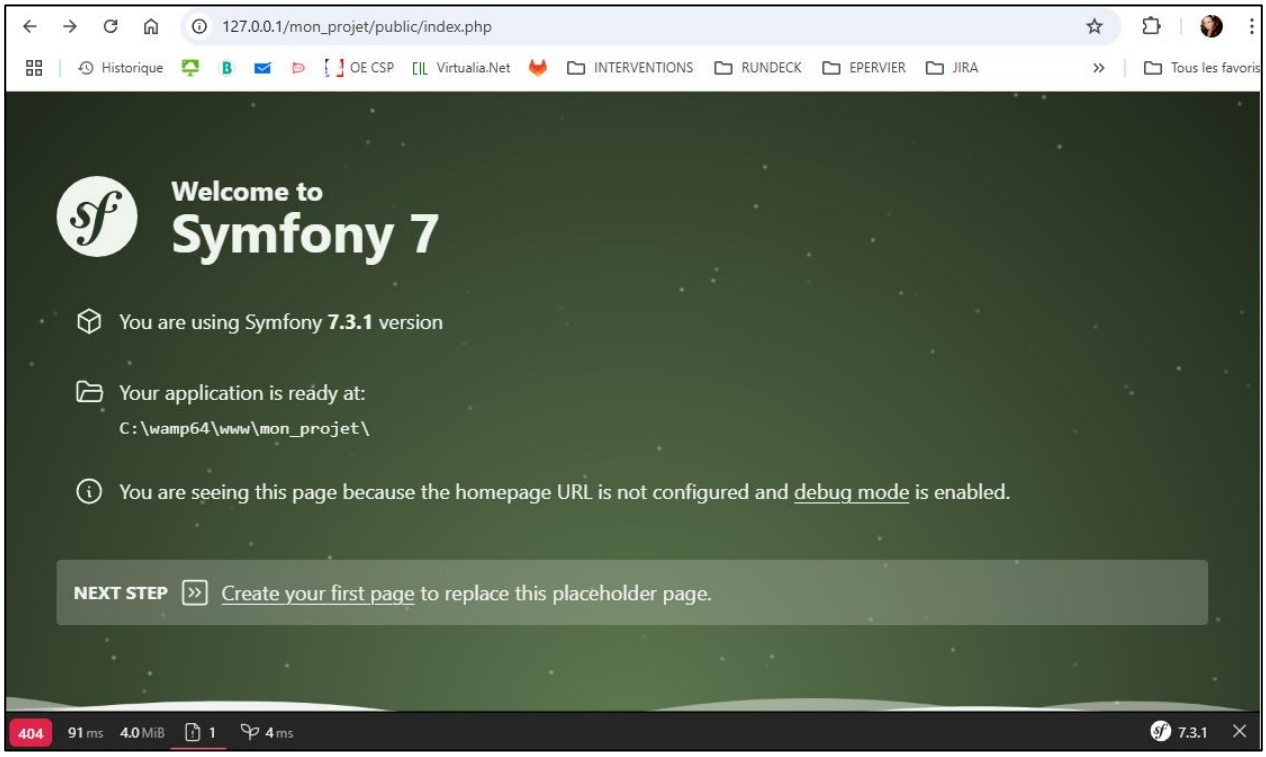
Vider le cache (si nécessaire) :

```
PS C:\wamp64\www\mon_projet> php bin/console cache:clear --env=dev
```

