



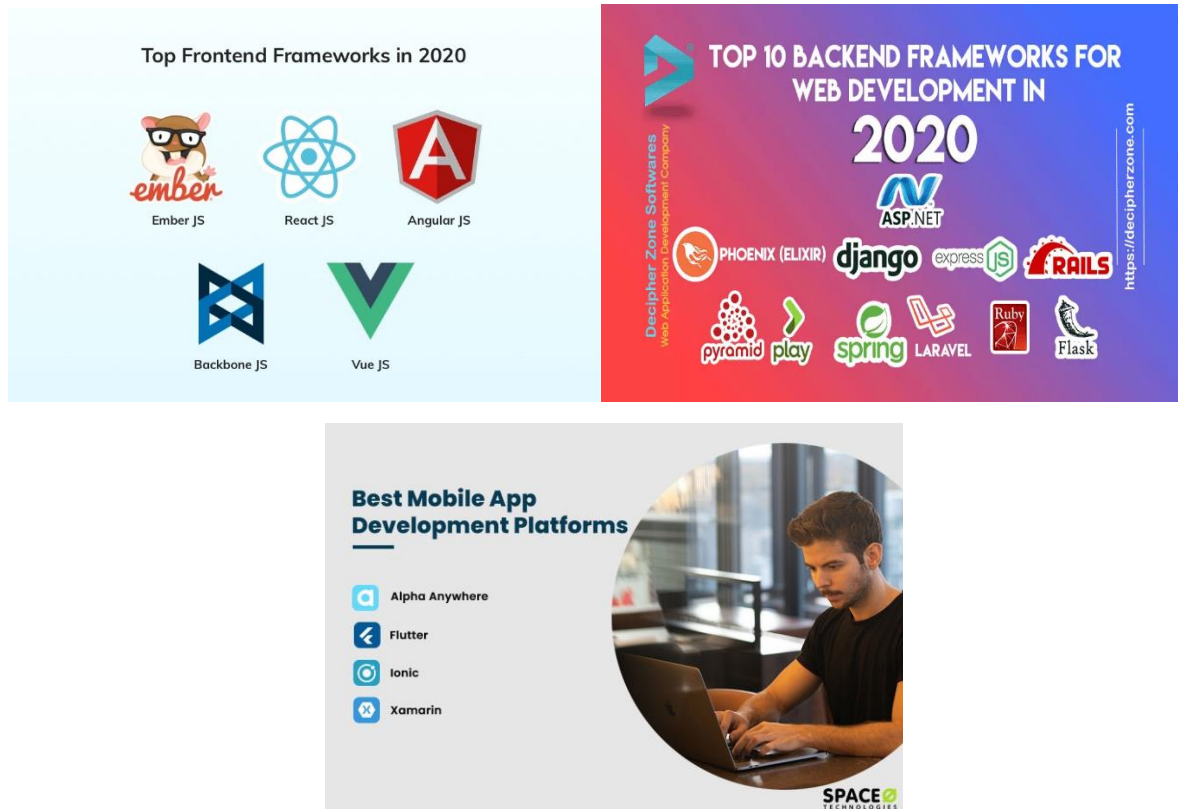
TP1 Web 2.0 : Symfony 6 FIA2-GL

Installation et Programmation contrôleur Web

sofiene.benahmed@issatso.rnu.tn

Table des matières

- Introduction
 - Prérequis
- Étape 1 : Prise en main de l'environnement d'exécution PHP en ligne de commande pour Symfony
 - Étape 1-a : vérification de l'environnement de mise au point PHP
 - Étape 1-b : découverte de Composer
 - Étape 1-c : ligne de commande Symfony
- Étape 2 : Création d'un contrôleur
- Étape 3 : Construction de Vue : le fichier templates/base.html.twig



Best Mobile App Development Platforms (**Last Updated:** February 2, 2022)

Introduction

Cette séance permet de familiariser les étudiants avec l'environnement de programmation PHP et de mise au point basé sur Symfony, pour l'écriture de programmes interactifs en PHP lancés en ligne de commande.

Prérequis

On considère que vous avez acquis les notions de base du langage PHP.

Étape 1 : Prise en main de l'environnement d'exécution PHP en ligne de commande pour Symfony

Cette étape consiste à prendre en main l'environnement d'exécution de PHP en ligne de commande, ainsi que certains outils de mise au point pour le framework Symfony

Étape 1-a : vérification de l'environnement de mise au point PHP

1. Installer composer à travers ce lien : <https://getcomposer.org/download/>
2. Lancez un terminal
3. Vérifiez que la commande composer est présente sur notre installation de PHP, et que vous disposez d'une version de Composer supérieure ou égale à 2 :
4. `composer -V`

par exemple :

```
Composer version 2.1.6 2021-08-19 17:11:08
```

5. Vérifiez que l'interpréteur PHP en ligne de commande est présent, et que la version de PHP installée est supérieure ou égale à 7.4 :
6. `php -v`

par exemple :

```
PHP 8.1.1 (cli) (built: Dec 15 2021 10:36:13) (NTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.1.1, Copyright (c) Zend Technologies
```

