TWIG - SYMFONY

LE MOTEUR DE TEMPLATE FLEXIBLE, RAPIDE ET SÉCURISÉ POUR PHP

TWIG - SYMFONY

Un Template est en fait un fichier texte qui peut générer tous les formats basés sur du texte (HTML, XML, CSV, JSON, Latex, ...).

Généralement on utilise des **templates** "PHP", c'est à dire mêlant du code texte (souvent HTML) et du PHP.

Ensuite tout cela est interprété, et traduit en quelque chose d'affichable (souvent du HTML).

Cette solution est pratique et rapide à mettre en place, mais syntaxiquement lourde à écrire, et souvent rejetée par les **intégrateurs** qui n'aiment pas écrire du PHP.

De nombreux Framework utilisent donc des moteurs de Template ne se basant pas sur PHP directement. (en réalité, tout le code sera traduit en PHP, puis en HTML ensuite).

TWIG INSTALLATION DANS UN PROJET - SYMFONY

Pour installer **Twig** dans un projet PHP il faut déjà installer **Composer** qui est le **gestionnaire de dépendance** pour les projets PHP.

Rdv sur ***https://getcomposer.org/*** pour obtenir la version adaptée à votre système d'exploitation.
Vous devez créer un répertoire de projet sur votre serveur, par exemple **Atelier_Twig** et ouvrir une fenêtre de **terminale** dans ce dossier.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

PS C:\wamp64\www\TP_DWWM\Back\Atelier_Twig_Presentation> composer require "twig/twig:^3.0"
```

Ce qui aura pour effet d'installer Twig dans votre projet. Un répertoire **Vendor** sera créé ainsi que 2 fichiers **composer.json** et **composer.lock**.
Vous devrez créer par exemple un script d'entrée pour votre application **index.php** et un répertoire **templates** qui contiendra tous vos fichiers Twig.

TWIG INSTALLATION DANS UN PROJET - SYMFONY

Anatomie de votre fichier **index.php** de démarrage :

index.php

Back > Atelier_Twig_Presentation > index.php > ...

```php
<?php

require_once 'vendor/autoload.php';

$loader = new \Twig\Loader\FilesystemLoader('templates');
$twig = new \Twig\Environment($loader, [
    'cache' => false,
]);

?>

```

TWIG - SYNTAXE - SYMFONY

Le **moteur de Template** de Symfony est Twig qui utilise 3 notations :

{{ ... }} "Dit quelque chose" : permet d'écrire une variable ou le résultat d'une expression

{% ... %} "Fait quelque chose" : permet de contrôler la logique du Template. Permet l'exécution de fonctionnalités comme les boucles ou les tests.

{# ... #} "Commente quelque chose" : Permet d'écrire un commentaire.

Twig utilise aussi la notion de filtre (noté avec |). Qui permet de faire des manipulations sur les variables.

TWIG INSTALLATION DANS UN PROJET - SYMFONY

Enrichissement de votre fichier **index.php** et liaison avec le fichier Twig **index.html.twig**

index.php × index.html.twig

Back > Atelier_Twig_Presentation > index.php > ...

```php
<?php
require_once 'vendor/autoload.php';
$loader = new \Twig\Loader\FilesystemLoader('templates');
$twig = new \Twig\Environment($loader, [
    'cache' => false,
]);

$prenom = "Sacha";

echo $twig->render('index.html.twig', [
    'prenom' => $prenom,
]);

?>
```

Avec en prime, la façon de passer une variable au fichier Twig

TWIG INSTALLATION DANS UN PROJET - SYMFONY

Au niveau du script Twig : **index.html.twig**

index.php | index.html.twig

Back > Atelier_Twig_Presentation > templates > index.html.twig

```
1
2    <h1>{{prenom}}, bienvenue dans cet Atelier Twig </h1>
3
4
```

Pour obtenir ce rendu dans votre navigateur lorsque vous demandez le script **index.php**

Sacha, bienvenue dans cet Atelier Twig

TWIG - SYNTAXE - SYMFONY

Twig intègre une très longue liste de tags, de filtre et de fonctions que vous pouvez retrouver dans la documentation officielle : https://twig.symfony.com/.

Vous pouvez également créer vos propres filtres ou fonctions.

Twig intègre également les structures de contrôles classiques

{% if ... %} ... {% elseif ... %} ... {% else %} ... {% endif %}

{% for ... in ... [if ...] %} ... {% endfor %}

TWIG INSTALLATION DANS UN PROJET - SYMFONY

Un autre exemple avec utilisation d'une **boucle for** :

index.php

