# SYMFONY : créer des fonctionnalités dans SYMFONY

Nous allons voir les 3 piliers fondamentaux de SYMFONY :

- Les contrôleurs (traitement PHP des données)
- Doctrine (accès aux données, création de requête SQL)
- TWIG (langage de rendu)

## La logique du Controller

Une application WEB est une application qui est capable de répondre à un navigateur lorsque celui-ci appel une adresse (www.monsite.com/accueil | www.monsite.com/blog), il faut que l'application soit capable de comprendre que l'on est appelé à cet endroit-là et de fabriquer une réponse qu'elle va pouvoir renvoyer au navigateur pour que celui-ci l'affiche, c'est par exemple fabriqué une page HTML que l'on va renvoyer au navigateur ou bien renvoyer une redirection par exemple, que le navigateur rappel une nouvelle adresse et que le processus se remette en place au début. A quoi sert un controller dans SYMFONY :

- Ecouter une adresse, ce que l'on appel dans SYMFONY une route,
- Ecouter et analyser une requête http que le navigateur à envoyer à la route
- Fabriquer une réponse http qui soit convenable
- La renvoyer au navigateur afin de générer un affichage ou une redirection ou un téléchargement etc…

Strucure des dossiers :

- src : entièreté des codes PHP de notre application
- templates : fichiers qui gèrent l'affichage , le rendu (TWIG)

Dans la console, nous allons créer un controller :

php bin/console make:controller

Puis taper le nom du controller (première lettre en majuscule) : BlogController

Voir les fichiers crées par SYMFONY

created: src/Controller/BlogController.php

created: templates/blog/index.html.twig

Modifier le fichier 'blog/index.html.twig' (le titre <h1>) et envoyer dans l'URL la route '/blog', nous pouvons observer les modifications en front sur le navigateur.

---

Création de la méthode home() dans le fichier BlogController.php seulement :

```php
/**
     * @Route("/", name="home")
     */
    public function home(): Response
    {
        return $this->render('blog/home.html.twig', [
            'title' => 'Bienvenue sur le blog Symfony',
            'age' => 25
        ]);
    }
```

Création du fichier home.html.twig dans le dossier 'template/blog', ajouter 2 paragraphes contenant du texte, puis aller à http://localhost:8000/ et actualisé. On observe bien les 2 paragraphes au premier chargement de l'application.

Nous sommes maintenant capable de créer un fichier template en ayant définit une route.

---

# Langage TWIG

TWIG est un langage de rendu. Avantage :

- simplicité du langage : facilite l'écriture des affichages, apporte beaucoup de fonctionnalités
- Absence de PHP au sein de nos affichages, permet d'abstraire les affichages de balises PHP
- Plus simple pour un intégrateur

TWIG est une librairie que l'on peut utiliser dans d'autres projets PHP, sans SYMFONY.

Syntaxe :

- {{ }} → double accolades, ce que l'on appelle l'interpolation.
- {% %} → commande (cmd ex : if() )

Dans home.twig.html rajouter les lignes :

```
<h1>{{ title }}</h1>
{# On affiche le contenu de la variable 'title' avec le moteur TWIG #}

{% if age > 18 %}
    <p>Tu es majeur !!</p>
{% else %}
    <p>Tu es mineur !!</p>
{% endif %}
```

---

Nous allons maintenant nous occuper de la structure global de l'aaplication. Rendez vous sur https://bootswatch.com/flatly/

Nous allons ouvrir le fichier base.html.twig qui est le template de base de l'application, c'est la structure de base de l'application. C'est un gabarit de base qui sera dupliqué sur toute nos pages. A chaque que nous créerons une autre page, nous dirons qu'elle s'inscrit dans le template de base base.html.twig

```
{% block body %}
```

Chaque page va pouvoir personnaliser ces différents blocks.

Récupérer la <nav> et la copier/coller juste au-dessus du block 'body' et modifier la <nav> et les liens.

Mettre le block {% block body %} dans une 

Pour hériter d'un template dans le template de base 'base.html.twig' ajouter les lignes suivantes dans home.html.twig :

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

{% block body %}
```

Tout le contenu, les paragraphes se trouve ici !! Dans le bloc body !

```
{% endblock %}
```

Actualiser la page google http://localhost:8000/

---

Cliquer sur les liens du menu : Articles et SymBlog pour apercevoir le rendu. Nous allons maintenant nous occuper de la page 'Articles'. Pour cela rendons nous dans le fichier index.html.twig

Pourquoi index.html.twig ? parce que la fonction index() dans le fichier BlogController.php renvoie la page 'blog/index.html.twig'

Supprimer tout le contenu du block body. Ajouter ce code 3 fois dans le block body :

```
<section class="articles">

        <article>
            <h2>Titre de l'article</h2>
            <div class="metadata">Ecrit le 30/12/2019 à 14:44 dans la catégorie Politique<
</div>
            <div class="content">
                <img src="" alt="">
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam neque nisi
, accumsan id quam eget, pulvinar maximus libero. Donec eget sagittis nibh. Morbi auctor,
augue non dignissim mattis, arcu eros ornare tellus, semper condimentum mauris metus ac ma
ssa. Vestibulum a porta mi. </p>

                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam neque nisi
, accumsan id quam eget, pulvinar maximus libero.</p>

                <a href="" class="btn btn-primary">Lire la suite</a>
            </div>
        </article>

    </section>
```

Nous allons procéder à l'affichage d'un seul article,  rendons nous dans le fichier BlogController.php et créer la méthode show()

```
/**
     * @Route("/blog/12", name="blog_show")
     */
    public function show(): Response
    {
        return $this->render('blog/show.html.twig');
    }
```

On ajoute l'interpolation sur les liens  des paragraphes 'Lire la suite'

{{ path() }} → chemin avec 2 paramètres : le nom de la route 'blog_show'

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

On peut observer que chaque lien nous renvoi pour l'instant vers l'ID 12, c'est normal, cela fait référence à la méthode show() et la route que l'on a définit (@Route('/blog/12', name='blog_show'))  nous ferons la modification plus tard. Modifier la route pour voir que symfony met à jour automatiquement les liens.

Nous allons maintenant créer le fichier show.html.twig dans le dossier 'templates/blog' et ajouter le code suivant : 1 paragraphe + les modifications.

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

{% block body %}

    <section class="row articles">

            <article  class="col-md-3 mx-auto text-center border border-dark mb-3 p-3 mt-
3 mr-1">
                <h2>Titre de l'article</h2>
                <div class="metadata">Ecrit le 30/12/2019 à 14:44 dans la catégorie Politi
que</div>
                <div class="content">
                    <img src="http://placehold.it/250x150" alt="">
                    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam neque
nisi, accumsan id quam eget, pulvinar maximus libero. Donec eget sagittis nibh. Morbi auct
or, augue non dignissim mattis, arcu eros ornare tellus, semper condimentum mauris metus a
c massa. Vestibulum a porta mi. </p>

                    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam neque
nisi, accumsan id quam eget, pulvinar maximus libero.</p>

                    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam neque
nisi, accumsan id quam eget, pulvinar maximus libero. Donec eget sagittis nibh. Morbi auct
or, augue non dignissim mattis, arcu eros ornare tellus, semper condimentum mauris metus a
c massa. Vestibulum a porta mi. </p>

                    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam neque
nisi, accumsan id quam eget, pulvinar maximus libero. Donec eget sagittis nibh. Morbi auct
or, augue non dignissim mattis, arcu eros ornare tellus, semper condimentum mauris metus a
c massa. Vestibulum a porta mi. </p>
                </div>
            </article>

        </section>

{% endblock %}
```