Étape 1-b : découverte de Composer

L'objectif de cette séquence est d'examiner les fonctionnalités de type ligne de commande fournies par la console Symfony, pour les développeurs

Définition :

Composer permet d'installer des centaines de packages librement disponibles. On les trouve sur Packagist . Il permet de gérer les dépendances d'un projet et également de créer le squelette d'une application Symfony et également d'installer des *recettes flex* (recipes) comme *composer*

require mailer pour installer *SwiftMailer* ou *composer require api* pour installer *APIPlatform* avec toutes ses dépendances.

Vous allez utiliser la commande *create-project* de *composer*, qui va créer le répertoire de développement d'une application *Symfony* et y télécharger le code source d'un squelette d'application.

1. Avec l'invite de commandes créer un nouveau dossier et se déplacer dedans.
2. Taper *composer create-project symfony/skeleton demo*
3. Taper *cd demo*
4. Lancer Visual studio Code en tapant *code* . (le point est exprès)
Voici la signification des arguments :
create-project commande *create-project* de *composer* (cf. documentation *composer*)
symfony/skeleton référence du squelette d'application standard d'application Web
demo répertoire de développement *Symfony* à créer.

Ignorez les éventuels messages affichés à la fin de l'installation (« What's next? » , « Database Configuration » et « How to test? ») qui ne sont pas pertinents dans notre contexte, pour l'instant.

Notations utile :

bin/console : script exécutable PHP présent dans le sous-répertoire *bin/*, utilisé en ligne de commande pour le développement *Symfony*;

src/ sources du squelette d'application *Symfony*. Il contient le code PHP de base de l'application sur laquelle vous travaillerez par la suite;

vendor/ contient le code de toutes les bibliothèques PHP du framework *Symfony* qui seront utilisées. Elles ont été téléchargées par *composer*.

Étape 1-c : ligne de commande *Symfony*

L'objectif de cette séquence est d'examiner les fonctionnalités de type ligne de commande fournies par la console *Symfony*, pour les développeurs

La commande *bin/console* fournie propose d'accéder en ligne de commande à différentes fonctionnalités du framework *Symfony*, notamment des utilitaires intéressants pour les phases de développement et de tests.

Essayez les commandes suivantes dans un terminal, depuis l'intérieur du répertoire du projet *Symfony* :

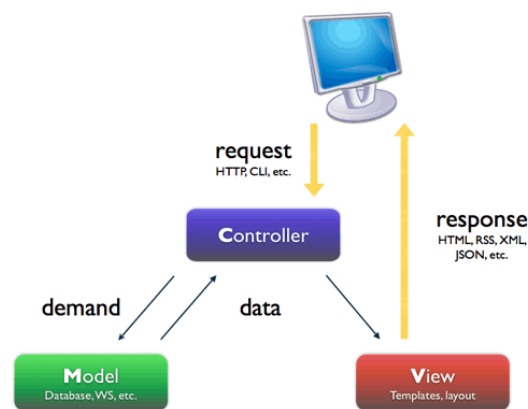
1. Vérifiez la version de *Symfony* utilisée dans le projet
2. *php bin/console -V*
3. Affichez la liste des sous-commandes disponibles (avec la sous-commande *list*, ou directement sans argument passé à *bin/console*) :
4. *bin/console [list]*

Lancer notre application web à travers *Symfony CLI* en tapant *symfony server:start*

L'architecture MVC

Pour le développement Web, les solutions les plus populaires pour organiser notre code de nos jours est la mise en place d'une **architecture MVC**. En résumé, l'architecture MVC définit un cadre d'organisation de notre code en accord avec sa nature. Ce modèle permet une séparation de notre code en **trois couches** :

- La couche **Modèle** contenant le traitement logique de vos données (les accès à la base de données se trouvent dans cette couche).
- La **Vue** est la couche où interagit l'utilisateur (un moteur de template fait parti de cette couche). Dans symfony, la couche vue est principalement faite de Templates PHP. Ces fichiers sont stockés dans les différents dossiers templates/ .
- Le **Contrôleur** est un morceau de code qui appelle le modèle pour obtenir certaines données qu'il passe à la Vue pour le rendu au client.