```php
<?php
require_once 'vendor/autoload.php';
$loader = new \Twig\Loader\FilesystemLoader('templates');
$twig = new \Twig\Environment($loader, [
    'cache' => false,
]);

$rongeurs = ["Lapin", "Souris", "Rat", "Ecureuil", "Campagnole"];

echo $twig->render('index.html.twig', [
    'rongeurs' => $rongeurs,
]);

?>
```

index.html.twig

```twig

<ul>
    {% for rongeur in rongeurs %}
        <li>
            <a href="">{{rongeur | upper }}</a>
        </li>
    {% endfor %}
</ul>
```

- LAPIN
- SOURIS
- RAT
- ECUREUIL
- CAMPAGNOLE

TWIG - HÉRITAGE ET BLOCKS - SYMFONY

Il est fréquent dans un projet d'avoir des éléments communs à chacune des pages (header, menu, footer, ...). La logique de Symfony et d'un Framework en général, consiste à ne jamais dupliquer du code. Ainsi, il ne devrait y avoir qu'une seule fois la structure du site web qu'on pourrait réutiliser autant de fois que nécessaire.

C'est possible avec les templates et TWIG. Nous allons définir une base, et dans cette base définir des emplacements dans lesquels nous pourrions venir mettre les éléments en fonction de la page choisie.

Dans TWIG cela s'appelle des **blocks**.

Prenons le fichier **layout.html.twig** par défaut :

TWIG - HÉRITAGE ET BLOCKS - SYMFONY

index.php | layout.html.twig | index.html.twig

Back > Atelier_Twig1 > templates > layout.html.twig

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>
            {% block title %}Bienvenue sur ce Site Web
            {% endblock %}
        </title>
            {% block stylesheets %}
                <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
                integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
                crossorigin="anonymous">
                <link rel="stylesheet" href="css/style.css">
            {% endblock %}
    </head>
    <body>

        <div class="container">
        <h1><img src="img/html.png" alt="html" width="80px">Atelier Twig</h1>
        {% block body %}{% endblock %}
        </div>

        {% block javascripts %}
        <script src="https://code.jquery.com/jquery-3.5.1.js"
        integrity="sha256-QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc="
        crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
        integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
        crossorigin="anonymous"></script>
        {% endblock %}
    </body>
</html>
```

Dans le **layout** de base, nous voulons intégrer **BootStrap**, **JQuery** et **BootStrap JS**

Puis une base avec un titre **h1** qui contient une **image.**

TWIG - HÉRITAGE ET BLOCKS - SYMFONY

Ce Template définit le squelette du document HTML.
Il ne devrait y avoir qu'une seule fois ce type de code dans votre projet. Les blocks sont identifiés avec les ligne {% block nom %}.

Le Template définit ici 4 blocks : title, stylesheets, body et javascripts

C'est à dire qu'à chaque fois qu'on va utiliser ce Template (de façon héritée) on pourra définir le contenu de chacun de ces 4 blocks, sans modifier le reste de la structure.

TWIG - HÉRITAGE ET BLOCKS - SYMFONY

index.php | layout.html.twig | index.html.twig ×

Back > Atelier_Twig1 > templates > index.html.twig

```twig
{% extends "layout.html.twig" %}

{% block title %}{{ parent() }} - Fils{% endblock %}

{% block body %}
<form action="#" method="POST">
    <table class="table">
        <tr>
            <td><label for="prenom">Prénom</label></td>
            <td><input type="text" name="prenom" id="prenom"></td>
        </tr>
        <tr>
            <td><label for="nom">Nom</label></td>
            <td><input type="text" name="nom" id="nom"></td>
        </tr>
        <tr>
            <td><input type="submit" name="btnSubmit" value="Envoyer"></td>
            <td><input type="reset" name="btnReset" value="Annuler"></td>
        </tr>
    </table>
</form>
{% endblock %}
```

TWIG - HÉRITAGE ET BLOCK - SYMFONY

Ce Template **index.html.twig** est un Template enfant/fils du Template parent **layout.html.twig**.

La ligne 1, permet de définir que ce Template "hérite" (extends) du Template **layout.html.twig**.
C'est grâce à cette ligne que l'on peut écrire dans les blocks existant.

Remarque, il n'est pas nécessaire de définir dans l'enfant tous les blocks présents dans le parent.

CRÉDITS

OEUVRE COLLECTIVE DE L'AFPA

Sous le pilotage de la DIIP
et du centre sectoriel Tertiaire

EQUIPE DE CONCEPTION

M. Restoueix Sacha (Formateur)

Date de mise à jour : 08/01/2021
Date de dépôt légal : 2021