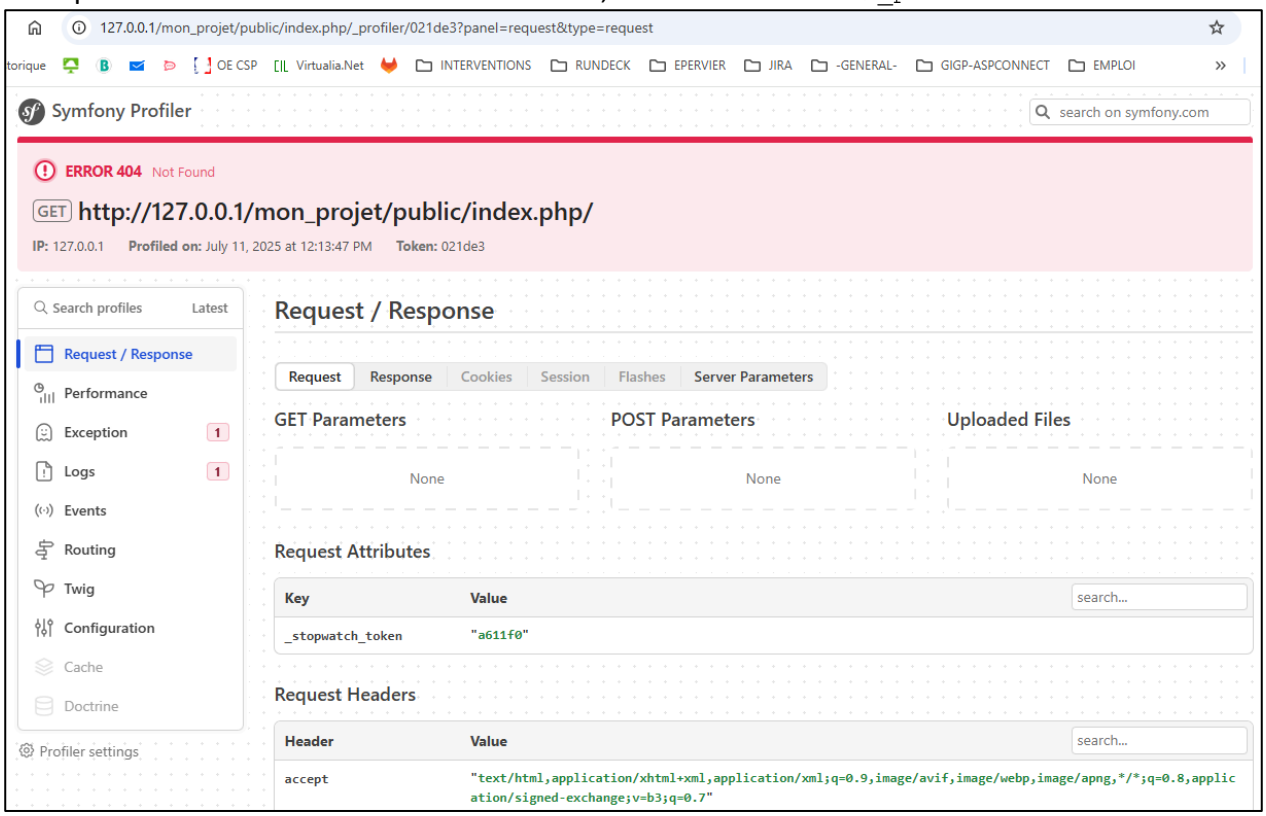
Vérifiez que le bundle est bien enregistré dans `config/bundles.php` :

```
Symfony\Bundle\WebProfilerBundle\WebProfilerBundle::class => ['dev' => true, 'test' => true],
```

La toolbar devrait maintenant apparaître en bas de vos pages web lorsque vous naviguez sur votre application en mode développement.



Vous pouvez accéder au Profiler via la Toolbar, ou bien via l'URL …`/_profiler`.



Familiarisiez-vous avec l'utilisation de la Toolbar et le Profiler.

2.4 Configuration

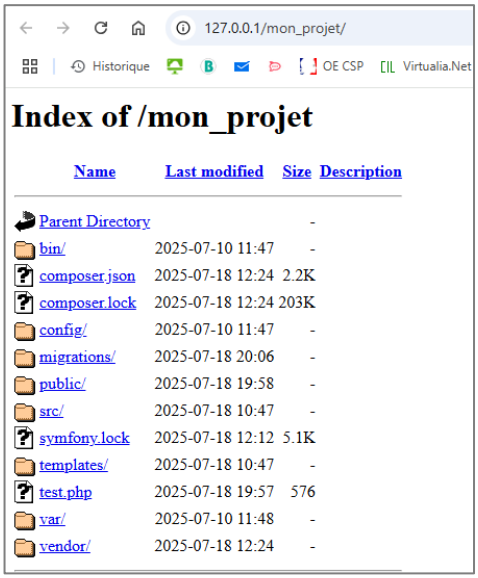
Alias, Virtuals Hosts & Fichiers .htaccess Apache

Actuellement seule l' url http://127.0.0.1/mon_projet/public/index.php fonctionne (et http://127.0.0.1/mon_projet/public/ mais identique).

```
C:\wamp64\www\
└── mon_projet_symfony\
    ├── bin/
    ├── config/
    ├── public/           ← Point d'entrée web ACTUEL
    │   ├── index.php
    │   └── .htaccess
    ├── src/
    ├── templates/
    ├── var/
    └── vendor/
```

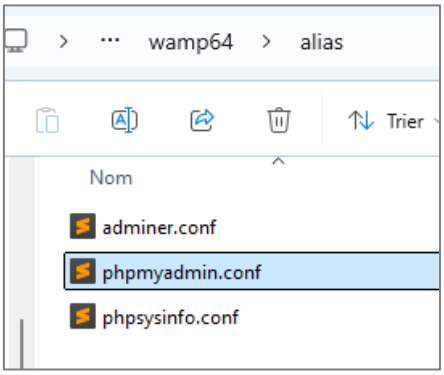
Or nous aimerions avoir la route suivante comme point d'entrée de l'application : **http://127.0.0.1/mon_projet/**

Pour l'instant cette url donne :