Étape 2 : Création d'un contrôleur

on doit installer tout d'abord le maker `symfony composer req maker --dev`

- 1) `php bin/console make:controller`
- 2) Nommer le contrôleur `BlogController`
- 3) On obtient les fichiers : `BlogController.php` et `template/blog/index.html.twig`
- 4) Examiner les 2 fichiers.
- 5) Dans le fichier `index.html.twig` changer le mot hello par salut et recharger la page `https://127.0.0.1:8000/blog`.
- 6) Dans la classe `BlogController` du fichier `BlogController.php` ajouter la méthode

```
#[Route('/', name: 'home')]
public function home()
{
    return $this->render('blog/home.html.twig');
}
```

Et pour que ça fonctionne il faut créer un nouveau fichier dans le répertoire `blog`. Voici son code pour le moment : `<h1>Soyez le Bien Venue</h1> (p>Lorem40)*2` (ajouter l'extension Emmet à Visual Studio code). Dans le navigateur chercher `https://127.0.0.1:8000/`

Le langage Twig



Définition

Twig est un moteur de templates pour le langage de programmation PHP, utilisé par défaut par le framework Symfony. Il a été inspiré par Jinja, moteur de template Python.

- Twig permet de séparer la présentation des données du traitement : Twig permet de définir en dehors de la page web des filtres que l'on pourra appliquer à la donnée.
- Twig permet la personnalisation de page web : Un block menu, un block recherche, un block contenu. . . le tout défini dans un template lui-même héritable.
- Twig permet de rendre les pages web plus lisibles, plus claires. Twig par son langage est moins invasif que PHP et se substitue à celui-ci.
- Twig est rapide.
- Twig apporte de nouvelles fonctionnalités.
- Twig est facile à apprendre.

Exercice

- 1) Modifier le fichier `blog/home.html.twig` à fin qu'il affiche les entiers entre 1 et 10
- 2) Modifier le fichier `blog/home.html.twig` à fin qu'il affiche les entiers entre 1 et x (x est un paramètre de la méthode `render`).

Étape 3 : Construction de Vue : le fichier templates/base.html.twig

Il est nécessaire d'installer twig en tapant `symfony composer req twig`
Observer le fichier `base.html.twig` :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>{% block title %}Welcome!{% endblock %}</title>
    {% block stylesheets %}{% endblock %}
  </head>
  <body>
    {% block body %}{% endblock %}
    {% block javascripts %}{% endblock %}
  </body>
</html>
```

Ce fichier joue le rôle de référence de mise en page pour tout les interfaces. Il décrit tout ce qu'on doit mettre dans le head et le body de la page. Pour ce faire bootswatch nous plusieurs templates à exploiter. Visitez le site <https://bootswatch.com> et observer les thèmes offerts par bootstrap. Choisir le thème Flatly puis afficher dans le navigateur le fichier `flatly/bootstrap.min.css`. Il suffit maintenant d'insérer le lien vers ce thème.

```
<!DOCTYPE html>
<html>
  <head>
```

```
<meta charset="UTF-8">
<title>{% block title %}Welcome!{% endblock %}</title>
<link rel="stylesheet" href="https://bootswatch.com/5/flatly/bootstrap.min.css">
{% block stylesheets %}{% endblock %}
</head>
```

Puis choisir le code source d'un Navbar de votre choix et l'insérer juste après le `<body>` dans le fichier.

```
<body>
#insérer ici le code associé votre Navbar puis éliminer les composants qui ne sont pas nécessaire à ce moment
{% block body %}{% endblock %}
{% block javascripts %}{% endblock %}
</body>
```

En fin pour que les pages s'inscrivent au nouveau thème il suffit d'ajouter à l'entête de fichier `index.html.twig` par exemple :

```
{% extends 'base.html.twig' %}
```

Et indiquer le contenu de body :

```
{% extends 'base.html.twig' %}
{% block body %}
#insérer ici votre contenu exemple :
<article>h2{Titre de l'Article }+div.metadata{Edité le
06/02/2022}+div.content>img+(p>lorem15)*2+a.btn.btn-primary{Lire la suite})*3
{% endblock %}
```

Pour accéder au détail de chaque Article on ajoute dans le fichier `BlogController` la méthode suivante :

```
#[Route('/blog/12', name: 'blog_show')]

public function show()
{
    return $this->render('blog/show.html.twig');
}
```

Dans le fichier `index.html.twig` ajouter (Pour chaque article)

```
<a href="{{ path() }}" class="btn btn-primary">Lire la suite</a>
```

Dans le dossier blog créer un nouveau fichier `show.html.twig` et écrire dedans le code suivant :

```
{% extends 'base.html.twig' %}

{% block body %}

<article>
<article>
    <h2>Titre de l'Article </h2>
    <div class="metadata">Edité le 06/02/2022</div>
    <div class="content">
        
        <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Rem
debitis maiores facilis exercitationem quas explicabo.</p>
```

```
<p>Sequi molestiae natus voluptates blanditiis dolores nihil earum  
amet, vel dignissimos voluptatem, beatae animi quos.</p>  
</div>  
</article>  
</article>  
{% endblock %}
```

Référence :

1. <https://symfony.com>
2. <https://openclassrooms.com/fr/courses/5489656-construisez-un-site-web-a-l-aide-du-framework-symfony-5>
3. Livre : Développer avec Symfony2 Clément Camin (Eyrolles)