Installation

Présentation du framework Symfony

Symfony est un framework PHP open-source qui permet de développer des applications web robustes et maintenables. Il repose sur l'architecture MVC (Modèle-Vue-Contrôleur) et propose de nombreux composants réutilisables.

Caractéristiques principales :

- Respect des bonnes pratiques et des standards PHP.
- Utilisation de composants modulaires et réutilisables.
- Gestion avancée des routes et du cache.
- Intégration avec Doctrine ORM pour la gestion de base de données.
- Système de templating performant avec Twig.
- Sécurité avancée avec authentification et autorisation.

1. Langage PHP

Plutôt que d'installer PHP et ses extensions de manière isolée, il est recommandé d'installer sur son ordinateur une suite logicielle « **tout en un** » telle que **WAMP** ou **MAMP**.

Cela permet de bénéficier d'un package d'outils comme par exemple le système de gestion de bases de données **MySQL** et l'interface graphique **phpMyAdmin**.

L'installation de cette suite logicielle est détaillée dans la formation dédiée au langage PHP.

2. Gestionnaire de dépendances (Composer)

C'est quoi ?

Composer est un outil dédié à la gestion des **dépendances** PHP d'un projet. Les dépendances sont toutes les bibliothèques dont votre projet dépend pour fonctionner.

Pour rappel, une bibliothèque développée pour le framework Symfony est appelée « **Bundle** »

Symfony est constitué de nombreuses bibliothèques qui ont elles-mêmes leurs propres dépendances (on dit qu'il « dépend » d'elles). **Composer est donc indispensable pour pouvoir installer ce framework** car c'est lui qui va s'assurer que chaque bibliothèque aura tout ce qu'il faut pour fonctionner.

Sans Composer ou plus globalement sans gestionnaire de dépendances, il serait extrêmement **compliqué de gérer manuellement** :

1. **La mise à jour** des bibliothèques externes.
2. **Leurs dépendances** avec d'autres bibliothèques.
3. **L'autoload** des classes utilisées par ces librairies.

Derrière la magie de Composer… il y a Packagist !

Packagist est une sorte de **gros annuaire** référençant toutes les informations propres à chaque bibliothèque. On y retrouve notamment :

- Le **mainteneur** principal
- Le **site web** de la bibliothèque
- Le **nombre de téléchargements**
- …

Ce qui nous importe le plus, ce sont les **sources** ainsi que les **dépendances**. Composer va récupérer les informations nécessaires sur ce site pour pouvoir **télécharger les bibliothèques et leurs dépendances**.

Composer sera extrêmement utilisé tout au long de notre développement via des lignes de commande, afin d'**installer** et **mettre à jour** nos bundles.

Vérifier que Composer a bien été installé

Vérifions que Composer a bien été installé avec la ligne de commande :

composer --version

php composer.phar --version

Installer des bundles

L'installation **d'une dépendance spécifique** se fait via la ligne de commande suivante (sans les * évidemment) :

composer require <nom_du_bundle>

L'installation de **l'ensemble des dépendances** nécessaires à un projet se fait via la ligne de commande :

composer install

Mettre à jour des bundles

La **mise à jour des dépendances** se fait via la ligne de commande suivante :

composer update

Mais comment Composer va-t-il pouvoir récupérer les sources des bibliothèques ?

La plupart du temps c'est par l'intermédiaire du gestionnaire de version Git qu'il récupère et télécharge les archives des sources concernées.

3. Gestionnaire de version (Git)

Lorsqu'on travaille à plusieurs sur un même projet, et donc sur le même code source, il est important de pouvoir le modifier **simultanément**, **en sécurité** et aussi de **manière explicite**. Un logiciel de **versioning** c'est tout ça à la fois !

Vous pouvez :

- **Coder en parallèle** avec un autre développeur et vos modifications seront fusionnées.
- **Revenir sur des versions précédentes** de votre application (les fameux backups !)
- **Documenter** vos modifications (qui a fait quoi).