Méthode 1 - Créer un alias (recommandé si suffisant)

Mettre en place un alias devrait être suffisant. Les alias sont définis dans le répertoire éponyme de WAMP :



📖 **Documentation** : https://httpd.apache.org/docs/2.0/mod/mod_alias.html#alias

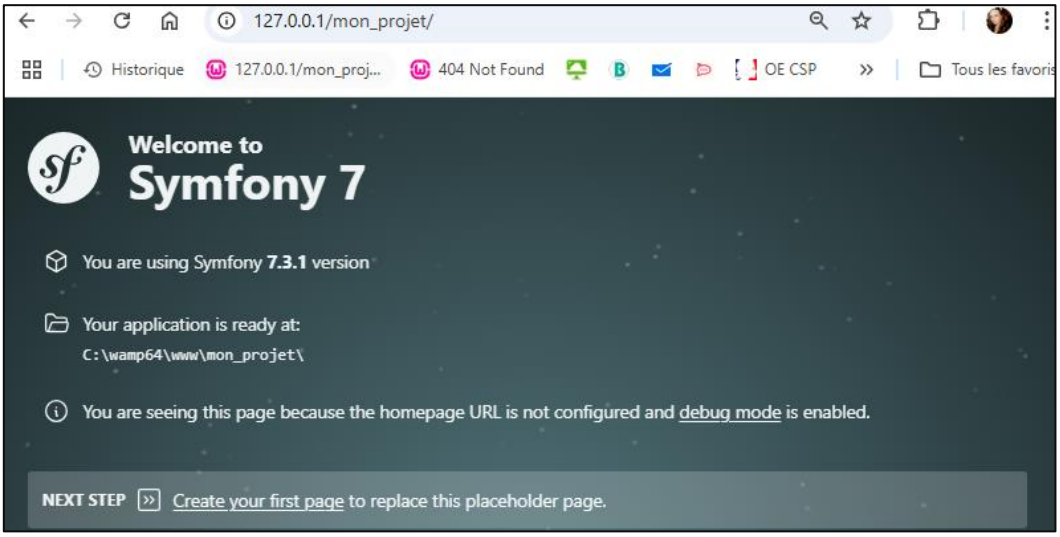
Créer votre fichier :

monprojet.conf

```
Alias /mon_projet "${INSTALL_DIR}/www/mon_projet/public"

<Directory "${INSTALL_DIR}/www/mon_projet/public/">
    Options +Indexes +FollowSymLinks
    AllowOverride all
    Require local
</Directory>
```

Puis redémarrer les services WAMP (via Clic gauche sur l'icône WAMP > Redémarrer les services) pour qu'il soit pris en compte.



http://127.0.0.1/mon_projet/

Méthode 2 – Définir un Virtual Host (pour information)

☞ Renseignez-vous sur les Virtual Hosts et comment en créer un.

Renseignez-vous sur les Virtual Hosts et comment en créer un.
Virtual Hosts =>

```
httpd-vhosts.conf
1  # Virtual Hosts
2  #
3  <VirtualHost *:80>
4    ServerName localhost
5    ServerAlias localhost
6    DocumentRoot "${INSTALL_DIR}/www"
7    <Directory "${INSTALL_DIR}/www/">
8      Options +Indexes +Includes +FollowSymLinks +MultiViews
9      AllowOverride All
10     Require local
11   </Directory>
12 </VirtualHost>
13 #
14 <VirtualHost *:80>
15   ServerName monprojet
16   ServerAlias monprojet
17   DocumentRoot "${INSTALL_DIR}/www/mon_projet/public"
18   <Directory  "${INSTALL_DIR}/www/mon_projet/public/">
19     Options +Indexes +Includes +FollowSymLinks +MultiViews
20     AllowOverride All
21     Require local
22   </Directory>
23 </VirtualHost>
24 #
```

- Activer le module **rewrite** d'Apache si non fait :
  Clic gauche sur l'icône **WAMP** (de votre barre des tâches) > **Apache** > **Modules Apache** > Cochez **rewrite_module** > Redémarrez les services WAMP

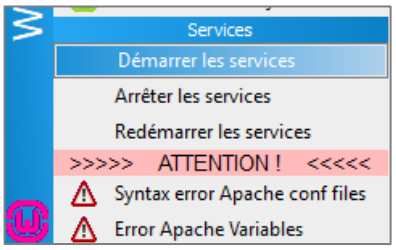
- Créer un VirtualHost :
  Clic gauche sur l'icône WAMP > Vos **Virtual Host** > **Gestion des Virtuals Hosts (**http://localhost/add_vhost.php?lang=french**)**
  Ou manuellement : éditer le fichier
  *C:\wamp64\bin\apache\apache2.4.62.1\conf\extra**httpd-vhosts.conf***

Exemples :

```
<VirtualHost *:80>
    ServerName mon-projet.local
    DocumentRoot "C:/wamp64/www/mon_projet/public"
    <Directory "C:/wamp64/www/mon_projet/public">
        AllowOverride All
        Require local
    </Directory>
</VirtualHost>
```

Ensuite il convient de redémarrer les services WAMP (via Clic gauche sur l'icône WAMP > Redémarrer les services) (Ou, dans notre cas, Clic droit sur l'icône WAMP > Tous > Outils > Redémarrage DNS devrait suffire).

