5 minutes

Introduction à Twig

Qu'est-ce que `Twig` ?

Nous avons déjà commencé un peu à voir `Twig`, dans ce chapitre nous allons l'étudier en détail.

`Twig` est un moteur de `templates` pour le langage `PHP`. Il est recommandé et utilisé par défaut par `Symfony`.

Un moteur de `template` permet de générer dynamiquement des pages `HTML` à partir de `templates` qui sont écrits avec une syntaxe particulière (syntaxe `Twig`) et des variables `PHP`.

Autrement dit, le moteur va prendre un `template Twig` et optionnellement des variables et générer une page `HTML` qui sera retournée au client.

C'est ce qui se passe lorsque l'on fait : `$this->render()` et que l'on passe un `template`.

Mais pourquoi utiliser `Twig` et pas simplement du `PHP` ? Tout simplement pour un important gain de productivité et pour pouvoir séparer proprement la logique dans les `Controllers` et les vues dans les `Templates`.

`Twig` fournit en effet beaucoup de fonctionnalités qui seraient très chronophages à mettre en place dans des `templates PHP`, nous les verrons toutes dans le chapitre.

Installation de l'extension pour `Twig`

Dans l'onglet `Extensions` de `VS Code`, recherchez `Twig Language 2`.

Installez l'extension.

Toujours dans `VS Code`, allez dans `Fichier > Préférences > Paramètres`.

Recherchez `files.associations` dans la barre de recherche.

Cliquez sur `Ajouter l'élément`. Dans clé mettez `*.html` et dans valeur `twig` puis sauvegardez.

Au même endroit, recherchez `emmet.includeLanguages`. Mettez `twig` en clé et `html` en valeur et cliquez sur `ok`.

Recommandations pour les noms des `templates Twig`

Pour le nommage des `templates` il est **recommandé d'utiliser le `snake case`**.

A savoir, les noms de `templates` doivent contenir uniquement des minuscules et les mots doivent être séparés par des `underscores` (_).

Par exemple : `blog_accueil.html.twig`.

Autre recommandation, il faut que les `templates` finissent par l'extension `.formatfinal.formatinitial`. Dans la très grande majorité des cas cela sera `.html.twig`.

Cela signifie que le format initial est `twig` et le format final `html`.

Structure par défaut des `templates`

Par défaut les `templates` sont placés dans le dossier `templates` à la racine du projet.

La configuration se trouve dans le fichier `config/packages/twig.yaml` :

```
twig:
  default_path: '%kernel.project_dir%/templates'

when@test:
  twig:
    strict_variables: true
```

Copier

Il est recommandé de ne pas toucher à cette structure par défaut.

`Namespaces` de `templates`

Si vous souhaitez séparer certains `templates` vous pouvez utiliser les `namespaces` de `templates`.

Il suffit d'ajouter dans la clé `paths` d'autres répertoires où seront placés les `namespaces`. Par exemple :

```
twig:
  default_path: '%kernel.project_dir%/templates'
  paths:
    'email/templates': 'email'

when@test:
  twig:
    strict_variables: true
```

Copier

Ici vous indiquez que dans le dossier `email/templates` seront situés les `templates` du `namespace email`.

Vous pouvez ensuite utiliser `@namespace` pour utiliser des `templates` de `namespace` dans les contrôleurs.

Prenons un exemple : placez la configuration que nous avons vu dans `config/packages/twig.yaml`.

Créez ensuite à la racine le dossier `email` dans lequel vous créez un dossier `email/templates`. Dans ce dernier dossier, créez un fichier `welcome_email.html.twig` :

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>
  {% block title %}Welcome!
  {% endblock %}
  </title>
  {# Run `composer require symfony/webpack-encore-bundle`
         and uncomment the following Encore helpers to start using Symfony UX #}
  {% block stylesheets %}
  {#{{ encore_entry_link_tags('app') }}#}
  {% endblock %}

  {% block javascripts %}
  {#{{ encore_entry_script_tags('app') }}#}
  {% endblock %}
</head>
<body>
  {% block body %}
  <h1>Hello</h1>
  {% endblock %}
</body>
</html>
```

Copier

Vous pouvez ensuite l'utiliser dans un contrôleur de cette manière :

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController extends AbstractController
{
  #[Route('/', name: 'index')]
  public function index(): Response
  {
    return $this->render('@email/welcome_email.html.twig');
  }
}
```

Copier