المعهد العالي للدراسات التكنولوجية بمدنين
Institut Supérieur des Etudes Technologiques de Médenine

Atelier Framework Coté Serveur

TP3

La gestion des vues avec Twig
(Symfony 6)

Enseignante : K.MECHLOUCH	Classe : L2DSI

1. Syntaxe de base de Twig

Le template est le meilleur moyen d'organiser et de restituer le code HTML dans une application. Les templates dans Symfony sont créés avec Twig.

1.1. Affichage des données

On utilise :

{% %} : pour faire une action : un set de variable, une condition *if*, une boucle *for* ; **{{ }}** : pour afficher quelque chose.

```
<b>{{ 'bonjour' }}</b>    => bonjour
{% set data = '<h1>bonjour</h1>' %} => <h1>bonjour</h1>
    {{ data }}
```

L'affichage des données de différents types : objet, tableau… :

Twig	PHP

{{ produit.nom }}
{{ produit.categorie.nom }}
	<?php
	echo $produit->getNom();
	echo $produit->getCategorie()->getNom();
	?>

{{ produit["nom"] }} ou {{ produit.nom }}	<?php echo $produit["nom"]; ?>
{% set categorie = 'meubles' %} {{ "il y a des #{categorie} dans notre magasin" }} {{ 2341 / 5 = #{2341 / 5} }}	<?php $categorie = 'meubles'; echo sprintf('Il y a des %s dans notre magasin', $categorie); echo printf('2341 / 5 = %d', (2341 / 5)); ?>

L'affichage de données avec suppression d'espace entre deux balises à gauche ou à droite ou les deux :

- Supprimer les espaces entre les balises HTML

```
{% spaceless %}
    <div>          <strong>foo bar</strong>
    </div>
{% endspaceless %}
```

⇨ `<div><strong>foo bar</strong></div>`

- Supprimer les espaces à gauche puis des deux côtés avec le caractère '-' :

```
<li>    {{- 'bonjour' }}    </li>
<li>    {{- 'aurevoir' -}}    </li>
```

⇨
```
<li>bonjour    </li>
<li>aurevoir</li>
```

1.2. Les variables

- Valeur simple:

```
{% set pi = 3.14 %}
{% set x = 'Bonjour' %}
```

- Tableau simple avec une série de valeurs :

```
{% set tableau=[1, 2, 3] %}
```

- Tableau avec des clés :

```
{% set tableau={key1:value1, key2:value2} %}
```

- Tableau avec valeur et clé :

```
{% set x = [3, {"mot": "Bonjour"}] %}
```

- Afficher le contenu de 'mot' donc 'Bonjour' :

```
{{ x[1]['mot'] }}
```

- Déclarer deux variables en même temps // salutation='bonjour' // qui='DSI':

```
{% set salutation, qui = 'bonjour', 'DSI' %}
```

- x contient le texte entre les 2 balises :

```
{% set x %}
    <div id="pagination">
        …
    </div>
{% endset %}
```

1.3. Concaténation

Twig	PHP

{{"Description du produit :" ~ produit.description }}	<?php
	  echo "Description du produit:".
	  $produit->getDescription();
	?>

{% set texte = salutation ~ name|lower %}
{{ texte }}
	<?php
	  $texte =
	  $salutation.strtolower($name);
	  echo $texte;
	?>

{{ (salutation ~ name)|lower }}	<?php echo strtolower($salutation. $name); ?>

1.4. Condition IF : {% if … %} … {% endif %}

```
{% if produits is empty %}
   il n'y a plus de produit
{% endif %}

{% if ((a==1 and b>0) or not c==0) and d is defined %}
   {% set resultat = (d + a * b) / c %}
   {{ resultat }}
{% endif %}

{% if 'DSI' starts with 'D' %}
   commence par D
{% endif %}

{% if 'DSI' ends with 'I' %}
   se termine par I
{% endif %}

{% if 5 not in [1, 2, 3] %}
   5 non présent
{% endif %}
```

- **is null:** si est null ;
- **is constant:** tester s'il s'agit d'une constante ;
- **is divisible by(*x*):** si est divisible par *x* ;
- **is even:** si est pair ;
- **is odd:** si est impair ;
- **is iterable:** si est du type itérable (comme une liste) ;
- **is same as:** comparer 2 variables (en php correspond = =).

```
{{ article is null? 'yes': 'no' }}    // affiche yes ou no
{{ var is null }}        // Affiche true ou false
```

1.5. La boucle FOR: {% for … %} {% endfor %}

- Afficher les chiffres de 0 à 9 :

```
{% for i in 0..9 %}  // pareil que {% for i in range(0, 9) %}
   {{ i }}
{% endfor %}
```

- Afficher la liste des produits :

```
{% for produit in produits %}
   {{ produit.nom }}
{% endfor %}
```

- Avec une condition :

```
{% for produit in produits if produit.etat==1 %}
   {{ produit.nom }}
{% endfor %}
```

- Boucle *for* avec une condition vide :

```
{% for article in articles %}
    {{ article.nom }}
{% else %}
    pas d'article trouvé
{% endfor %}
```

- Clés et valeurs :

```
{% for key, value in table %}
   {{ key }} {{ value }}
{% endfor %}
```

- Afficher les 10 premiers users :

```
{% for user in users|slice(0, 9) %}
   <li>{{ user.username }}</li>
{% endfor %}
```

