Templates

Le moteur de templates est responsable de l'affichage des vues. Il est conseillé d'utiliser Twig, plutôt que PHP, comme moteur de templates.

Installation

```
composer require twig
```

Éléments syntaxiques

Twig utilise 3 types de tags :

{{ ... }}	“affiche quelque chose” équivalent du echo de php : affiche une variable ou le résultat d'une expression.
{% ... %}	“Fait quelque chose”: tag agissant sur la logique du template ; il est utilisé par exemple pour faire des itérations, ou des contrôles.
{# ... #}	“Commente quelque chose”: équivalent du PHP /* comment */. Il est utilisé pour ajouter des commentaires sur 1 ou plusieurs lignes. Le commentaire ne sera pas présent dans la page réponse.

Twig contient aussi des filtres (filters), qui permettent de modifier le contenu affiché.

L'appel suivant affiche la variable **title** en majuscules :

```
{{ title|upper }}
```

Tags

La liste complète des tags est fournie dans la documentation twig

Afficher {{...}}

Afficher une variable	`Nom : {{nom}}`
Afficher un élément de tableau	`Utilisateur en position 0 : {{users[0]}}`
Accès aux membres (utilisation implicite du getter getNom())	`Nom d'utilisateur : {{user.nom}}`
Application d'un filtre	`Pseudo en majuscules : {{ pseudo|upper }}`
Utilisation d'une combinaison de filtres	`Message : {{ news.texte|striptags|title }}`
Affichage d'une date de type DateTime	`Date : {{ date|date('d/m/Y') }}`
Concaténation	`Identité : {{ nom ~ ' ' ~ prenom }}`

Filtres

upper	Met toutes les lettres en majuscules.	`{{ var|upper }}`

striptags	Supprime toutes les balises XML.	`{{var|striptags }}`
date	Formate la date selon le format donné en argument. La variable en entrée doit être une instance de DateTime.	`{{ date|date('d/m/Y') }}`
format	Insère des variables dans un texte, équivalent à printf.	`{{ "Il y a %s pommes et %s poires"|format(153, nb_poires) }}`
length	Retourne le nombre d'éléments du tableau, ou le nombre de caractères d'une chaîne.	`Longueur de la variable : {{ texte|length }}`
raw	Désactive l'auto-escape sur l'affichage de la variable.	`Elément HTML : {{ element|raw }}`

Faire {{#...}}

if

voir Doc twig if

Permet de poser une condition, comme en php

```
{% if online == false %}
    <p>Our website is in maintenance mode. Please, come back later.</p>
{% endif %}
```

for

voir Doc twig for

Equivalent du foreach php :

```
<h1>Members</h1>
<ul>
    {% for user in users %}
        <li>{{ user.username|e }}</li>
    {% endfor %}
</ul>
```

Fonctions utiles

attribute

Permet d'accéder au membre d'un objet :

```
{{ attribute(object, method) }}
{{ attribute(object, method, arguments) }}
{{ attribute(array, item) }}
```

isDefined

Test l'affectation d'une variable (isset de php)

```
{% if var is defined %} … {% endif %}
```

Structuration de template

Très souvent, les templates ont besoin de partager certains éléments : header, footer, barres de menus...

Twig permet de répondre à cette problématique à partir de l'héritage et des blocs dans les templates :

Soit un template père définissant le style, ainsi que quelques bloc nommés affichant ou non du contenu.
Il est possible de créer des templates fils héritant du père et remplaçant ou ajoutant leur propre contenu dans les blocs définis.

C'est ainsi que Twig permet de structurer un ensemble de pages, en uniformisant la présentation et en évitant les répétitions de code.

Héritage et blocs

Soit le template principal suivant :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>{% block title %}Test Application{% endblock %}</title>
    </head>
    <body>
        <div id="sidebar">
            {% block sidebar %}
                <ul>
                    <li><a href="/">Home</a></li>
                    <li><a href="/blog">Blog</a></li>
                </ul>
            {% endblock %}
        </div>

        <div id="content">
            {% block body %}{% endblock %}
        </div>
    </body>
</html>
```

Il est possible de définir un template enfant de la façon suivante :

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

```
{% block title %}My cool blog posts{% endblock %}

{% block body %}
    {% for entry in blog_entries %}
        <h2>{{ entry.title }}</h2>
        <p>{{ entry.body }}</p>
    {% endfor %}
{% endblock %}
```

Parent

Il est possible de réutiliser le contenu affiché un bloc du template parent en utilisant la fonction **parent()** :

```
{% block sidebar %}
    <h3>Table of Contents</h3>

    {# ... #}

    {{ parent() }}
{% endblock %}
```

Inclusion

Soit le template suivant affichant le détail d'un article :

```
<h2>{{ article.title }}</h2>
<h3 class="byline">by {{ article.authorName }}</h3>

<p>
    {{ article.body }}
</p>
```

Il est possible de réutiliser ce template pour afficher la liste des articles, par inclusion :

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

{% block body %}
    <h1>Recent Articles<h1>

    {% for article in articles %}
        {{ include('article/article_details.html.twig', { 'article': article }) }}
    {% endfor %}
{% endblock %}
```

Liens

Soit la route suivante définie dans le contrôleur Welcome :

```
use Symfony\Component\Routing\Annotation\Route;

class WelcomeController extends Controller
{
    /**
     * @Route("/", name="welcome")
     */
    public function index()
    {
        // ...
    }
}
```

Il est possible de générer l'url correspondant à cette route en passant à la fonction **path** la propriété **name** de la route : welcome

```
<a href="{{ path('welcome') }}">Home</a>
```