Base de Symfony

Détail du cours :

- **Date**: 05/2024
- **Symfony**:
  - CLI 5.8.19
  - Symfony 7.0.7
- **Node**: 21.7.1
- **Package Manager**:
  - npm 10.5.2
  - composer 2.7.6
- **OS**: win32 x64
- **Inspiration** : https://www.youtube.com/watch?v=nnIBXYGdXHk&list=PLI3CtU4THqPawV0hRF8Qqn0RVEHSjYgfy

Symfony est un framework back-end en PHP développé par les français de "SensioLabs".

Il permet de faciliter le travaille des développeurs back en gérant de façon optimisé les taches les plus répétitives du développeur back-end.

Avec le temps il permet de plus en plus l'utilisation de fonctionnalités front, même si on n'est pas sur une SPA non plus (quoi que).

Nous allons voir les bases ensemble, mais pour pousser plus loins, toute la documentation est disponible sur le site officiel :

https://symfony.com

1. Installation

1. php

Symfony étant un framework PHP, pour l'utiliser il faut une installation PHP sur votre ordinateur.

Faites attention à ce que les extensions suivante soit bien activé dans le fichier **php.ini**;

```
extension=fileinfo
extension=intl
extension=pdo_mysql
; et peut être d'autres.
```

2. Composer

Pour fonctionner, Symfony dépend du gestionnaire de bibliothèque PHP nommé composer, qui lui même a besoin d'une installation php.

3. Symfony Cli

Ce qui suis est valable pour une installation sur windows.

Symfony utilise sont propre CLI (Command Line Interface). Pour l'installer, on peut le faire manuellement en le téléchargeant et en l'ajoutant aux variables d'environnement.

Mais nous allons nous mâcher le travail en installant "SCOOP".
SCOOP est un installateur de programme en ligne de commande.

https://scoop.sh/

Il vous faudra utiliser dans powershell la seconde commande ci dessous, et possiblement la première commande si c'est la première fois que vous lancer un script avec powershell:

```
> Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
> irm get.scoop.sh | iex
```

Nous pourrons ensuite utiliser scoop en ligne de commande.
Faisons le tous de suite pour installer Symfony :

```
scoop install symfony-cli
```

Symfony est maintenant installé, passons à la création d'un nouveau projet.

4. Extensions VScode et paramètres utiles

- PHP INTELEPHENSE
- PHP NAMESPACE RESOLVER
- TWIG LANGUAGE 2
- PHP InteliSense (optionnellement)

Et dans les paramètres :

- Emmet Include languages inclure dans twig le html.

2. Création d'un nouveau projet

Pour créer un nouveau projet, plaçons nous là où nous souhaitons le dossier de ce nouveau projet et tapons la commande suivante :

```
# Pour une application web classique
symfony new nom_projet --version="7.0.*" --webapp

# Pour un micro service
symfony new nom_projet --version="7.0.*"
```

- "--webapp" est optionnel mais permet d'ajouter automatiquement tous les paquets les plus utile à la création d'un application web.
- "--version" Permet de préciser la version voulu de Symfony;

Votre projet Symfony est maintenant créé.

Pour le lancer, pas besoin d'utiliser docker ou autre serveur local. Symfony comporte son propre serveur local (On a quand même besoin d'une installation de PHP sur notre ordinateur).

```
symfony serve;
```

Par défaut symfony se lancera à l'adresse suivante :

- http://127.0.0.1:8000

Et tant qu'on sera en environnement de développement, si on n'a pas configuré de page d'accueil, il affichera une page avec les liens vers la documentation.

La première fois qu'on lance un projet symfony, il y a beaucoup d'élément à traiter qu'il mettra en mémoire cache pour les prochaines fois.
Donc le premier chargement et à chaque fois que le cache sera vidé, le chargement peut être long.

3. Structure d'un projet Symfony

Voyons de quoi est composé un projet symfony ensemble.

- ".git" invisible dans vscode, les projets symfony font un git init par défaut.
- "bin" ce dossier comprend la gestion des commandes de symfony, on n'y touchera pas.
- "config" contient toute la configuration des différents paquets utilisé par Symfony. On y viendra que si l'on souhaite paramétrer certains outils.
- "migrations" commence vide mais contiendra toute nos mises à jours de la BDD.
- "public" est le dossier qui doit être la racine de notre serveur quand on déploie un projet symfony. Il contient actuellement simplement notre "index.php".
- "src" va être notre principale zone de travail. Il contient :
    - "Controller" Contiendra tous nos futurs Controllers.
    - "Entity" Ce sont des classes qui représenterons nos tables en BDD.
    - "Repository" Eux seront nos Models.
    - "Kernel.php" C'est le coeur de Symfony, on n'y touchera pas.
- "template" Il contiendra toutes nos vues.
    - "base.html.twig" Est la base de toute nos pages il contient notre head et notre body.
- "tests" comme son nom l'indique sert aux tests de symfony, on n'en aura pas besoin tout de suite.

- "translations" sert aux traductions, on ne l'utilisera pas pour l'instant.
- "var" contiendra tous nos logs, c'est à dire les détails des réussites et échecs de notre projet.
- "vendor" contient tous les paquets utilisés par notre projet Symfony.
- ".env" contient toute nos variables d'environnements.
    - Environnement de travail,
    - paramètre de connexion à la BDD,
    - mailer,
    - clef secrètes et j'en passe.

  Ce fichier est commité par git, pour ne pas dévoiler vos informations, il est conseillé de créer un fichier ".env.local" qui lui ne sera pas commité.
- ".env.test" de même mais réservé à l'environnement de test.
- ".gitignore" automatique nous avons un gitignore déjà bien rempli.
- "composer.json" la liste des paquets utilisés par le projet.
- "composer.lock" le détail des versions et options utilisées pour chaque paquet.
- "compose.override.yaml" et "compose.yaml" servent au paramétrage de docker si on souhaite générer une BDD directement ou gérer le serveur de mail.
- "phpunit.xml.dist" quelques paramétrages de php pour le projet.
- "symfony.lock" le détails des bibliothèques utilisées par Symfony.