La variable **loop** de la boucle for :

```
{{ loop.index }}     // Numéro de l'itération courante en commençant par 1
{{ loop.index0 }}  // Numéro de l'itération courante en commençant par 0
{{  loop.revindex  }}  //  Nombre  itérations  restantes  avant  la  fin  en
commençant par 1
{{  loop.revindex0  }}  //  Nombre  itérations  restantes  avant  la  fin  en
commençant par 0
{{ loop.first }}        // La première itération?  True ou false
{{ loop.last }}            // La dernière itération? True ou fals
{{ loop.length }}        // Nombre total d'itérations
```

1.6. Commentaire : {# … #}

```
{# mon commentaire #}
```

1.7. Les filtres

Un filtre agit comme une fonction. Ils sont déjà définis mais vous pouvez créer les vôtres si vous le souhaitez. Pour appliquer un filtre à une variable, il faut séparer le nom de la variable et celui du filtre par un *pipe* (|).

- upper : met la chaîne de caractères qui le concerne en majuscule ;
- lower : fait exactement l'inverse d'upper et va mettre toutes les lettres en minuscule ;
- title : met la première lettre de chaque mot de la chaîne de caractère en majuscule.
- length : retourne le nombre de caractères dont la chaîne se compose ;
- escape ou e : pour échapper du code HTML vous pouvez utiliser escape ou e.

Exemple :

Twig	PHP

{% for i in 0..produits|length %}
 	
{% endfor %}
	<?php
	for ($i=0;$i<count($produits);$i++)
	{
	}
	?>

{{ "bonjour dsi"|trim|upper }} Ou {% filter upper %} Bonjour dsi. C'est la troisième séance. {% endfilter %}	<?php echo strtoupper(trim("bonjour dsi")); ?>

1.8. Les fonctions

Une fonction effectue un traitement (contrairement à un filtre qui est destiné à faire une conversion).

Exemple :

- La fonction dump : affiche les détails d'un objet ou d'un tableau.

```
{{ dump(produit) }}
```

- La fonction max : retourne le max d'une série de valeurs.

```
{% set tab = [1, 5, 2, 3] %}
{{ max(tab) }}              // affiche 5
```

1.9. Inclure des templates (Symfony)

Inclure des fichiers template :

```
{% include('template.html') %}
```

1.10. Extension

extends peut être utilisée pour étendre un modèle à partir d'un autre :

```
{% extends 'header.html' %}
```

1.11. Render controller (Symfony)

Faire appel à une action du contrôleur.

Exemple *:* récupérer les trois derniers articles les plus récents :

```
{{ render(path('latest_articles', {max: 3})) }}
{{ render(url('latest_articles', {max: 3})) }}
```

1.12. Les liens

- Vers fichiers assets :

```
<img src="{{ asset('images/logo.png') }}" />
```

- Vers url / :

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

- Vers url / avec des paramètres :

```
<a href="{{ path('article_show', { 'slug': article.slug }) }}">
      {{ article.title }}
</a>
```

- Vers url absolue :

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

1.13. Débogage

La commande *lint:twig* vérifie que vos modèles Twig n'ont pas d'erreurs de syntaxe :

- Vérifier tous les templates :

```
php bin/console lint:twig
```

- Vérifier un dossier ou un template :

```
$ php bin/console lint:twig templates/email/
$ php bin/console lint:twig templates/article/recent_list.html.twig
```

2. Travail à faire

Exercice 1

➢ **Installation**

Commencez par l'installation de TWIG:

composer require twig

➢ **Contrôleur**

Changez votre contrôleur (créé dans le tp 2) pour hériter de **AbstractController** :

```
<?php
  namespace App\Controller;
  use Symfony\Component\HttpFoundation\Response;
  use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
  use Symfony\Component\Routing\Annotation\Route;
```

```php
class HelloController extends AbstractController
{
  /**
   * @Route("/hello")
   */
  public function sayHello()
  {
    return new Response('Hello!');
  }
  /**
    * @Route("/bonjour/{nom}")
    */
  public function bonjour($nom)
  {
    //return new Response("Bonjour $nom !");
    return $this->render('bonjour.html.twig', ['nom' => $nom,]);
  }
}
```

- **Template twig**

Mettez en place le template correspondant **bonjour.html.twig** dans le dossier « templates » :

```twig
{# templates/bonjour.html.twig #}
{% extends 'base.html.twig' %}
{% block body %}
    <h1>Bonjour {{ nom }}</h1>
{% endblock %}
```

Le fichier **base.html.twig** :

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

Exercice 2

On veut afficher le nombre de rendez-vous, pour un docteur, chaque jour durant une semaine.

1. Définissez un template de base, **base.html** :

```twig
<!DOCTYPE html>
<html lang="fr">
<head>
```

```
    {% block head %}
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.css" />
    <title>{% block title %}{% endblock %}</title>
    {% endblock %}
</head>
<body>
    <section id="content">
      {% block content %}{% endblock %}
    </section>
<footer id="footer">
    {% block footer %}
        Copyright &copy;
        <script>
            document.write(new Date().getFullYear());
        </script>
        All rights reserved
    {% endblock %}
</footer>
</body>
</html>
```

1. Définissez un template qui hérite de template de base, **rendezvous.html.twig** et qui permet d'afficher le nombre de rendez-vous (stocké dans un tableau) durant une semaine.
2. Enfin, utilisez ce template dans un contrôleur **DocteurController.php**.
   Utilisez un tableau qui donne un nombre quelconque pour chaque jour.