En cas d'erreur de syntaxe, WampServer peut vous l'indiquer :



Méthode 3 : via Fichier .htaccess

☞ Renseignez-vous sur les Fichier .htaccess, leurs intérêts et leur syntaxe.

🕮 **Documentation** : https://httpd.apache.org/docs/2.4/fr/howto/htaccess.html

Fichier .htaccess =>

Utilisation et intérêts =>

Créez un fichier `.htaccess` dans le dossier racine `C:\wamp64\www\mon_projet\` avec un contenu similaire à :

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ public/$1 [L]
```

Cela devrait vous permettre d'accéder au projet via `http://127.0.0.1/mon_projet/`

Environnements Symfony

Vous n'avez qu'une seule application, mais il est probable que vous ayez besoin qu'elle se comporte différemment à différents moments. Par exemple durant la phase de développement, vous aurez besoin de tout logguer et d'avoir de bons outils de débogage. Et arès le déploiement en env. de production, vous voudriez que cette même application soit optimisée pour la vitesse et enregistre uniquement les erreurs.

Symfony permet cela en définissant proposant plusieurs environnements, qui sont, par défaut :

- **`dev`** pour le développement local,
- **`prod`** pour la production (application « finale »),
- **`test`** for tests automatisés

  Remarque : il possible de créer soit même d'autres environnements.

Vous pouvez définir des configurations spécifiques à chacun de ces environnements.

Cependant à chaque instant, un seul environnement est utilisé, l'environnement actif, qui est définit dans le fichier .env.

```
# .env (or .env.local)
      APP_ENV=dev
```

Laissez l'env. de dev. activé pour l'instant et tant qu'on sera en phase de développement (logique…)

Activer le mode DEBUG

Méthode 1 - Via la variable d'environnement (**recommandé**)

```
# .env (or .env.local)
# set it to 1 to enable the debug mode

APP_DEBUG=1
```

OU Méthode 2 - Via le fichier de configuration

Yaml

```yaml
# config/packages/framework.yaml

framework:
    debug: true
```

Le mode DEBUG de Symfony offre des informations détaillées sur les erreurs avec des stack traces complètes, facilitant grandement le diagnostic des problèmes.

⚑ **Attention** : En production, le mode DEBUG impérativement être désactivé pour des raisons de sécurité et de performance.

Conclusion

Les avantages de l'utilisation de WAMP sont nombreux :

- Installation simple avec interface graphique
- Gestion centralisée des services web
- phpMyAdmin intégré pour notre BDD MySQL

Cette configuration vous donne un environnement de développement Symfony complet et fonctionnel sur Windows, parfait pour commencer vos projets web.

Mais il existe des alternatives, telles que :

- **Docker** avec des conteneurs pour un environnement plus portable

- **Symfony CLI** qui inclut un serveur de développement optimisé

  📖**Documentation** : https://symfony.com/doc/current/setup/symfony_cli.html

  Télécharger le Symfony CLI sur https://symfony.com/download pour installer un serveur de développement web et utiliser la commande `php bin/console server:run` ***pour le démarrer.***

- **XAMPP**, alternative à WampServer

  📖**Documentation** : https://fr.wikipedia.org/wiki/XAMPP

[TP 1] Annexe 1 : Commandes utiles

Installer/créer un projet Symfony

via Composer

```
composer create-project symfony/skeleton mon_projet
```

```
composer create-project symfony/skeleton:7.3.*-dev mon_projet
```

Autre possibilités (sans Composer) :

```
# run this if you are building a traditional web application
$ symfony new my_project_directory --version="7.3.x" --webapp

# run this if you are building a microservice, console application or API
$ symfony new my_project_directory --version="7.3.x"
```

La commande « symfony new --webapp [nom du projet] » permet de créer un nouveau projet Symfony incluant tous les composants généralement nécessaires à une application Web standard, alors que la commande « symfony new [nom du projet] » ne créerait qu'un projet minimaliste.

Voir les commandes disponibles

```
php bin/console
```

Connaître sa version de Symfony

```
php bin/console --version
```

Mettre à jour composer, les dépendances

```
composer update
```

Vider le cache

```
php bin/console cache:clear --env=dev
```

Debug config

```
php bin/console debug:config
```

Obtenir de l'aide sur une commande

```
<commande> --help
```

Par exemple : **`php bin/console debug:config --help`**