Ici, Git va donc nous permettre de récupérer le code source des bibliothèques avec lesquelles on travaille : **nos fameuses dépendances**.

Vérifions que Git a bien été installé avec la ligne de commande :

git --version

Configuration de Git

Commençons par configurer notre Git en définissant un **pseudo** et d'une **adresse email**.

git config --global user.name "votre_pseudo"

git config --global user.email "votre_email@mail.com"

4. Symfony CLI

Comme la majorité des frameworks, Symfony possède sa propre **CLI** (**C**ommand-**L**ine **I**nterface).

La CLI confère aux développeurs **un ensemble d'outils** utiles au développement et à l'exécution de l'application en local, comme par exemple :

- La création d'un projet Symfony
- La gestion et la sécurité du projet
- D'exploiter un serveur local rapide
- …

Ces fonctionnalités sont contenues à l'intérieur d'un client téléchargeable, nommé « **binaire** » Symfony.

Lien de téléchargement de Symfony

Il est possible d'installer la CLI via :

- Un **installateur de paquets** comme par exemple **Homebrew** (macOS) ou **Scoop** (Windows)
- Le **téléchargement du binaire** depuis **GitHub**.

Récapitulatif

Pour pouvoir développer avec le framework Symfony, vous devez avoir installé au moins :

- Le langage **PHP**
- Le gestionnaire de dépendances **Composer**
- Le gestionnaire de versions **Git**
- La **Symfony CLI**

La ligne de commande suivante permet de savoir rapidement **si votre environnement est bien opérationnel** :

symfony check:requirements

Mise en place d'un projet Symfony

Création du projet

La création d'un nouveau projet Symfony se fait via la ligne de commande suivante :

symfony new <nom_projet>

Cette commande va installer un projet Symfony avec une **configuration minimale** (utile pour les APIs, application de Console et Services).

Néanmoins, la plupart du temps, nous préférons installer notre projet Symfony **avec tous les paquets utiles** à la création d'une application web standard. Il faudra alors ajouter à cette ligne de commande le flag ***--webapp*** (anciennement --full sur Symfony <= 5.*).

symfony new <nom_projet> --webapp

Notez qu'il existe également le flag --demo, téléchargeant un projet de démonstration complet. Ce flag n'est bien entendu pas à utiliser lors de l'installation d'un nouveau projet sur lequel vous souhaitez travailler, mais il s'avère très utile pour **apprendre la structure et le fonctionnement de Symfony**.

symfony new <nom_projet> --demo

Lancement du serveur

On distingue **2 manières** de démarrer un serveur Symfony :

Démarrage en tâche principale

Démarrer le serveur en tâche principale **empêche d'exécuter d'autres lignes de commande** pendant que le serveur tourne.

symfony server:start ou ***symfony server***

Démarrage en tâche de fond

Démarrer le serveur en tâche de fond **permet d'exécuter d'autres lignes de commande** pendant que le serveur tourne. On précisera cela avec le flag -d.

symfony server:start -d

Cette méthode sera à privilégier car nous verrons par la suite que nous aurons régulièrement recours à l'invite de commande pour effectuer d'autres opérations sur notre projet.

Accès au site

Le serveur est systématiquement démarré sur le **localhost** (**127.0.0.1**) au port 8000. Si plusieurs serveurs sont lancés simultanément, alors les projets seront accessibles sur les ports successifs : 8001, 8002…

- 127.0.0.1:8000
- localhost:8000

Le projet étant encore vierge et aucune page d'accueil n'étant définie, la page suivante sera affichée par défaut.

On distingue également une barre en bas de page. Il s'agit de la **barre de débogage** de Symfony (plus communément appelée « Profiler »). Cette barre contient de nombreuses informations extrêmement utiles lors de votre développement.

Arrêt du serveur

Lorsqu'on termine sa session de travail avec Symfony, le serveur est automatiquement stoppé. Cette opération n'est donc pas nécessaire. Notez qu'il peut néanmoins parfois être utile de réaliser cette opération manuellement avec la commande : ***symfony server:stop***